University of Alberta

Testing the Internet State Management Mechanism

by

Andrew F. Tappenden

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Spring 2010
Edmonton, Alberta

Examining Committee

James Miller, Electrical and Computer Engineering

Vicky Zhao, Electrical and Computer Engineering

Mrinal Mandal, Electrical and Computer Engineering

Eleni Stroulia, Computer Science

Frank Maurer, Computer Science, University of Calgary

Abstract

This thesis presents an extensive survey of 100,000 websites as the basis for understanding the deployment of cookies across the Internet. The survey indicates cookie deployment on the Internet is approaching universal levels. The survey identifies the presence of P3P policies and dynamic web technologies as major predictors of cookie usage, and a number of significant relationships are established between the origin of the web application and cookie deployment. Large associations are identified between third-party persistent cookie usage and a country's e-business environment.

Cookie collection testing (CCT), a strategy for testing web applications, is presented. Cookies maintained in a browser are explored in light of anti random testing techniques, culminating in the definition of seeding vectors as the basis for a scalable test suite. Essentially CCT seeks to verify web application robustness against the modification—intentional or otherwise—of an application's internal state variables. Automation of CCT is outlined through the definition of test oracles and evaluation criterion.

Evolutionary adaptive random (eAR) testing is proposed for application to the cookie collection testing strategy. A simulation study is undertaken to evaluate eAR against the current state-of-the-art in adaptive random testing—fixed size candidate set, restricted random testing, quasi-

random testing, and random testing. eAR is demonstrated to be superior to the other techniques for block pattern simulations. For fault patterns of increased complexity, eAR is shown to be comparable to the other methods.

An empirical investigation of CCT is undertaken. CCT is demonstrated to reveal defects within web applications, and is found to have a substantial fault-triggering rate. Furthermore, CCT is demonstrated to interact with the underlying application, not just the technological platform upon which an application is implemented. Both seeding and generated vectors are found to be useful in triggering defects. A synergetic relationship is found to exist between the seeding and generated vectors with respect to distinct fault detection. Finally, a large significant relationship is established between structural and content similarity measures of web application responses, with a composite of the two similarity measures observed to be superior in the detection of faults.

Acknowledgments

Dr. James Miller is to be commended for his support as a teacher, supervisor, mentor, and friend. His patience and guidance were pivotal in the development and construction of my dissertation, and without his support this work would not have been possible. He is an excellent supervisor and even better on the squash court—where I have yet to best him. For all these things I will always be grateful—thanks James.

Without question Kristen Tappenden requires commendation for her contributions to this work. As my de facto editor she has graciously provided countless hours of tedious work primping and polishing my manuscripts. More than just a pretty face and a key eye for grammatical errors, she has given me the emotional support and guidance which only a partner can provide, and I am a better man for it.

To my supportive parents, Alan and Connie Tappenden, I would like to acknowledge the decades of encouragement and belief in my ability. It is because of you that I have had the opportunity to pursue higher education, and for this I very am grateful. I would also like to extent my gratitude to the Tappenden and Gruber families for their support and encouragement.

Funding for this research was provided by the Natural Sciences and Engineering Research Council (NSERC) of Canada and iCORE. The Department of Electrical and Computer Engineering at the University of Alberta is to be acknowledged for the numerous teaching opportunities provided during the tenure of my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last 20 years there has been a major shift in the computing paradigm, driven primarily by the widespread adoption of the Internet and associated technologies. The Internet allows for the exchange of information via the HyperText Transfer Protocol (HTTP). In response to the stateless nature of HTTP requests and the growing complexity of web applications, a HTTP state management mechanism has evolved (Kristol, 2001; Kristol & Montulli, 1997, 2000). The fundamental component of this mechanism is the HTTP cookie (cookie).

Since their inception in 1994[1], cookies have existed in the realm of implementation-specific details, rather than within a state of formal specification. The evolution of the cookie specification from the initial 1994 Netscape implementation (Netscape Communications Corporation, undated) to the most recent cookie specification, RFC 2965 (Kristol & Montulli, 2000), published in 2000, has been chronicled by Kristol, one of the principle authors and participants in the formal definition of the cookie mechanism (2001). Although a formal specification has existed for cookies since RFC 2109 (Kristol & Montulli, 1997), cookie technology continues to evolve, further distancing implementation details from the specifications. Since the release of the RFC 2965 specification, many user-agents (browsers) have been reluctant to adhere to the most recent specification, primarily due to the lack of adoption by the two major players—Internet Explorer (IE) and Firefox (Mozilla / Netscape). In fact,

[1] Cookies first appeared in the initial public release of the Netscape Navigator browser in September of 1994 (Kristol, 2001).

RFC 2965 has yet to find widespread adoption a full eight years after its release. Further addendums, such as the *HttpOnly* option introduced by Microsoft as part of IE version 6 Service Pack 1 (IE6 SP1) (Microsoft Corp., 2002, 2007), have been proposed and implemented, further distancing current implementations from the Internet Engineering Task Force (IETF) specification.

Table 1-1. Cookie implementation and browser version

Browser Version	Netscape (1994)	RFC 2109 (1997)	RFC 2965 (2000)	HttpOnly (2002)
IE5	☑	☑	☒	☒
IE6	☑	☑	☒	☒
IE6 SP1	☑	☑	☒	☑
IE7	☑	☑	☒	☑
IE8	☑	☑	☒	☑
Firefox < 2.0.0.5	☑	☑	☒	☒
Firefox ≥ 2.0.0.5	☑	☑	☒	☑
Firefox 3.0	☑	☑	☒	☑
Firefox 3.5	☑	☑	☒	☑
Netscape 9.0b3	☑	☑	☒	☑
Safari 4.0.4	☑	☑	☒	☑
Opera 10	☑	☑	☑	☑

To date the only major browser currently supporting RFC 2965 is Opera, which based upon several recent polls has less than 1% market share (Net Applications, 2007; OneStat.com, 2007; The Counter.com, 2007). Despite the lack of support for RFC 2965, the *HttpOnly* addendum has found widespread adoption amongst all recent browsers. Table 1-1 provides a summary of cookie support amongst various browsers; it is clear that the cookie technological landscape is anything but straightforward. The incomplete specification and fluctuating

implementations of cookies within browsing environments raises a number of important testing issues. This dissertation seeks to provide a clear picture of cookie usage across the Internet, and offer an effective testing strategy for web applications based around the HTTP state mechanism.

1.1 Thesis Contributions

It is our contention that cookies have been generally ignored and frequently omitted from many discussions within the literature, specifically in regards to the testing of web applications. Current discussions have focused primarily on privacy concerns related to cookie usage and the proposal of new permutations of cookie technology (Alvin, 2004; Juels, Jakobsson, & Jagatic, 2006; Park & Sandhu, 2000; Samar, 1999). Discussions focusing on the testing of web applications that include an explicit mention of the HTTP state mechanism are sparse.

This thesis presents the results of an investigation undertaken to highlight the prevalence of cookie deployment on the Internet, and provide an analysis of the heightened complexity associated with the verification and validation activities of applications that utilize cookies. The need for testing strategies that explicitly acknowledge cookies and the increased complexity that they encapsulate will be established, and a novel testing theory that is empirically demonstrated to detect faults within real-world web applications will be presented.

1.2 Thesis Outline

The remainder of this dissertation will be organized as follows. Chapter 2 will present a definition of a cookie and an overview of the relevant works within the literature. Chapter 3 will present the results of an extensive survey that was undertaken as part of the current study, revealing the nature of cookie deployment across the Internet. Chapter 4 will provide further insights into current cookie usage with specific emphasis on country of origin and cookie-specific technologies. Chapter 5 will present

a testing strategy for testing cookie usage within web applications, with conclusions drawn from the surveys described in previous chapters. Chapter 6 will present a novel evolutionary adaptive random testing strategy that can be used as the basis for cookie collection testing. Chapter 7 presents the results of an empirical evaluation undertaken to assess the ability of the testing theory to detect faults within real-world web applications. Finally, Chapter 8 summarizes the major contributions of this dissertation and recommends avenues of future research directly related to this work.

Chapter 2

An Overview of Cookies, Web Application Testing, and Adaptive Random Testing[2]

The Internet has quickly become an indispensable medium upon which many facets of modern economies have come to rely. The widespread adoption of the Internet has enabled the emergence of a truly global economy, ubiquitously connecting people and businesses from all countries. Cookies, one of the fundamental Internet technologies, were integral in the establishment of e-commerce, revolutionizing both the way users could interact with web-applications, and the way in which these systems were developed. Cookies enabled e-commerce sites to provide a state-based shopping experience to users, hence the omnipresent shopping cart metaphor. Although development of these types of applications was possible with previous technologies, cookies offered a simple, robust, built-in alternative on which the e-commerce industry could build applications through a single technology.

The remainder of this chapter will be organized as follows: Section 2.1 will provide a definition of the state-of-the-art regarding cookies. Section 2.2 will outline other investigations undertaken to understand the deployment of web-specific technologies across the Internet. Section 2.3 will discuss related work and motivations for the investigation with regard

[2] *A version of this chapter has been published. Tappenden, A. F., & Miller, J. (2008). A Three-Tiered Testing Strategy for Cookies. Paper presented at the Software Testing, Verification, and Validation, 2008 IEEE International Conference on.*

to testing web applications. Section 2.4 will outline the current state-of-the art in regards to anti random and adaptive random testing strategies. Finally, Section 2.5 will summarize the key motivations for the research undertaken within this dissertation.

2.1 Cookies: Defining the HTTP State Mechanism

The HTTP cookie was first introduced as a feature within the initial public release of Netscape Navigator in September of 1994 (Kristol, 2001). Since Netscape's release, cookies have become an integral part of many web applications (Kristol, 2001; Kristol & Montulli, 1997, 2000). Cookies have existed in a state of vendor-specific implementations, and while attempts have been made to standardize the usage of this technology across technological platforms—RFC 2109 (Kristol & Montulli, 1997) and RFC 2965 (Kristol & Montulli, 2000)—standard compliance has yet to be achieved.

All active cookie specifications outline two complementary HTTP headers, *Set-Cookie* and *Cookie*[3], set by the server and client respectively. These two headers work in conjunction to provide four basic cookie operations to the application: create, read, update, and delete. Create, update, and delete are all achieved through the *Set-Cookie* header, which is used by a HTTP server to store a *name=value* pair (a cookie) on a client machine. Figure 2-1 provides an EBNF representation (ISO/IEC, 1996) of the grammar for the *Set-Cookie* header found amongst most browsers. This grammar is an adaptation and amalgamation of those found in Netscape's original HTTP Cookie Specification (NS Cookie) and RFC 2109 (Kristol & Montulli, 1997; Netscape Communications Corporation, undated), and represents the majority of browser implementations. RFC 2965 is not included in the definition provided in Figure 2-1 because it is largely ignored by the majority of browser implementations (Net

[3] RFC 2965 defines a *Set-Cookie2* and *Cookie2* headers to be used in place of or next to the *Set-Cookie* and *Cookie* headers.

Applications, 2007; OneStat.com, 2007; The Counter.com, 2007). The grammar presented does include the newest addition to cookie implementation, the HttpOnly (Microsoft Corp., 2002) addendum, as it is beginning to gain widespread adoption and is used by a substantial number of web-applications.

Set-Cookie is a field implemented within an HTTP header sent from the server to the client as a response to a client request. This header is subject to the grammatical rules of an HTTP header, as outlined in RFC 2616 (Fielding et al., 1999). One such provision that is assumed throughout the discussion is that any terminal (value between "") is considered to be case-insensitive unless otherwise stated. The definitions of an *attribute*, *value*, *date,* and *DIGIT* (**boldface** non-terminals from Figure 2-1 and Figure 2-2) are taken directly from RFC 2616. Definitions for the *attribute* and *value* non-terminals are provide in Figure 2-3, and *date* and *DIGIT* non-terminals in Figure 2-4. Together these figures provide an EBNF grammatical definition of the *Set-Cookie* header field which has been adapted from RFC 2109 (Kristol & Montulli, 1997), NS Cookie (Netscape Communications Corporation, undated), and the HttpOnly (Microsoft Corp., 2002) addendum, to accurately reflect the current status of cookie implementations across the majority of browsers.

```
set-cookie        = "Set-Cookie: ", cookie;
cookie            = name, "=", value, {"; ",cookie-av};
name              = attribute;
cookie-av         = ("comment=",value)
                  | ("domain=", value)
                  | ("max-age=", value | "expires=", date)
                  | ("path=", value)
                  | ("secure")
                  | ("version=",{DIGIT}+)
                  | ("httponly");
                  (* only one of each attribute=value (av)
                  can be included *)
```

Figure 2-1. Definition of the Set-Cookie Header

An EBNF definition of the *Cookie*e* header field is provided in Figure 2-2. Similar to the *Set-Cookie* definition, the *Cookie* definition also relies upon the definition of *attribute* and *value* from RFC 2616 (Figure

2-3). Like the *Set-cookie* header, the *Cookie* header presented in Figure 2-2 is an amalgamation of the RFC 2109 (Kristol & Montulli, 1997) and NS Cookie (Netscape Communications Corporation, undated) specifications; however, as the HttpOnly (Microsoft Corp., 2002) specification does not change the *Cookie* header, it is not relevant to the definition of the *Cookie* header field.

```
cookie              = "Cookie:", [cookie-version],
                    cookie-value, {(";" | ","),
                    cookie-value};
cookie-version      = "$version", "=", value, (";" | ",");
cookie-value        = name, "=", value, [";", path],
                    [";", domain];
name                = attribute
path                = "$path", "=", value
domain              = "$domain", "=", value
```

Figure 2-2. Definition of the Cookie Header

```
attribute           = token;
value               = token | quoted-string;

token               = {CHAR – (tspecials | CTLs)}
tspecials           = "(" | ")" | "<" | ">" | "@" | ","
                    | ";" | ":" | "\" | <"> | "/" | "["
                    | "]" | "?" | "=" | "{" | "}" | SP
                    | HT;

quoted-string       = <">,{qdtext | quoted-pair},<">;
qdtext              = TEXT - <">;
quoted-pair         = "\", CHAR

TEXT                = OCTET – (CTL – LWS);
LWS                 = [CRLF], {SP | HT}+;
OCTET               = ? any 8-bit sequence of data ?;
CHAR                = ? any US-ASCII character (0 - 127) ?;
ALPHA               = ? any ASCII alphabetic character ?
SP                  = ? US-ASCII SP, space (32) ?;
<">                 = ? US-ASCII double-quote mark (34) ?;
CTL                 = ? any US-ASCII control character
                    (0 - 31) and DEL (127) ?;
HT                  = ? US-ASCII HT, horizontal-tab (9) ?;
```

Figure 2-3. Definitions of a*ttribute* and *value* (Fielding et al., 1999)

```
date                = rfc1123-date | rfc850-date
                    | asctime-date;
rfc1123-date        = wkday,",",SP,date1,SP,time,SP,"GMT";
rfc850-date         = weekday,",",SP,date2,SP,time,SP,"GMT";
asctime-date        = wkday,SP,date3,SP,time,SP,4*DIGIT;
date1               = 2DIGIT, SP, month, SP, 4*DIGIT;
                    (* day month year (e.g., 02 Jun 1982) *)
date2               = 2*DIGIT, "-", month, "-", 2*DIGIT;
                    (*day-month-year (e.g., 02-Jun-82) *)
date3               = month, SP, ( 2*DIGIT | ( SP, DIGIT ));
                    (* month day (e.g., Jun 2) *)
time                = 2*DIGIT, ":", 2*DIGIT, ":", 2*DIGIT;
                    (*00:00:00 - 23:59:59*)
wkday               = "Mon" | "Tue" | "Wed" | "Thu" | "Fri"
                    | "Sat" | "Sun";
weekday             = "Monday" | "Tuesday" | "Wednesday"
                    | "Thursday" | "Friday" | "Saturday"
                    | "Sunday";
month               = "Jan" | "Feb" | "Mar" | "Apr" | "May"
                    | "Jun" | "Jul" | "Aug" | "Sep" | "Oct"
                    | "Nov" | "Dec" ;
DIGIT               = "0" | "1" | "2" | "3" | "4" | "5"
                    | "6" | "7" | "8" | "9"
SP                  = ? US-ASCII SP, space (32) ?
```

Figure 2-4. Definitions of *date* and *DIGIT* (Fielding et al., 1999)

The *Set-Cookie* header has a number of optional attribute-value pairs (*cookie-av* in Figure 2-1) that can be associated with a cookie. According to the specifications, the ordering of the pairs is inconsequential; however, only one of each attribute-value pair can be present for a given cookie. These pairs, often defined by the non-terminal *value* within the original specifications (Kristol & Montulli, 1997, 2000; Netscape Communications Corporation, undated), require further definition. According to RFC 2109, the *version={DIGIT}+* attribute-value pair is the only required *cookie-av* in the *Set-Cookie* header. However, in practice this value is not required due to the overlapping implementations of the NS Cookie and RFC 2109 specifications, and is optional in the definition provided in Figure 2-1. The *version* attribute-value pair is passed with a cookie from the server to a user-agent to identify that the cookie follows the standard outlined in RFC 2109. This requires that the specific pair *version=1* be passed along with a given cookie. Nonetheless,

due to the interwoven nature of the standards and implementations, this field is generally ignored.

The domain attribute appears in the form *domain=value*. With this option, a server can specify the domain for which a cookie is valid; the specified domain must *domain match* the request URI (Kristol & Montulli, 1997). Moreover, the specified domain must be contained within the request domain, i.e. *x.y.com* domain matches *.y.com* but *y.com* does not match *.x.y.com*. Domain matching has been seen as a difficult procedure to implement, due to such external factors as the varying length of top-level domains, e.g. *.com* and *.co.uk*, and the prospect of multiple servers existing within a single domain. These problems have yet to be resolved (Kristol, 2001). However, for the purpose of this discussion we will ignore the domain matching aspect of the input validation and focus primarily on the underlying grammar of the attribute-value pair. If the domain is explicitly set, that is the *domain=value* pair is specified for a given cookie, the *value* must start with a period, followed by two or more alphanumeric strings separated by a single period, as defined in Figure 2-5. In the presence of the *domain=value* pair, a given cookie is valid within the explicitly defined domain (assuming that the value domain matches the request domain). Cookies set without an explicit *domain=value* pair will have the value set, by default, to the domain of the request URI.

```
domain-av          = "domain=", ( domain-value
                   | <">, domain-value, <">);
domain-value       = ".", alphanumeric, {".",
                   alphanumeric}};
alphanumeric       = {ALPHA|DIGIT}+
```

Figure 2-5. Definition of the *domain=value* Pair

```
max-age-av         = "max-age=", ( max-age-value
                   | <">, max-age-value, <">);
Max-age-value      = {DIGIT}+
```

Figure 2-6. Definition of the *max-age=value* Pair

The expiry attribute appears in one of two forms, either as *expires=date* or *max-age=value*. Although these two forms have different grammars, they both work to provide the same functionality. As specified previously, this attribute-value pair is optional, but only one occurrence of either of these two attribute-value pairs is valid. The e*xpires=date* pair represents a specific date at which a cookie is set to expire and is defined in Figure 2-1 and Figure 2-4. The *max-age=value* pair, on the other hand, defines the number of seconds that are to elapse before the cookie is no longer valid. This value is a non-negative integer and its grammar is presented in Figure 2-6. The presence of an expiry attribute-value pair (*max-age=value* or *expires=date*) defines the cookie as persistent (will be stored until the expiration), whereas its omission defines the cookie as a sessional cookie, i.e. the cookie will be deleted upon session termination (typically when a browser execution is terminated).

The *path=value* attribute-value pair exists to define for which subset of URLs a cookie is valid. Much like the domain attribute-value pair, the validity of the path value depends upon the original request URI. As noted by Kristol (Kristol, 2001), the process of validating the path value against the original URI is not strictly defined, and is outside the realm of a grammatical definition for the input. The grammar for the *path=value* attribute pair is provided in Figure 2-7. As with the other fields, the *path=value* pair is optional, and if it is not explicitly set, the path value will default to the path provided in the original request URI. The final attribute-value pair within the *Set-Cookie* header is *comment=value* pair. This pair is optional and has no functional value to the client or server, but exists to provide the end-user or developer with information about the cookie.

```
path-av           = "path=", ( path-value,
                  | <">, path-value, <">);
path-value        = {"/",alphanumeric}
Alphanumeric      = {ALPHA|DIGIT}+
```

Figure 2-7. Definition of the *path=value* Pair

```
httponly          = ε | "httponly";

secure            = ε | "secure";
```

Figure 2-8. Definition of *httponly* and *secure* Attributes

The final two *Set-Cookie* attributes do not have associated values and are defined in Figure 2-8. The *httponly* attribute, as previously noted, was defined by Microsoft as a deterrent to cookie theft through the use of malicious JavaScript, known as cross-site scripting (XSS) attacks. This attribute has precisely two values, *httponly* or *ε* (not present) and indicates to the browser whether or not the *document* object within any embedded JavaScript should be granted access to this cookie and its associated values. As the latest addendum to the cookie specification, this attribute is not supported by all browsers, and its presence could cause a cookie to be rejected by a strict user-agent. The *secure* attribute, much like the *httponly* attribute, does not have an associated value and it provides instruction to the browser via its presence or omission. The presence of this attribute instructs the browser to only send this cookie over "secure" channels. The concept of what constitutes a "secure" cookie was defined in the NS Cookie specification (Netscape Communications Corporation, undated) as being any HTTP over SSL (HTTPS) connection. However, as of RFC 2109, the notion of a "secure" channel has become more ambiguous, with the idea being that the user-agent should allow the user to define which channels are secure (Kristol & Montulli, 1997). Although this ambiguity was introduced into the cookie specification, it has largely been ignored in favor of the original definition of HTTPS as a secure connection.

The *Cookie* header, defined in Figure 2-2, is the primary way in which cookies are read by a web-application. This header is passed by the client to the server as part of an HTTP request and contains several attribute-value pairs. Similar to the *Set-Cookie* header, these pairs are almost identical to the *version=value*, *domain=value,* and *path=value*

pairs defined in Figure 2-1, Figure 2-5, and Figure 2-7 respectively, except that a *$* character is inserted at the beginning of the attribute to indicate that the value is not a cookie but a cookie attribute (see Figure 2-2). The primary function of this header is to provide the required *name=value* pairing to the application. The grammatical definition for this pairing provides very little insight into the values that can be passed within a cookie, and the majority of these values adhere to application-specific constraints which are not recoverable from the specifications.

Together, the *Set-Cookie* and *Cookie* headers define and maintain a collection of cookies within a browser. This collection of cookies is accessible to a web application based upon six criteria: domain, path, time, browser session, browser environment, and connection type. These criteria relate directly to four of the attribute-value pairs defined within the cookie grammar: *domain*, *path*, *expires*/*max-age*, *httponly*, and *secure*.

2.2 Current Internet Surveys

In recent years, several studies of cookie usage have been conducted. Unfortunately, many of these studies provide incomplete information due to limitations of the research. Security Space, for example, produces a monthly cookie usage report (2006a), but explicitly states that the survey only considers the request for the HTML document—it does not load or execute any images, scripts or other objects embedded within the page. Because of this omission, the results obtained from the Security Space survey do not accurately reflect the current technological landscape of modern cookie deployment within complex multifaceted web applications. To further understand the ways in which cookies are currently being deployed across the Internet, a survey that fully loads and executes all objects within a webpage is required.

Reay et al. (2006; 2007) conducted two investigations studying the adoption of the P3P privacy policy across a large cross-section of the

Internet. These studies incorporated the P3P implementation statistics collected from a pool of 100,000 sites, complete with geographic information, providing a breakdown of P3P usage on a nation-by-nation basis. Although the primary concern of P3P policies is in relation to cookie usage (Cranor et al., 2006; W3C, 2006), these surveys unfortunately did not provide any insight into the cookie usage of the web applications surveyed.

Other studies have explicitly examined the usage of dynamic web technologies such as ASP, Cold Fusion, JSP, and PHP. A recent study conducted by Port80 Software examined the HTTP Header signatures from the Fortune 1000 Companies websites (Port80 Software, 2007). While this study provides insight into the distribution of dynamic web frameworks within the web, the sample size is limited and the technology detection mechanisms heavily favor dynamic-web technologies that come bundled with backend server software, such as ASP and JSP. Another relevant survey conducted by Doyle and Lopes (2008) examined the breadth of dynamic web technologies currently employed by the web community. While the survey provides an extensive list of potential technological platforms, it provides no insight into the deployment of these technologies.

2.3 Current Web Testing Strategies

While there have been several web application testing strategies proposed in the literature (Andrews, Offutt, & Alexander, 2005; Bellettini, Marchetto, & Trentini, 2005; Di Lucca, Fasolino, Faralli, & De Carlini, 2002; Elbaum, Rothermel, Karre, & Fisher Ii, 2005; Kung, Liu, & Hsia, 2000; Jeff Offutt & Wu, 2009; J. Offutt, Wu, Du, & Huang, 2004; Ricca & Tonella, 2001; Tappenden, Beatty, Miller, Geras, & Smith, 2005; Tonella & Ricca, 2004; Xu, Xu, & Jiang, 2005), there has been no proposal that adequately addresses the specific complexities resulting from the use of cookies in web applications. The strategies proposed by Andrews et al.

(2005), Kung et al. (2000), and Xu, Xu, and Jiang (2005) simply do not address the role or effect of cookies in modern web applications. Strategies outlined by Bellettini et al. (2005), Kung et al. (2000), Elbaum et al. (2005), Ricca, and Tonella (2001), and Offutt et al. (2009; 2004) acknowledge the existence and usage of cookies within web applications but do not incorporate this novel application state mechanism into their work. In a recent survey of the literature, Alalfi et al. (2009) compare a wide-array of modeling techniques for web applications, concluding that only three of the twenty-four models investigated consider cookies.

Three testing strategies in the literature were found to incorporate cookies to a greater extent. Tonella and Ricca suggest a two-layer model for the white-box testing of Web applications (2004), advocating the use of cookies as a mechanism within the testing strategy itself. Their method utilizes a cookie to record trace information for JavaScript code executing on the client machine. Although this represents a novel use of cookie technology, it does not advance the discussion regarding the testing of web applications utilizing cookies. However, Tonella and Ricca's work (2004) presents an interesting scenario in which cookies usage is not merely limited to the simple task of maintaining state. The cookies were not sent as part of an HTTP Header as initially defined by RFC 2109 (Kristol & Montulli, 1997) and RFC 2965 (Kristol & Montulli, 2000), but rather as part of a script running on a client browser. This demonstrates that cookie usage and implementation has moved beyond a simple state token, and is becoming a complex programming paradigm unto itself.

Di Lucca, Fasolino, Faralli, and De Carlini (2002) outline a testing strategy that involves the creation of a series of decision tables describing test cases. This testing strategy and accompanying web application model acknowledges and incorporates cookies; however, the incorporation of cookies does not reflect the current utilization of cookies within modern web applications. Di Lucca et al. categorize cookies as either the “state before test” or “expected state after test” (2002). With this distinction in

the testing framework, the cookies are simply a resulting state, rather than a driving force behind the test cases. The strategy contains the implicit assumption that cookies only contain state information, and that modifying cookies only occurs once per HTTP GET request; that is, modifying cookies does not occur without the submission of a form or clicking of a hyperlink. In modern web applications, this assumption is not valid. Cookies can be created, modified or deleted in a variety of ways, as witnessed in the work of Tonella et al. (2004). Although the work of Di Lucca et al. (2002) makes provisions for cookies within the testing framework, it does not provide an adequate testing strategy for dealing with the utilization of cookies in modern web applications.

Tappenden et al. (2005) describe a security testing strategy for web applications that explicitly includes cookies. However, as the testing strategy is constrained to security aspects of web applications, it does not address a large number of functional aspects of web applications, and is by no means comprehensive. Notwithstanding, the testing strategy proposed by Tappenden et al. (2005) does highlight a major concern overlooked by testing strategies proposed by Tonella et al. (2004) and Di Lucca et al. (2002)—the potential that exists for the malicious exploitation of cookies. Because of their prominent use for session identification, cookies pose a significant threat to security for any web application. Cross site scripting (XSS) is a very common type of attack executed on web applications and "[t]he most common behavior of XSS attacks … is to gather cookies" (Cook, 2003). With ten to twenty-five XSS exploits found in commercial web applications each month (Cgisecurity.com, 2002) and the prevalence of cookie gathering, incorporating cookies into a testing strategy for web applications becomes an essential security priority. Furthermore, with a new brand of exploitation coined "Cross Site Cooking" (Zalewski, 2006), the HTTP state mechanism is quickly becoming a major security liability for web applications.

2.4 Anti and Adaptive Random Testing

Anti random and adaptive random testing methodologies will be explored in this thesis as a potential solution for the testing of cookies within web applications. Since the inception of adaptive random testing (ART) in 2001 (T. Y. Chen, Tse, & Yu), a number of ART methods have been proposed (T. Y. Chen, De Hao, Tse, & Zongyuan, 2007; T. Y. Chen, Kuo, & Liu, 2007; T. Y. Chen, Kuo, Merkel, & Ng, 2003; T. Y. Chen, Leung, & Mak, 2004; T. Y. Chen & Merkel, 2006; T. Y. Chen, Merkel, Wong, & Eddy, 2004; T. Y. Chen et al., 2001), including most recently quasi-random testing (T. Y. Chen & Merkel, 2007; Chi & Jones, 2006). ART methodologies seek to increase the effectiveness of random testing by spreading the test cases evenly across the input domain. This goal, similar to that of the Anti Random testing algorithm (Malaiya, 1995), involves the random generation of test cases with a selection criterion used to evaluate the best available candidate.

Anti random testing prescribes the selection of test cases to maximize the Cartesian or Hamming distance from all previous test cases and has been shown to be effective through a series of empirical evaluations (Malaiya, 1995; von Mayrhause, Chen, Hajjar, Bai, & Anderson, 1998; Yin, Lebne-Dengel, & Malaiya, 1997). Anti Random testing explicitly seeks to select test cases that are as far away as possible from all previous cases. This method shares a number of similarities with ART. However, the distinct difference is that anti random testing does not contain random elements. Currently, an effective, scalable implementation is not available. Furthermore, not only is the anti random testing method computationally expensive, for any real-world testing scenario, it is simply intractable. Although attempts have been made to correct this bottleneck (von Mayrhause et al., 1998), these methods remain ineffective when the input-space is "balanced", resulting in random test case selection.

The basis for ART methodologies is rooted in the observation that faults often occur within *failure regions*, or *error crystals* within the input domain (Ammann & Knight, 1988; F. T. Chan, Chen, Mak, & Yu, 1996; T. Y. Chen & Merkel, 2007; Finelli, 1991). The goal of anti random testing, ART methodologies and quasi-random testing is to spread the test cases evenly across the input domain to increase the likelihood of triggering a fault within the application. A recent empirical analysis of thirteen ART strategies (Mayer & Schneckenburger, 2006) concluded that Fixed Size Candidate Set (FSCS) (F. T. Chan et al., 1996; T. Y. Chen, Leung et al., 2004; T. Y. Chen et al., 2001), and Restricted Random Testing (RRT) (K. P. Chan, Chen, & Towey, 2002) were the two most effective ART methods currently available. Accepting these findings, these two methods along with the quasi-random Sobol sequence are henceforth considered to represent the state-of-the-art in adaptive random testing.

FSCS testing is an ART strategy that uses a distance-based selection criterion to evaluate a fixed set of randomly generated test case candidates (T. Y. Chen, Leung et al., 2004; T. Y. Chen et al., 2001). An initial test case is selected at random, and for each subsequent test a group of k candidates are generated from which the candidate with the maximum-minimum distance from any other existing test case is selected. As the size of the test set grows, so does the computational requirement for each subsequent test, as each candidate is evaluated against all existing tests. For a test set with n elements, the time required to generate a subsequent test, $FSCS_{n+1}(k,n)$, grows linearly with the size of the test set, and $FSCS_{n+1}(k,n) \in O(k \cdot n)$. However, k is fixed; and as n grows, the value of k becomes insignificant. Thus, $FSCS_{n+1}(k,n) \in O(n)$. The execution of this method, when applied to the generation of a sequence of n test cases, results in an execution time of $FSCS(k,n) \in O(k \cdot n(n+1)/2)$, or $FSCS(k,n) \in O(n^2)$. Hence, the algorithm executes within the order of quadratic time. Optimization techniques, such as mirroring, have been introduced to reduce the computational footprint of FSCS (T. Y. Chen et al., 2003).

However, even with these techniques, execution of the algorithm remains within the order of quadratic time.

Like FSCS, RRT uses a distance-based selection criterion in the selection of subsequent test cases. However, instead of a fixed set of candidates from which to choose the next test, RRT candidates are generated until one lies outside of the exclusion zone present around each existing test (K. Chan et al., 2002; K. P. Chan, Chen, Kuo, & Towey, 2004; T. Y. Chen, De Hao et al., 2007). The size of the total exclusion zone (including overlapping zones) for the tests within the test set is proportional to the input domain, and is defined by the constant R. Because this value remains constant, the size of the exclusion zone around any test shrinks as the number of tests, n, within the test set increases. Because RRT does not have a fixed candidate set, and relies upon repetitious random selection for the generation of subsequent tests, calculation of the algorithm runtime is not meaningful. In the worst case, the algorithm can become unresponsive if the entire input domain is excluded, or if randomly generated tests all lie within an exclusion zone. Despite these worst-case possibilities, empirically the algorithm has been demonstrated to perform on average within $RRT(n) \in O(n^2 \log(n))$ (Mayer & Schneckenburger, 2006).

Quasi-random sequences are a set of mathematical sequences that are characterized by the *low-discrepancy* property, providing a sequence with a uniform distribution of values. The low-discrepancy property exhibited by quasi-random sequences ensures that these sequences are constructed as evenly-spaced as mathematically possible. One such algorithm is the Sobol sequence, as defined by Sobol (1967), and recently proposed for use within the field of software testing by Chi and Jones (2006). The computation of the Sobol sequence is outlined by Fox (1986), and can be implemented with marginal computational cost; i.e., $Sobol(n) \in O(s[(\log n)/\log q_s]^2)$, where n is the number of points to be generated, s is the dimensions of the hypercube for which the points are generated, and

q_s is the first prime number greater than or equal to s. This low computational cost is one of the primary benefits of quasi-random testing when compared to the increased computational complexity of ART methods. Despite this benefit, quasi-random testing has a number of considerable limitations, one of which is the inherent deterministic nature of quasi-random sequences. That is, the quasi-random sequences are not random, and subsequent execution of algorithms yield the same sequence. Attempts have been made to address this problem through the use of sequence scrambling (T. Y. Chen & Merkel, 2007). However, an adequate scrambling method that retains the low-discrepancy of the quasi-random sequence, and is computationally feasible, is not currently available for real-world testing applications (i.e. problems larger than a two-dimensional hypercube). Despite this disadvantage, quasi-random testing remains an effective tool in a testing environment where the computational cost of generating test inputs is of principal importance.

Genetic algorithms have been applied within a wide array of problem domains including software testing. The use of genetic algorithms for test input generation has been the focus of numerous approaches (Harman et al., 2004; Michael, McGraw, & Schatz, 2001; Xiao, El-Attar, Reformat, & Miller, 2007). However, unlike ART, these approaches are based upon white-box code coverage metrics as an assessment of fitness. Other adaptations within the software testing domain include, but are not limited to, the prioritization of pre-existing tests within a recursive test suite (Li, Harman, & Hierons, 2007), and the stress testing of distributed systems (Garousi, 2008; Garousi, Briand, & Labiche, 2008). Although the application of evolutionary algorithms to various problem domains is not unique, the application of this technique for the generation of ART input data is novel.

2.5 Motivations for Research

Web application and development practices continue to mature and are constantly evolving. To date, the cookie has played a crucial role in the development of the current e-commerce technological landscape. While much is still changing within the web development paradigm, including the recent development of frameworks such as AJAX, cookies remain fundamental to web applications. For example, consider Google's Gmail web application (2008). Despite its heavy reliance on the AJAX framework, Gmail was still observed to set 21 cookies. Clearly, cookies are essential even within an AJAX application. As these applications continue to mature, cookie usage is expected to remain vital, as it is critical that these applications are able to function across a wide array of system configurations and browser versions. Despite the heterogeneous nature of end-user web platforms, cookies remain a common denominator amongst all browsers. Given this reality, future web endeavors must thoroughly understand the role that cookies play in the technological landscape in which web applications exist.

As the primary state token for many applications, cookies serve as the glue holding internal components together. In addition to this fundamental operation, cookies are increasingly utilized in more sophisticated situations such as for compiling user statistics. This type of operation often involves multiple hosts and spans a time period of several years. Such usage of cookies is often combined with equally sophisticated third-party web applications, whose execution is hidden from end-users. Although third-party cookie deployment raises a number of privacy issues, it is now commonplace on the Internet. An increasing number of privacy-conscious users routinely delete cookies, which threatens the validity of the statistics collected by web applications (comScore Inc., 2007b). Furthermore, this habitual removal of cookies was found to affect first- and third-party cookies equally (comScore Inc., 2007b), posing a threat to all web applications utilizing cookies—not just advertising and traffic

management applications. In addition to users actively removing cookies, third-party software agents are being developed that actively reject all third-party cookies, and selectively reject first-party persistent cookies on the basis of perceived functionality (Yue, Xie, & Wang, 2007). These trends toward selective cookie rejection pose a significant challenge to the development of modern web applications—a challenge that is not adequately addressed by any of the current testing strategies.

Chapter 3

Cookies: A Deployment Study[4]

In order to obtain an accurate picture of cookie usage within web applications, a rigorous survey of the Internet is required. This chapter outlines the results of an extensive investigation undertaken to highlight the prevalence of cookie deployment on the Internet, and provide an analysis of the heightened complexity associated with their use. The Internet survey that was undertaken is unique in that it was executed within a web-browsing environment, allowing for full loading and execution of all components at each of the surveyed sites.

This chapter will demonstrate the need for testing strategies that explicitly acknowledge cookies and the increased complexity that they encapsulate. The remainder of the chapter will be organized as follows: Section 3.1 will outline the Internet survey that was conducted; Section 3.2 will present and discuss the significance of the results obtained; Section 3.3 will relate the results of the survey to a number of real-world examples, and discuss the principle testing implications; finally, Section 3.4 will summarize the chapter highlighting the key contributions of the survey.

3.1 Study Methodology

3.1.1 Site Selection

The Alexa top 100,000 list (2006b) was used as the starting point for the survey of cookie usage in web applications. This list was selected for two

[4] *A version of this chapter has been published. Tappenden, A. F., & Miller, J. (2009). Cookies: A Deployment Study and the Testing Implications. ACM Trans. Web, 3(3), 1–49.*

primary reasons. First, the Alexa top 100,000 list represents a very large sample population, which allows for an accurate and representative sample of current Internet sites. The second reason is that the list represents the most popular websites, based upon traffic information (Alexa Internet Inc., 2006a). This "popularity" measure is based upon both the page *views*, number of pages viewed on a host, and the *reach* or number of different users who access a host (Alexa Internet Inc., 2006a). The Alexa ranking is a based on the geometric mean of the *reach* and *views* quantities. Essentially, this ranking provides a list of the most popular 100,000 sites accessed on the Internet, comprising the vast majority of websites online users are likely to encounter. According to Alexa (2006a), websites excluded from this list have less than a 0.00125% chance of being accessed by an average Internet user. Hence this list may be considered a highly representative sample of the *usable* Internet. While the sampling bias of the Alexa rating is well documented, it remains in use because it is the best of the currently available web ranking data (Baker, 2007). Furthermore, the Alexa top 100,000 list has been used in other academic studies, such as the investigation undertaken by Reay, Beatty, Dick and Miller (2007) to study the adoption of the P3P privacy policy.

3.1.2 Firefox Extension

Due to the large sample population (100,000 sites), the survey required an automated data gathering mechanism. Hence, an extension for the Mozilla Firefox Internet browser (Mozilla Corporation, 2006) was produced. The extension accepts a list of URLs to survey, and then visits each URL, fully loading the page, associated images and JavaScript components. Once the page finishes loading, the extension records all of the cookie information. The data collected regarding individual cookies includes name, value, host, path, secure connection required and expiry date. These fields all directly correspond to the specification provided in both RFC 2109 and 2965 (Kristol & Montulli, 1997, 2000). Additional

information, such as creation time and associated URL was also recorded for each cookie. The extension used the browser to send a GET request to a web server specifically designed to parse the URL, and store the information for each site and cookie into a database for future analysis.

3.1.3 Study Implementation

Although the process was automated, supervision of the system was required due to the inability of the extension to deal with interactive client-side GUI events, such as dialog boxes. If a site were to prompt the user with a dialog box, the automated process would stall, and require external input to continue; as a policy, 'OK' was selected for all dialog boxes, and 'CANCEL' was selected for all login prompts. Other supervision was required in the case of sites crashing the web browser. If this occurred, the survey was re-started at the URL of the site that crashed the server, and the survey continued. Sites that could not be surveyed due to re-occurring crashing are not included in the results.

3.2 Discussion of Results

This section will discuss the major results and implications that were gathered from the survey. These results provide a basis for discussions into the complexity of modern web applications.

3.2.1 Success Rates

The study resulted in cookie usage information being collected for 98,006 individual websites. The data collected represents 98% of the original list; Table 3-1 provides a summary of the reasons why the remaining sites were excluded. As shown in Table 3-1 the majority of sites not surveyed contained missing DNS entries (50.5%), or incorrectly configured firewalls (31.5%). The 1,994 sites that could not be surveyed highlight the rapid turnover and uncertainty present on the Internet today. Web applications face intense time-to-market pressures and are often unprepared for incredibly high traffic volumes arising from growing Internet popularity.

In the face of such pressures, many web applications simply grow too fast and have to be abandoned or shut down for server maintenance or upgrades; however, due to the competitive nature of the Internet marketplace, web applications that cannot handle large traffic volumes are often unsuccessful in the long term.

Table 3-1. Inaccessible Sites

Reason	Number of Sites	Percentage
DNS not found	1,007	50.5%
Port Closed or Filtered	629	31.5%
Crashed Browser	95	4.8%
Server Timed Out	263	13.2%
Totals	1,994	100%

3.2.2 Number of Sites Using Cookies

The study conducted found that 67.4% (66,031) of the 98,006 Internet sites used cookies. This percentage is much higher than suggested by a previous study conducted by Security Space (2006a), which suggests that only 24.6% of sites utilize cookies. The discrepancy between the two studies arises from the differences in the survey methodologies utilized. The Security Space survey only looked for cookies from "web pages and not any images, applets, or other objects that may be contained in those pages" (2006a); that is, the survey did not fully load the web page and only the accessed the basic HTML source for the requested URL. This is in direct contrast to the methodology of the current investigation, in which the web page was completely loaded including not only the HTML, but all related images, JavaScript components, and other objects.

The most notable difference in the two approaches is that the current survey tabulates cookie usage arising from embedded images and JavaScript components. As an example of this difference, take for instance the web application accessed by the URL http://www.apple.com. At the time of the survey it was observed that the application utilized four cookies. Upon more detailed manual inspection, it was found that this web application would not have been tabulated if surveyed by the technique outlined in the previous survey (Security Space, 2006a).

Without loading any of the images, only two of four cookies were set on the client machine; accessing without JavaScript components resulted in only one of four cookies being present, and accessing without the JavaScript components and images resulted in zero of four cookies being present. Furthermore, when accessing the web application with JavaScript disabled, the application explicitly reported to the user that JavaScript must be enabled to properly view the page. This web application explicitly required JavaScript to function properly, and the survey presented by Security Space simply did not support this technology. In order to mitigate these false-negative identifications the rigorous survey methodology is required.

The current study only accessed the initial page for each URL surveyed. Some of the sites accessed were merely façades, providing a very simple interface to access subsequent dynamic content. Examples of this instance include splash pages that did not provide any dynamic functionality, but merely included links to the dynamic areas of the web application, and pages that provided a language selection interface to enable users to select a supported internationalization. These types of initial pages provide very limited functionality, and as a result some web applications that utilize cookies in subsequent browser interactions were not recorded. Because of these instances, the results obtained in the survey, notably that 67.4% of sites that utilize cookies, should be considered as a minimum bound, as the survey included both interactive dynamic web applications and static web application front-ends.

3.2.3 An In-Depth Look at the Number of Cookies per Site

An in-depth study of a smaller subset of web applications, probing deeper into each application could be the basis for future research. However, such an undertaking would require a very large pool of human and technological resources to achieve any reportable results in a feasible time schedule. As a proof of concept, our survey implementation was adapted

to survey a very small subset of the original sample space. This exploration focused on 40 sites selected randomly from the original sample space of 100,000. A list of subsequent URLs from each of the 40 sites was produced using a third-party web crawling tool (Kals, 2007). This tool was executed with the same domain criterion (only URLs from the same domain were followed) and a maximum crawling depth of 5 pages and running time of 10 minutes. Of the 40 sites crawled, 28 crawls were prematurely stopped due to the 10 minute maximum crawling time condition, and the average crawling time per site was 8 minutes. The initial list of 40 sites returned 23,990 further URLs for surveying. These 23,990 URLs were surveyed and compared against the baseline values for the site, provided by the survey of the initial page. A summary of the results is presented in Table 3-2. Although this analysis cannot be considered definitive due to the small sample size, the results suggest that the number of cookies present on the initial page account for 90% of the total cookies deployed by the web application. This number is derived from the addition of the percentage of sites that set an equal number of cookies and the number of sites that set less than the original site. It is important to note that although a specific URL may not set a cookie, it may still access the value stored in the cookie. Furthermore, cookie modification is only possible through the re-creation of a cookie. Therefore, each subsequent cookie recorded could represent either the modification of an existing cookie or the creation of a new cookie; this distinction was ignored as part of this proof-of-concept exercise. Further research into cookie usage within a single site is seen as a great complement to the study presented in this study, however the time requirements to perform such an in-depth study on a reasonably large cross-section of web applications is currently infeasible.

Table 3-2. In-Depth Crawling Results from 40 Individual URLs

Original URL	# of URLs	Original # of Cookies	URLs With More Cookies		URLs With Fewer Cookies		URLs With Equal Cookies	
http://webster.edu/	1365	0	62	5%	0	0%	1303	95%
http://sanyo-dsc.com/	347	0	1	0%	0	0%	346	100%
http://www.citibank.be/	947	0	495	52%	0	0%	452	48%
http://profitinfo.com/	95	0	19	20%	0	0%	76	80%
http://8minutedating.com/	771	0	97	13%	0	0%	674	87%
http://businesstown.com/	723	1	3	0%	90	12%	630	87%
http://logis-de-france.fr/	746	1	50	7%	103	14%	593	79%
http://maurytoday.com/	995	1	33	3%	24	2%	938	94%
http://netook.com/	37	1	15	41%	1	3%	21	57%
http://www.oilcareers.com/	1057	2	0	0%	1043	99%	14	1%
http://www.murauchi.com/	293	2	2	1%	19	6%	272	93%
http://www.fancl.co.jp/	1733	2	549	32%	1174	68%	10	1%
http://gogen-allguide.com/	1418	2	0	0%	0	0%	1418	100%
http://amcham.org.eg/	766	2	1	0%	88	11%	677	88%
http://booklog.jp/	641	3	0	0%	19	3%	622	97%
http://zionsbank.com/	612	3	65	11%	104	17%	443	72%
http://www.ccgp.gov.cn/	924	3	0	0%	923	100%	1	0%
http://jpstu.com/	816	3	1	0%	808	99%	7	1%
http://evermp3.com/	1226	3	0	0%	1026	84%	200	16%
http://ianhardy.net/	713	4	188	26%	6	1%	519	73%
http://2hot4blog.com/	161	4	0	0%	2	1%	159	99%
http://sctv.co.id/	700	4	0	0%	699	100%	1	0%
http://emusician.com/	712	5	2	0%	149	21%	561	79%
http://cronica.com.mx/	886	5	425	48%	457	52%	4	0%
http://www.dublinevents.com/	581	5	0	0%	0	0%	581	100%
http://www.drjays.com/	646	5	0	0%	37	6%	609	94%
http://cupid.it/	26	5	0	0%	24	92%	2	8%
http://catholicpeople.com/	57	5	0	0%	52	91%	5	9%
http://www.soukai.com/	275	6	0	0%	268	97%	7	3%
http://zw-star.com/	838	6	2	0%	835	100%	1	0%
http://plivazdravlje.hr/	205	6	12	6%	44	21%	149	73%
http://www.simplemachines.org/	289	6	8	3%	1	0%	280	97%
http://thewotch.com/	45	7	0	0%	16	36%	29	64%
http://netsh.com/	148	7	0	0%	147	99%	1	1%
http://www.linuxquestions.org/	251	8	107	43%	68	27%	76	30%
http://www.sciaga.pl/	120	10	17	14%	101	84%	2	2%
http://www.kfz-auskunft.de/	193	11	2	1%	190	98%	1	1%
http://www.buzznet.com/	246	12	205	83%	13	5%	28	11%
http://www.record.pt/	676	15	0	0%	673	100%	3	0%
http://www.galveston.com	710	16	0	0%	668	94%	42	6%
Totals	23990		2361	9.8%	9872	41.2%	11757	49.0%

Although the percentage of URLs that set more cookies than the original page is only 9.8%, there were 17 sites that set more than the original number of cookies on a subsequent page. This number suggests that although the initial page is a reasonable indication of the number of cookies used on the majority of subsequent pages on a site, it cannot be considered as a reliable proxy for the maximum number of cookies used by every subsequent page for an entire site. The biggest discrepancy between the number of cookies on the initial page, and the number of cookies used on an entire site, occurs when the initial page does not set any cookies. Of

the 40 sites surveyed, 5 of the sites did not set any cookies on the initial page, and all 5 of those sites set cookies on subsequent pages. This clearly articulates that the 67.4% of sites utilizing cookies must be viewed as a minimum bound on the number of sites that utilize cookies.

3.2.4 Cookie Usage vs. the Identification of Dynamic Web Technologies

Explicit separation of static and dynamic web sites was not feasible given the sheer volume of sites to be surveyed; however, the final redirected URL recorded in the survey can be used to identify sites that use dynamic web technology. Using the collected information, a subset can be extracted containing the sites surveyed that use common web application programming languages, such as PHP, ASP, and JSP. Furthermore, final URLs reflecting external data passed into the website via HTTP GET requests can also be identified by the presence of a "?" in the URL. Although this subset does not contain every web application surveyed, it does provide a set of URLs that are verified to be utilizing dynamic web technologies. Table 3-3 provides a summary of the results obtained when looking at this specific subset of the sample population. Within this subset, there is a much higher rate of cookie utilization, 86.5%, as compared to the general population.

Table 3-3. Cookie Usage amongst Sites Utilizing Dynamic Technologies

Dynamic Identifier	Number of Sites	Sites Utilizing Cookies	Percentage
.asp	2,939	2,725	92.7%
.jsp	943	867	91.9%
?	1,058	908	85.8%
.php	1,917	1,434	78.8%
Total	6,857	5,934	86.5%

Table 3-4. Identifiable Dynamic Sties vs. Cookie Usage Contingency Table

		Cookies		Total
		No	Yes	
Identifiable Dynamic Technology	**No**	31052	60097	91149
	Yes	923	5934	6857
Total		31975	66031	98006

Table 3-5. Chi-Squared Test – Dynamic Technologies vs. Cookie Usage

	Value	Degrees of Freedom	Asymptotic Significance (2-sided)
Pearson Chi-Square	1231.950	1	.000
Continuity Correction	1231.012	1	.000
N of Valid Cases			98006

Table 3-6. Risk Assessment – Dynamic Technologies vs. Cookie Usage

	Value	95% Confidence Interval	
		Upper	Lower
Odds Ratio for Cookies (No / Yes)	3.322	3.095	3.565
For cohort Identifiable Dynamic Technology = No	1.067	1.064	1.070
For cohort Identifiable Dynamic Technology = Yes	.321	.300	.344
N of Valid Cases			98006

To further characterize this difference and determine if a statistically significant difference exists between the two populations, a chi-squared test with Yates correction was applied. This analysis, applied to Table 3-4 and summarized in Table 3-5, identified the presence of a significant difference (χ^2=1231.012 and p<0.05) between sites identified as using dynamic technologies and the others with respect to cookie usage. A further relative risk assessment, summarized in Table 3-6, reveals that there was a 3.12 greater probability of cookie usage amongst sites that used identifiable dynamic web technologies. This dramatic increase in cookie usage is attributed to the role of cookies in maintaining state information within web application frameworks.

3.2.5 Rank vs. Cookie Usage

The selection of the Alexa top 100,000 list provided the ability to evaluate the results of the survey against the reported rankings of the sites. An easily identifiable relationship did not exist between the ranking of the site and the number of cookies used and is further supported by an observed linear correlation coefficient (R^2) of 0.009. The Alexa ranking mechanism has a well-understood bias, as it only collects information from Microsoft Internet Explorer (IE) clients when calculating their traffic statistics. Hence, Microsoft-oriented sites, specifically http://msn.com, the default start-page for IE, are likely to have their ranking inflated. Conversely, popular Internet technology sites, such as Slashdot[5] and Digg[6], or non-Microsoft platform specific sites, such as MacRumors[7] and LinuxHQ[8], experience rating suppression due to the disproportionate frequency of visits from a large numbers of non-IE clients. According to three ongoing surveys, in October 2006, IE accounted for over 50% of all browsing activities (Net Applications, 2006; Nielsen//NetRatings, 2007; The Counter.com, 2006; W3 Schools, 2006), with two of the three surveys suggesting that IE accounts for over 80% of Internet traffic. Although the Alexa rating does contain a bias towards IE clients, this effect is minimized due to the overwhelming market-share of IE.

As an exploratory analysis, the dataset was divided into 100 partitions, and plotted against the percentage of hosts that utilize cookies, as shown in Figure 3-1. Upon visual inspection, a non-linear relationship between ranking and cookie utilization is observed; however, this relationship cannot be considered definitive due to the arbitrary nature of the partition selection. This observed trend suggests that the more popular a site is; the more likely it is to use cookies.

---

[5] http://slashdot.org
[6] http://digg.com
[7] http://www.macrumors.com
[8] http://www.linuxhq.com

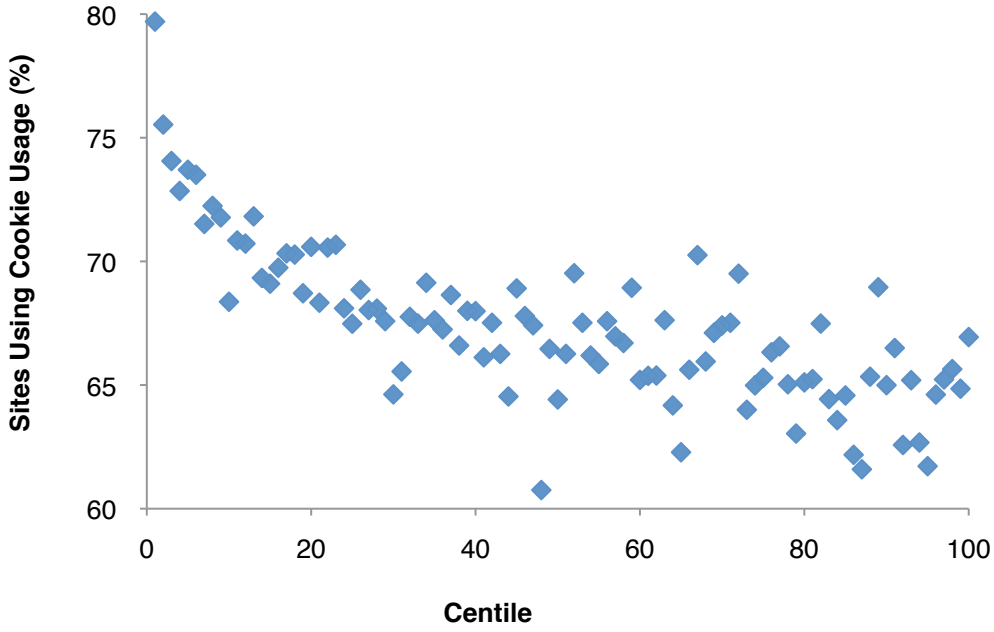


Figure 3-1. Centile Analysis of Cookie Usage (Values Present Only in the Range [60:80])

3.2.6 Number of Cookies per Site

Cookies introduce another layer of complexity to web applications. This subsection will focus upon the complexity added by the use of multiple cookies within a single web application. Figure 3-2 provides a histogram showing the number of cookies per site surveyed. It is observed that a small percentage (0.5%) of sites utilized more than 20 cookies, up to a maximum of 76 cookies. Table 3-7 provides a number of descriptive statistics related to the cookie count frequency analysis. To further understand the usage of multiple cookies, a box plot of the data, shown in Figure 3-3, reveals that the distribution has quartiles of 1, 3, and 6 when zero is excluded from the distribution, higher than that when zeroes were not excluded, 0, 1, and 4. From the box-plots it is clear that the majority of sites utilizing cookies operate within the 1 to 6 cookie range and although there are sites that utilize upwards of 13 cookies (above the 95th percentile), it is not clear why a web application would utilize such large

numbers of cookies, especially when one remembers that the survey only accessed the website homepage. Clearly this phenomenon requires further investigation.

Table 3-7. Descriptive Statistics for Cookie Count Frequency Analysis

N		98006
Mean		2.92
Standard Error of Mean		.012
Median		1.00
Mode		0
Standard Deviation		3.898
Variance		15.191
Range		76
Minimum		0
Maximum		76
Sum		286186
Percentiles	**25**	0
	50	1
	75	4

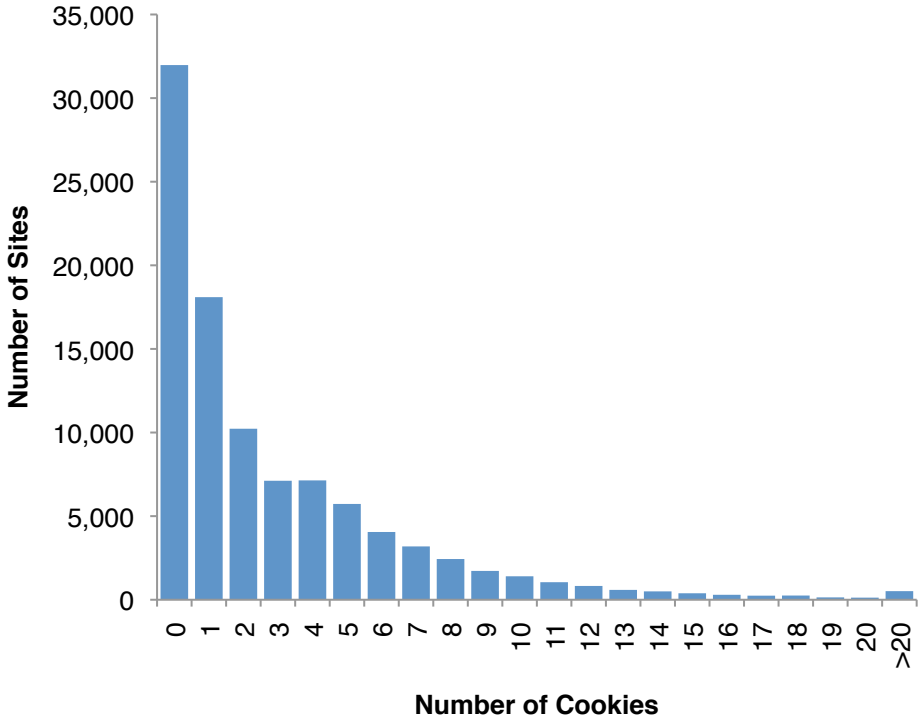
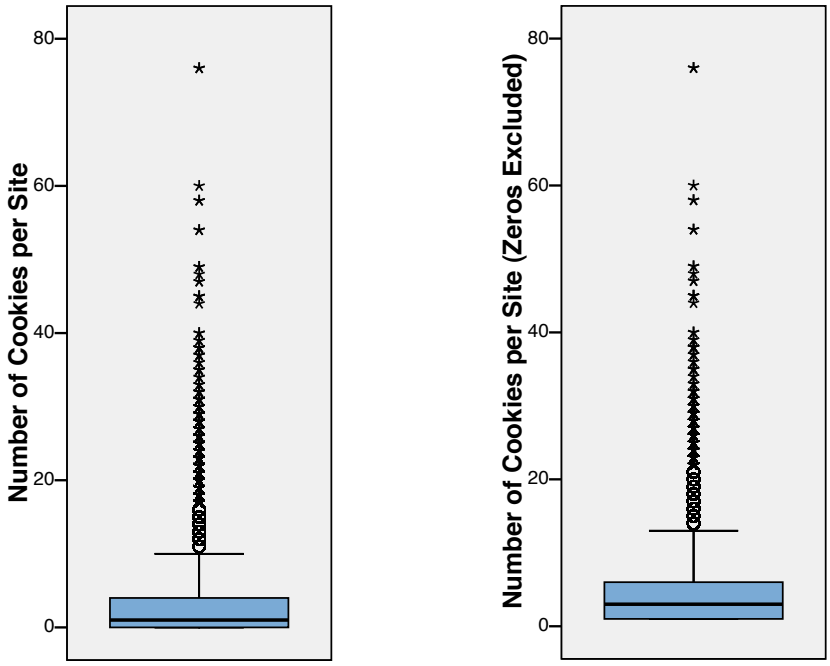


Figure 3-2. Number of Cookies per Site Histogram



(a) **(b)**

Figure 3-3. Box Plots of the Number of Cookies per Site with and without Zeroes

3.2.7 Third-Party Cookies

Cookies, and more specifically third-party cookies, have garnered public attention largely in part from media reports citing privacy concerns and cookie misuse (CBS News, 2002). A third-party cookie, or *unverifiable transaction* (Kristol, 2001; Kristol & Montulli, 1997), occurs when a cookie is set for a domain other than the domain explicitly accessed by the user i.e. a cookie set for the host www.ads.com when viewing the page http://www.example.com. This behavior is very common on the Internet, and is of key importance to the business model of many online advertising agencies. Because of the prevalence of this type of occurrence, cookie specifications mandate that user agents (web browsers) must by default reject third-party cookies (Kristol, 2001; Kristol & Montulli, 1997, 2000). Although this behavior is defined in the cookie specification, it is not adhered to by the two most popular browser clients available today—Internet Explorer (IE) and Firefox, which account for over 90% of all web

traffic, according to several ongoing surveys (Net Applications, 2006; The Counter.com, 2006; W3 Schools, 2006).

The current survey collected 286,186 cookies in total, 56.6% (161,999) of which were set by the originating domain (first-party cookies), and 43.4% (124,187) that were set by external domains (third-party cookies). Table 3-8 provides a summary of the number of sites that use first- and third-party cookies. The key result shown in Table 3-8 is that 54.3% of the sites surveyed utilize third-party cookies—technology that should disabled by default. However, because the most common browsers are released with third-party cookies enabled, disabling these cookies would have significant repercussions on these applications. This issue is further complicated for the 37.8% of applications that utilize a mixture of first- and third-party cookies. It is not known how these applications would react if a subset of their internal state data was missing; this is not a typical test case for many software projects. These complications arise simply from the prospect of strict user-agent adherence to the definition of cookies.

Table 3-8. Third-Party and First-Party Cookie Usage per Site

Cookie Combination	Number of Sites	Percentage
First-Party Only	30,151	45.7%
First-Party & Third-Party	24,979	37.8%
Third-Party Only	10,901	16.5%

Of the websites surveyed, those with a minimum of 27 cookies, with the exception of five cases, were found to have at least one first-party cookie. Similarly, websites with a minimum of 27 cookies, with the exception of five cases, were found to have at least one third-party cookie. Figure 3-4 suggests that a non-linear relationship exists between the ratio of the median of the first-party to the median of the third-party cookies. This relationship suggests that the ratio of first- to third-party cookies increases inversely-proportional to the total number of cookies used on a site. That is, as the number of cookies a site uses increases, the number of

third-party cookies increases, while the number of first-party cookies remains relatively low. This phenomenon is clearly observed in the two most extreme cases in which each site uses 76 cookies. In the case of these two sites, the ratios of first- to third-party cookies were 2 to 74 and 3 to 73. In general, the median ratio of first- to third-party cookies remains above 1.0 for sites containing 12 of fewer cookies, and remains below 1.0 for sites containing 13 or higher cookies. For sites that utilize 6 or fewer cookies, the media ratio of first- to third-party cookies was greater than or equal to 2.0. That is, on average these sites were observed to utilized 2 or more times the number of first- than third-party cookies. Conversely, on average, sites that utilized more than 24 cookies were observed to utilize more that twice the number of third- than first-party cookies.

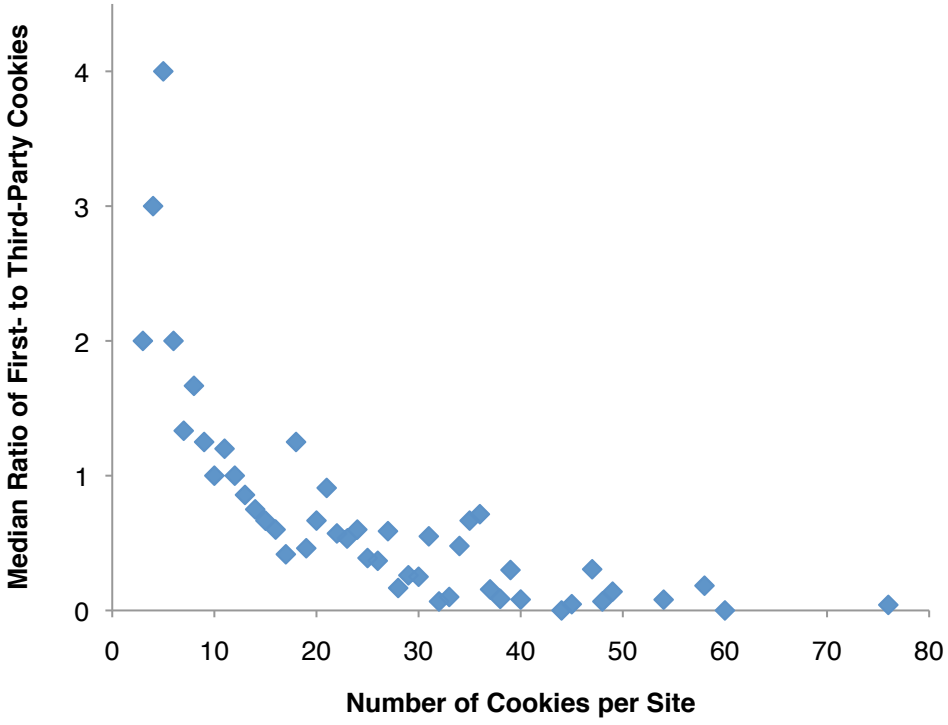


Figure 3-4. Number of Cookies per Site vs. the Median Ratio of First- to Third-Party Cookies

3.2.8 Cookie Lifespan

As defined by RFC 2109, 2965 (Kristol & Montulli, 1997, 2000), cookies can contain a MAX-AGE value; this value defines the lifespan of the cookie, and determines if the cookie is a sessional or a persistent cookie. Sessional cookies only last for the client session and are deleted when the browser closes. Persistent cookies, on the other hand, remain on the client machine until the defined expiry date (MAX-AGE value), deleted by the system based upon system resource utilization, or explicitly deleted by a user. Of the 286,186 cookies surveyed, 39.3% (112,398) were sessional and 60.7% (173,788) were persistent. As with first- and third-party cookies, sessional and persistent cookies often both exist within the same web application. Table 3-9 provides a summary of the number of sites that use sessional and persistent cookies. Websites with 23 or more cookies were observed to have at least one persistent cookie. In fact, with the exception of two cases, every website with a minimum of 15 cookies had at least one persistent cookie. Similarly, with the exception of three cases, all websites with a minimum of 18 cookies possessed at least one sessional cookie.

Table 3-9. Sessional and Persistent Cookie Usage per Site

Cookie Combination	Number of Sites	Percentage
Sessional Only	18,027	27.3%
Sessional and Persistent	35,538	53.8%
Persistent Only	12,466	18.9%

The usage trends of sessional and persistent cookies within the same site were observed to be quite different than that observed for first- and third-party cookies. Figure 3-5 suggests that a very different relationship exists between the ratio of the median of sessional to the median of persistent cookies, than that observed for first- to third-party cookies. Although an easily identifiable relationship was not present, a very interesting trend was observed. With the exception of four cases, the ratio of sessional to persistent cookies was bounded in the range (0.2, 1.0].

All of the exceptions occurred in sites that utilized more that 38 cookies, and represented less than 0.1% of the sites surveyed. This finding suggests that the majority of sites, that use more than one cookie, use a combination of sessional and persistent cookies; in fact, 74.1% of sites using multiple cookies utilized at least one sessional and persistent cookie. The observed bound in the ratio of medians further suggests that the majority of sites use more persistent than sessional cookies. This seemingly static bound is in stark contrast to the occurrence observed in the ratio of the medians of first- to third party cookies.

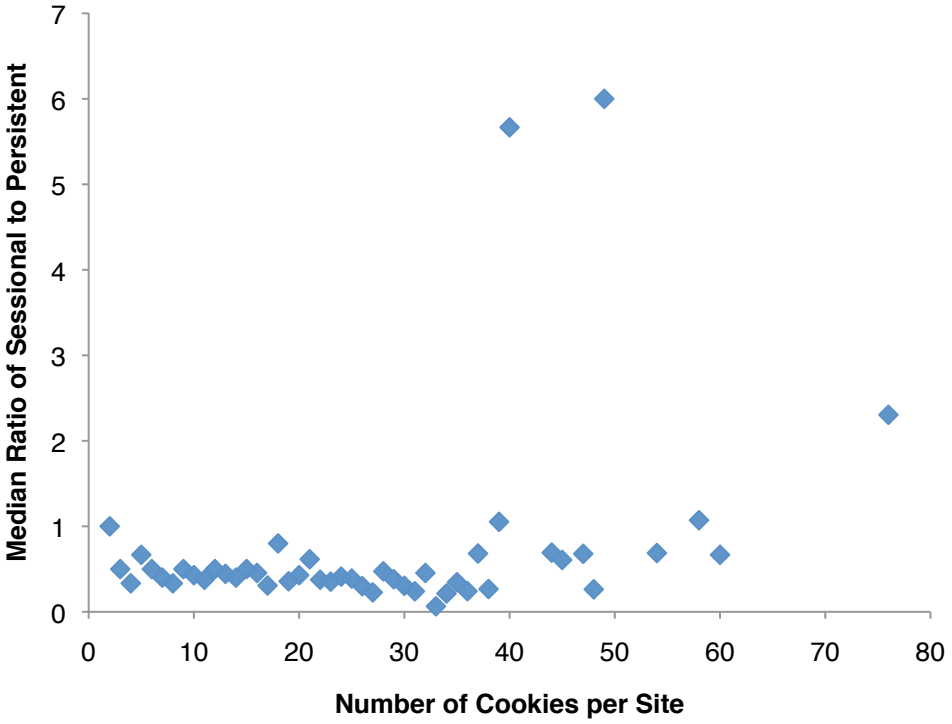


Figure 3-5. Number of Cookies per Site vs. the Median Ratio of Sessional to Persistent Cookies

Much like the ability to disable third party cookies, common browser clients also provide the ability to filter cookie requests based on the time-to-live duration. Browsers can be configured to reject persistent cookies and only accept sessional cookies; and conversely, to treat every cookie as a sessional cookie, deleting persistent cookies each time the browser is terminated. These configuration options, as mentioned in the

previous section, can have unforeseen effects on a web application, considering that different browser configurations can affect a subset of an application's internal state information. As shown in Table 3-9, 72.7% of web applications using cookies use at least one persistent cookie. These persistent cookies are the most likely to be rejected or deleted prematurely due to the various built-in browser configurations. Furthermore, it was observed that 74.1% of sites using multiple cookies utilize both persistent and sessional cookies in conjunction; it is unknown what affect the possible rejection of a subset of the system state information has on an application.

Table 3-10. Persistent Cookie Lifespan

Cookie Lifespan	Number of Cookies	Percentage
< 1 day	31,490	18.1 %
< 1 week	17,529	10.1 %
< 1 month (30 days)	12,114	7.0 %
< 1 year	25,169	14.5 %
< 1 decade	52,598	30.3 %
≥ 1 decade	34,888	20.1 %

Table 3-10 provides a summary of the various ranges in lifespan for persistent cookies. From Table 3-10 it is observed that approximately half of all persistent cookies surveyed are set to live for over one year (50.4%), and one fifth of persistent cookies are set to live for over a decade. Although not shown in the Table 3-10, the survey unearthed a small number of cookies (131) that were set to live for longer that one century.

A plot of the number of valid cookies vs. year, shown in Figure 3-6, exposes the existence of several key dates before which a large number of cookies are set to expire. The most dramatic increase in the number of cookies expiring occurs in 2007, 2008, 2012, 2017, and 2039. The initial two increases in expired cookies occur during the first two years, and appear to be consistent with an inverse relationship between the number of non-expired cookies and time. Two smaller increases occur in 2012 and 2017, suggesting that expiry dates within the range of 6 or 11 years are

more prominent than others in the range of 3 – 30 years. The final major increase in the number of expired cookies occurs in 2039. This increase of 9,205 was the second largest single year increase and was surpassed only by the increase during the first year. This is a very interesting finding as this represents a period of 32 years, not a number of any significance when compared with its immediate predecessor or successor. This dramatic increase is most likely the due to the known year 2038 problem that exists within many 32-bit operating systems that rely upon the POSIX time representation for the system clock. This problem will culminate at 03:14:07 UTC on Tuesday, January 19, 2038, at which point the system clocks will be reset to the year 1901. It appears that the majority of web-applications set cookies with expiry dates before this date, presumably to avoid having the cookies deleted, because the system could interpret the dates as being in the past.

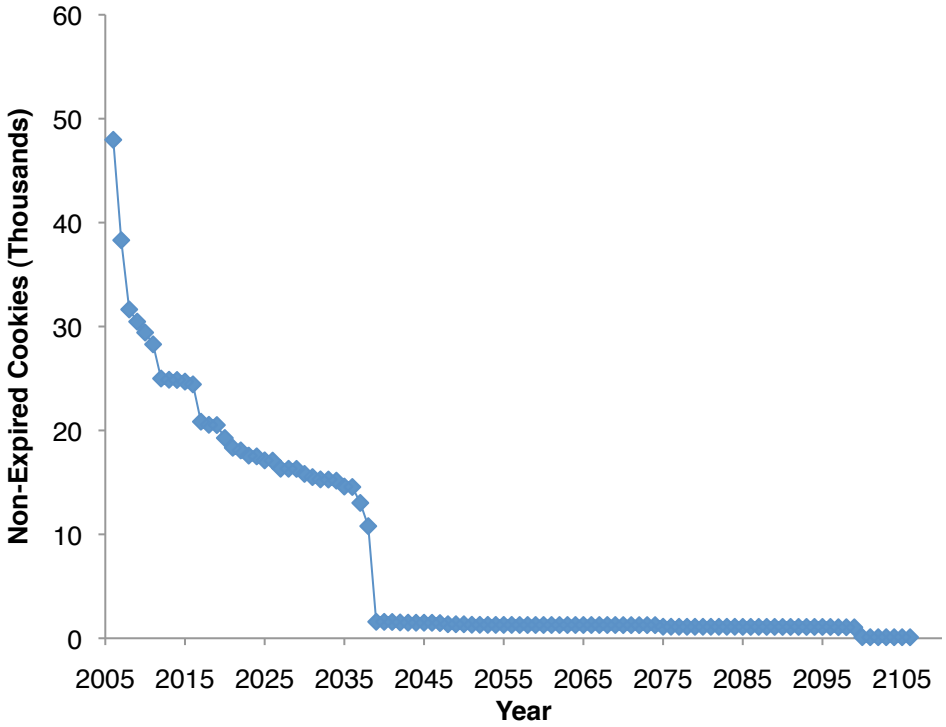


Figure 3-6. The Number of Non-Expired Cookies vs. Year

Table 3-11. Cookie Lifespan: 1 Persistent Cookie vs. 53 Persistent Cookies

Cookie Lifespan	1 Persistent Cookie		53 Persistent Cookies	
< 1 day	797	15.0 %	1	1.9 %
< 1 week	496	9.3 %	0	0 %
< 1 month (30 days)	522	9.8 %	0	0 %
< 1 year	419	7.9 %	1	1.9 %
< 1 decade	2,145	40.3 %	50	94.3 %
≥ 1 decade	946	17.8 %	1	1.9 %

Table 3-12. Cookie Lifespan vs. Number of Cookies per Site Contingency Table

		Number of Cookies per Site		Total
		> 1 Cookie	**1 Cookie**	
Cookie Lifespan	**< 1 year**	84068	2234	86302
	≥ 1 year	84395	3091	87486
Total		168463	5325	173788

Table 3-11 highlights the individual cookie lifespan for sites that only set one cookie and the single site that uses the most persistent cookies—53. The differences between the columns in Table 3-10 and Table 3-11 suggest that a significant difference exists in the time-to-live duration of a cookie set by a site that only utilizes one persistent cookie. To explore this difference, a contingency table was created and a chi-squared test with Yates correction was applied. This analysis, Table 3-12 and summarized in Table 3-13 identified the presence of a significant difference (χ^2=130.181 and p<0.05) in cookie lifespan between persistent cookies originating from sites that use only one cookie and those using multiple cookies. A further relative risk assessment, summarized in Table 3-14, revealed that there is a 1.3 greater probability of a lifespan over one year for those persistent cookies originating from sites with only one cookie than those originating from sites using multiple cookies.

Table 3-13. Chi-Squared Tests for Table 3-12

	Value	Degrees of Freedom	Asymptotic Significance (2-sided)
Pearson Chi-Square	130.499	1	.000
Continuity Correction	130.181	1	.000
N of Valid Cases			173788

Table 3-14. Risk Estimation for Table 3-13

	Value	95% Confidence Interval	
		Upper	Lower
Odds Ratio for Lifespan (< 1year / ≥ 1year)	1.378	1.304	1.457
For cohort: More Than One Cookie	1.010	1.008	1.011
For cohort: One Cookie	.733	.694	.773
N of Valid Cases			173788

Several issues arise from the use of sessional and persistent cookies. Cookies are often the explicit target for very common cross-site scripting exploits (Cgisecurity.com, 2002; Cook, 2003; Fogie, 2006). What security implications does the unauthorized access to a user's cookie(s) have on a web application? With a majority of cookies on the Internet being persistent, how does cookie theft affect the testing strategies used for a web application? Issues arising due to the theft of persistent cookies are not the only security risks facing web applications; sessional cookies, although specified to be deleted at the end of a browsing session, can be modified. Just like persistent cookies, these cookies can also be stolen and or modified to live for an undetermined period. How will a web application respond if the assumption that sessional cookies are not permanent is false? These questions are not only valid, but are necessary in ensuring that web applications can provide a secure and reliable operating environment to users over the Internet.

Table 3-15. Sessional/Persistent vs. First-/Third-Party Cookies

	Sessional		Persistent	
First-Party	77,499	27.1%	84,500	29.5%
Third-Party	34,899	12.2%	89,288	31.2%

Table 3-16. Chi-Squared Tests For First-/Third-Party vs. Sessional/Persistent

	Value	Degrees of Freedom	Asymptotic Significance (2-sided)
Pearson Chi-Square	11482.328	1	.000
Continuity Correction	11481.501	1	.000
N of Valid Cases			286186

Table 3-17. Risk Estimation for First-/Third-Party vs. Sessional/Persistent

	Value	95% Confidence Interval	
		Upper	Lower
Odds Ratio for Duration (Persistent / Sessional)	.426	.420	.433
For cohort First-Party	.705	.701	.710
For cohort Third-Party	1.655	1.638	1.671
N of Valid Cases			286186

To gain a better understanding of how cookies are utilized in modern web applications, an integrated analysis of cookie lifespan and usage of third-party cookies is required. It can be seen from Table 3-15 that the smallest contingent of cookies, 12.2%, are third-party sessional cookies. More specifically, 26.1% of third-party cookies are sessional, whereas the majority of third-party cookies, 73.9% are persistent. To further characterize these findings, a chi-squared test with Yates correction applied to the data in Table 3-15 and summarized Table 3-16, identified a significant difference (χ^2=130.181 and p<0.05) between the number of first- and third-party cookies that were persistent. A further risk assessment, summarized in Table 3-17, reveals that third-party cookies have a 1.66 greater probability of being persistent. This increase in the probability amongst third-party cookies is attributed to the usage of

third-party cookies within online advertising and tracking applications, an issue that will be discussed at length in the next section.

3.2.9 Online Tracking & Web Bugs

As mentioned in Section 3.2.7, third-party cookies have gained media attention due to privacy issues surrounding the tracking of online users. A mechanism for this type of tracking conducted without the user's knowledge has been labeled a *web bug* (Smith, 1999). Web bugs, given their name due to the covert nature of the technology, are usually planted within an HTML page as a small invisible image, or as part of an advertisement. Cookies used for these purposes are third-party and are often persistent, to facilitate the long-term tracking of user browsing habits. To identify the cookies used as web bugs, the survey dataset was analyzed with respect to third-party persistent cookies. This category of cookie was found to be predominantly set by a small number of Internet hosts, mainly comprising online advertisers.

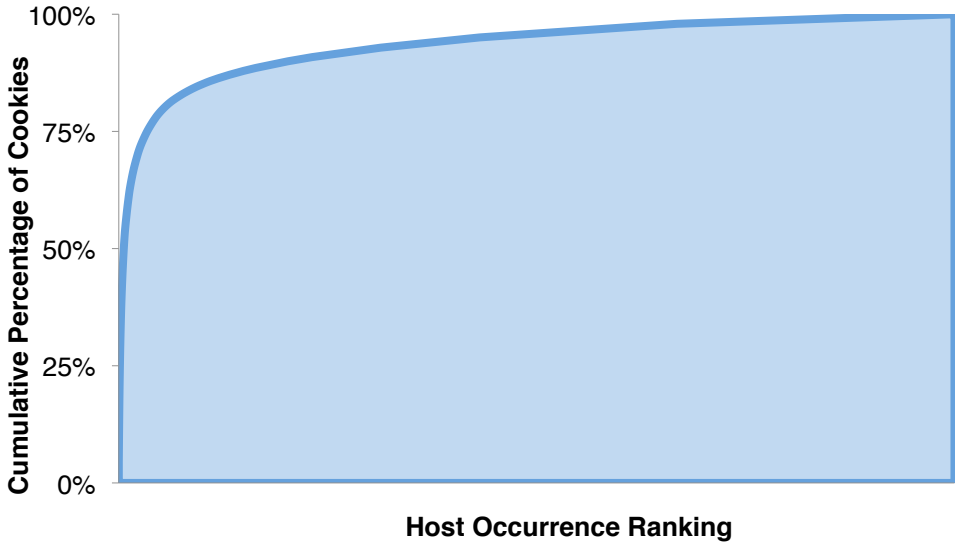


Figure 3-7. Host Occurrence vs. Cumulative Percentage of Third-Party Persistent Cookies

To analyze this portion of the data, each cookie was examined on a per host basis. For the purpose of the analysis, it was assumed that two hosts were equal if the second- and top-level domains were equal. A list of hosts ranked in descending order, based upon number of third party persistent cookies set by the host, is graphed against the cumulative percentage of cookies set in Figure 3-7. It is clear from the shape of the curve in Figure 3-7 that the majority of cookies come from a small minority of sites. From Figure 3-7 two interesting ratios can be selected: an 80/5 and 90/20 ratio. These ratios highlight the overwhelming use of this category of cookies by a small subset of the hosts. Upon further inspection, the 5% and 20% account for a very small percentage of the overall number of hosts setting cookies, 0.4% and 1.4% respectively. As assumed, persistent third-party cookies are used predominantly by a small number of Internet hosts. This finding further confirms that the majority of third-party persistent cookies surveyed are used as part of an online identification and advertising endeavors.

3.2.10 P3P Policy Adoption and Cookie Usage

Internet Explorer (IE) poses a unique set of challenges to web applications in that, by default, it is set to reject any third-party cookies set by a third-party host that does not contain a compact Platform for Privacy Preferences (P3P) policy. P3P is a standard that exists to allow the automatic retrieval and interpretation of a privacy policy by a user agent (W3C, 2006). IE also provides more stringent security settings, and on the predefined security zone setting of *high*, IE will reject all first-party and third party cookies from hosts without a compact P3P policy. Due to the large market share of IE amongst browsers currently used on the Internet (Net Applications, 2006; The Counter.com, 2006; W3 Schools, 2006), P3P adoption is mandatory for any host attempting to set a third-party cookie and for those who wish to utilize cookies regardless of the security setting a user's browser. To study the relationship between P3P

policy adoption and cookie usage the survey population was partition based upon the results of a recent survey conducted by Reay, Beatty, Dick and Miller (2007). The survey conducted by Reay et al. surveyed 95,824 sites from the same sample population; these sites cross-referenced with the current survey resulted in a new dataset of 94,552 sites with both P3P policy and cookie usage information. This new population was then partitioned upon the presence of a valid full and compact P3P policy. These two partitions were evaluated against cookie usage from several perspectives, as shown in Table 3-18. To further characterize the differences in populations, a number of chi-squared tests were undertaken, each yielding a significant difference for each of the partitions presented in Table 3-18. These results will be presented and discussed in the remainder of this section.

Table 3-18. Cookie Usage vs. P3P Policy Adoption

	Cookies		1st Party Cookies		3rd Party Cookies		Sessional Cookies		Persistent Cookies	
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
With Full P3P	1,911	375	1,768	518	1,161	1,125	1,549	737	1,611	675
Without Full P3P	62,020	30,246	51,621	40,645	33,678	58,588	50,320	41,946	45,057	47,209
With Compact P3P	1,942	259	1,759	442	1,117	1,084	1,517	684	1,647	554
Without Compact P3P	61,989	30,362	51,630	40,721	33,722	58,629	50,352	41,999	45,021	47,330

The first partition of the population examined was P3P policy adoption vs. Cookie usage. A chi-squared test with Yates correction, summarized in Table 3-19, reveals that significant differences (χ^2=272.485 and p<0.05, χ^2=436.510 and p<0.05) exist between those sites with valid P3P policies (full or compact) and those without. A further relative risk assessment, summarized in Table 3-20 reveals that there was a 2.44 greater probability of cookie usage amongst sites with a valid Full P3P policy, and a 3.60 greater probability amongst those with a compact P3P policy. This is a significant increase in probability of a site using cookies,

and is the single largest predictor of cookie usage identified by this survey. The increased probability of cookies usage is attributed to the close association of a compact P3P policy and cookies within the most widely used Internet browser—IE.

Table 3-19. Chi-Squared Test Results for Cookie Usage vs. P3P Adoption

		Value	Degrees of Freedom	Asymptotic Significance (2-sided)
Sites with a valid Full P3P Policy	**Pearson Chi-Square**	273.232	1	.000
	Continuity Correction	272.485	1	.000
Sites with a valid Compact P3P Policy	**Pearson Chi-Square**	437.474	1	.000
	Continuity Correction	436.510	1	.000
N of Valid Cases				94,552

Table 3-20. Risk assessment for cookie usage vs. P3P adoption

		Value	95% Confidence Interval	
			Upper	Lower
Sites with a valid Full P3P Policy	**P3P Policy? = No**	1.018	1.016	1.020
	P3P Policy? = Yes	.410	.367	.457
Sites with a valid Compact P3P Policy	**P3P Policy? = No**	1.023	1.021	1.024
	P3P Policy? = Yes	.278	.245	.317
N of Valid Cases				94,552

To further understand the nature of the relationship between P3P adoption and cookie usage, chi-squared tests with Yates correction were applied with respect to first-party, third-party, sessional, and persistent cookies and full and compact P3P usage. The results, summarized in Table 3-21 and Table 3-22, reveal a significant difference for each of the partitions. To further characterize and compare these differences, risk assessments were complied for each partition and are summarized in Table 3-23 and Table 3-24. The tests reveal that the presence of a valid full P3P policy is a predictor for the usage of all four sub-categories of cookies, especially first-party and persistent cookies which were found to have a 2.63 and 2.45 greater probabilities, respectively, than the sites without a valid full P3P policy. This trend was also observed amongst sites

with a valid compact P3P policy; however, a compact P3P policy was an even greater predictor of first-party and persistent cookies found to have a 3.07 and 3.05 greater probabilities respectively. This increase of first-party and persistent cookies on sites with a compact P3P policy is again attributed to the usage of a compact P3P policy within IE.

Table 3-21. Specific Cookie Usage vs. Sites with a Full P3P Policy

		Value	Degrees of Freedom	Asymptotic Significance (2-sided)
1st-Party Cookies vs. Full P3P Policy	**Pearson Chi-Square**	415.284	1	.000
	Continuity Correction	414.414	1	.000
3rd-Party Cookies vs. Full P3P Policy	**Pearson Chi-Square**	195.659	1	.000
	Continuity Correction	195.046	1	.000
Sessional Cookies vs. Full P3P Policy	**Pearson Chi-Square**	157.486	1	.000
	Continuity Correction	156.952	1	.000
Persistent Cookies vs. Full P3P Policy	**Pearson Chi-Square**	417.868	1	.000
	Continuity Correction	417.002	1	.000
N of Valid Cases				94,552

Table 3-22. Specific Cookie Usage vs. Sites with a Compact P3P Policy

		Value	Degrees of Freedom	Asymptotic Significance (2-sided)
1st-Party Cookies vs. Compact P3P Policy	**Pearson Chi-Square**	504.230	1	.000
	Continuity Correction	503.253	1	.000
3rd-Party Cookies vs. Compact P3P Policy	**Pearson Chi-Square**	187.193	1	.000
	Continuity Correction	186.582	1	.000
Sessional Cookies vs. Compact P3P Policy	**Pearson Chi-Square**	180.029	1	.000
	Continuity Correction	179.448	1	.000
Persistent Cookies vs. Compact P3P Policy	**Pearson Chi-Square**	584.964	1	.000
	Continuity Correction	583.921	1	.000
N of Valid Cases				94,552

The increase is much higher than those associated with third-party and sessional cookies, 1.77 and 1.89 amongst sites with full P3P polices, and 1.76 and 1.82 amongst those with compact P3P policies. In-fact the probabilities associated with third-party and sessional cookies were within the 95% confidence interval for both the full and compact P3P policy, and although full P3P policy seems to possess a higher predictor indicator for

third-party and sessional cookies, the difference cannot be considered significant, and it appears that the type of P3P policy does not have any differentiating affect on the probabilities of existence for these types of cookies. The observed increase is not fully explained; however, it is believed that due to the significant correlation between P3P policy and third-party cookies, sites adopting P3P policies are principally concerned with cookies, suggesting that these sites are more likely to use all types of cookies, not just first-party or persistent cookies.

Table 3-23. Risk Assessment for Cookie Usage vs. Site with a Full P3P Policy

		Value	95% Confidence Interval	
			Upper	Lower
First-Party Cookies vs. Full P3P Policy	**P3P Policy? = No**	1.021	1.019	1.023
	P3P Policy? = Yes	.380	.345	.419
Third-Party Cookies vs. Full P3P Policy	**P3P Policy? = No**	1.015	1.013	1.017
	P3P Policy? = Yes	.565	.521	.613
Sessional Cookies vs. Full P3P Policy	**P3P Policy? = No**	1.013	1.011	1.015
	P3P Policy? = Yes	.578	.530	.631
Persistent Cookies vs. Full P3P Policy	**P3P Policy? = No**	1.021	1.019	1.023
	P3P Policy? = Yes	.408	.374	.446
N of Valid Cases				94,552

Table 3-24. Risk Assessment for Cookie Usage vs. Site with a Compact P3P Policy

		Value	95% Confidence Interval	
			Upper	Lower
First-Party Cookies vs. Compact P3P Policy	**P3P Policy? = No**	1.021	1.019	1.023
	P3P Policy? = Yes	.380	.345	.419
Third-Party Cookies vs. Compact P3P Policy	**P3P Policy? = No**	1.015	1.013	1.017
	P3P Policy? = Yes	.565	.521	.613
Sessional Cookies vs. Compact P3P Policy	**P3P Policy? = No**	1.013	1.011	1.015
	P3P Policy? = Yes	.578	.530	.631
Persistent Cookies vs. Compact P3P Policy	**P3P Policy? = No**	1.021	1.019	1.023
	P3P Policy? = Yes	.408	.374	.446
N of Valid Cases				94,552

The fact that IE is the primary browser used on the Internet and that a P3P policy is required by the default IE settings, implies that IE

compliance may be the main driver behind P3P policy adoption, specifically with respect to third-party cookies. Any host with a P3P policy is therefore likely to set third-party cookies; however, on a host's original site, as accessed by this survey, all cookies from the host will be considered first-party. Due to the primary role of a P3P policy within IE the presence of a valid compact P3P policy suggests that these hosts have an increased interest in setting cookies; hence, the dramatic increase in first-party cookie usage amongst this partition. The complementary increase observed with respect to persistent cookies within these partitions can also be attributed to IE and P3P compliance. As discussed in Sections 3.2.8 and 3.2.9, the primary roles of third-party cookies is for online advertising and information gathering, requiring cookies to be both third-party and persistent, in order to provide user-specific advertisements based upon the user's online browsing tendencies. This association between persistent and third-party cookies, and the resulting association between persistent cookies and P3P policies are attributed to the increased usage of both persistent and first-party cookies amongst sites which have a P3P policy.

3.3 Real World Cookie Deployment

3.3.1 Case Study: Cookie Deployment Within a Single Site

To gain a better understanding of cookie deployment within a single site, one URL from the survey was selected for further investigation. This analysis focused on the online storefront located at *http://fossil.com,* which was selected for three reasons:

- The application was representative of typical e-commerce applications using the ubiquitous shopping cart metaphor.
- The application set a large number of cookies upon the initial visit and was within the 99th percentile of sites that set the highest number of cookies.

- The site set a significant number of both first- and third-party cookies upon an initial visit, providing the opportunity to investigate the usage of first- and third-party cookies.

This section will provide a brief analysis of the cookies used within the subject site, with comparisons to the results presented in the previous sections.

3.3.1.1 Third-Party Cookies: Web Bugs & Embedded Third-Party JavaScript

The application located at *http://fossil.com* was found to use twenty cookies, nine of which were third-party. The nine third-party cookies were set by seven different hosts and are summarized in Table 3-25. Each of the third-party cookies were set by hosts responsible for over 80% of all third-party persistent cookies as identified in Section 3.2.9. The third-party cookies were all set by means indiscernible to an average Internet user, either as an invisible web-bug or through the use of third-party JavaScript scripts embedded within the application. The site contained four web-bugs, each a blank image with miniscule dimensions (0x0 or 1x1 pixels) that linked to a third-party advertising agent. Each bug contained a series of variables that identified the originating site, and a number of other values that were not identifiable. These four web-bugs were responsible for six of the nine third-party cookies, while embedded third-party JavaScript scripts were responsible for the remaining three third-party cookies.

This practice of setting third-party cookies via embedded JavaScript scripts or web-bugs exemplifies the practice identified and censured as an *unverifiable transaction* in the original cookie specification (Kristol, 2001; Kristol & Montulli, 1997, 2000). Although this type of cookie usage is not obvious to the average end-user, it is important to note that these third-party agencies provide the application with elicited functionality. The means of accomplishing this functionality, embedded third-party scripts

and web-bugs, are implemented at the discretion of the original host. Further inspection of the page source reveals that the addition of these third-party audiences is well documented (with vendor-specific comments), along with the date on which they were added (see Figure 3-8). These third-party code segments represent COTS components that are included to fulfill a requirement of the system, albeit a requirement that is directly related to the business model of the application. An examination of the site's privacy policy acknowledges the use of cookies and web beacons (web-bugs). However, the policy does not explicitly mention the use of third-party cookies, nor the fact that information is passed between the site and third parties using cookies.

Table 3-25. Third-Party Cookies for *http://fossil.com*

Host	Path	Name	Value	Time to Live
.yahoo.com	/	B	dcmomfp30lh7h&b=3&s=64	30 years 2 months 4 days 18 minutes 11 seconds
.google.com	/	PREF	ID=c46876a2e3c6f7fa:TM=1175110897:LM=1175110897:S=c8BRBXDOWD6LOCyA	30 years 9 months 18 days 22 hours 32 minutes 18 seconds
.atdmt.com	/	AA002	001175110911-992108986/1176320511	4 years 11 months 27 days 4 hours 18 minutes 12 seconds
.advertising.com	/	C2	/TsCGZYcI04jFAH	42 years 2 months 24 days 18 hours 41 minutes 54 seconds
.doubleclick.net	/	id	800000cbaef6d9d	2 years 11 months 29 days 3 seconds
.doubleclick.net	/	test_cookie	CheckForPermission	15 minutes 3 seconds
.imiclk.com	/	IMI	15036298851	11 months 29 days 3 seconds
.imiclk.com	/	SG	kZu96CvC%3Apg%3DT2tg%26apg%3DT2tj%26st%3DA2uj%7Cabsplt%3Aab%3DA2b%5E%5E1	11 months 29 days 3 seconds
.perf.overture.com	/	SYSTEM_USER_ID	9KEDE4CQ261MM70R1864HF5RTS	3 years 11 months 29 days 2 seconds

All of the nine third-party cookies listed in Table 3-25 were persistent, and six of the cookies were set to live for more than one year. Two of the three exceptions were set to expire after 364 days; the third expired after only 15 minutes. The short-lived cookie by virtue of its name (*test_cookie*) functions to ensure that third-party cookies can be set, and

was accompanied by a second cookie from the same domain that was set to expire after three years less a day. Four of the nine cookies were set to live for more than 30 years, and the longest-living cookie was set to expire after 42 years, within the top 99th percentile of expiration dates encountered. In fact, this cookie was set to expire beyond the 2038 limit exposed in Section 3.2.8. All of the expiry dates were found to be relative to their creation date; i.e. they represented an offset from their creation dates.

```
598: …
599:
600: <!-- BEGIN ADVERTISING.COM TRACKING CODE (added 4/6/2006) -->
601: <img src="http://leadback.advertising.com/adcedge/lb?site=69550
     1&srvc=1&betr=fossil_lb_cs=1&betq=1649=364429" width="1" height="1"
     border="0">
602: <!-- END ADVERTISING.COM TRACKING CODE (added 4/6/2006) -->
603:
604: <!-- BEGIN: NEXTACTION ATLAS ACTION TAG (added 5/2/2006) -->
605:   <img height="1" width="1" src="http://switch.atdmt.com/actio
     n/apmfos_homepage_1/v3/ato.CUSTOMERID/"/>
606: <!-- END: NEXTACTION ATLAS ACTION TAG (added 5/2/2006)  -->
607:
608: <!-- BEGIN: I-MEDIA HOMEPAGE TRACKING TAG (added 5/2/2006) -->
609:   <IFRAME SRC="http://www.imiclk.com/cgi/r.cgi?m=3&mid=kZu96Cv
     C&ptid=HOME" FRAMEBORDER="0" SCROLLING="NO" WIDTH="0" HEIGHT="0">
     </IFRAME>
610: <!-- END: I-MEDIA HOMEPAGE TRACKING TAG (added 5/2/200
611: …
```

Lines 598 – 611 of http://www.fossil.com/jump.jsp

Figure 3-8. Documented Third-Party Cookie Inclusion From http://fossil.com

3.3.1.2 First-Party Cookies & the JSP Session Token

The eleven first-party cookies summarized in Table 3-26 were set by two hosts, *fossil.com* and *www.fossil.com*. Although both hosts share the same domain name and resolve to the same IP address, from a cookies perspective the two hostnames are considered distinct. Any cookie set for the domain *fossil.com* is considered first-party for the domain *www.fossil.com*, but the reverse does not hold true (Kristol & Montulli, 1997, 2000). Although originating from the same host, the definition of these cookies as first- or third-party is dependant on the initial request. This raises a number of questions in view of current developments within the academic community, namely CookiePicker, a browser extension being

developed by Yue et al. (2007). CookiePicker provides automatic cookie filtration, rejecting all third-party cookies and selectively rejecting any non-output altering first-party persistent cookies. This technology would, by default, reject half of the first-party cookies if the initial request were directed to *http://fossil.com*. This behavior is clearly unwarranted, and it emphasizes the need for careful consideration and testing when selecting all values associated with a cookie—name, value, host, path, secure, etc. This problem is further compounded by the default rejection of third-party cookies as outlined by the cookie specifications. Although this behavior has not been widely adopted by the most popular browsers, these cookies still remain at heightened risk of rejection. However, for the purposes of this discussion all of the cookies originating from either domain will be considered first-party, because the initial request to *http://fossil.com* was redirected to *http://www.fossil.com*, and all subsequent pages were served from the domain *www.fossil.com*.

Six of the eleven first-party cookies were set in the HTTP header response to the original request for the URL (see Figure 3-9). The initial request responded with the status *302 FOUND*, indicating that the page exists at a different URL; therefore a second request was made to the URL specified by the value in the *Location* field of the response. The request initially seeking *http://fossil.com* was redirected to the page *http://www.fossil.com/jump.jsp* on the host *www.fossil.com*. The second request, responded to by the host *www.fossil.com*, is responsible for setting the majority of the first-party cookies, and in fact sets a second *JSESSIONID* cookie, with a distinct host and value. As mentioned previously, all six of these cookies are considered to be first-party. These six cookies, *JSESSIONID* (*fossil.com*), *JSESSIONID* (*www.fossil.com*), *lastJVM_ID*, *mmlID*, *customer*, and *order* are set in the method outlined by the original cookie specification (Kristol & Montulli, 1997, 2000), and work together to provide the basic functionality of the application's shopping-cart checkout system.

Table 3-26. First-Party Cookies for *http://fossil.com*

Host	Path	Name	Value	Time to Live
fossil.com	/	JSESSIONID	rXUl_CoHMSu5	Sessional
www.fossil.com	/	ysm_CKAJTAOCVD4V5RMM2B78CVG78KJ4	ysm_PVAJTAOCVD4V5RMM2B78CVG78KJ4:1&ysm_SNAJTAOCVD4V5RMM2B78CVG78KJ4:1175110910209&ysm_LDAJTAOCVD4V5RMM2B78CVG78KJ4:0	Sessional
www.fossil.com	/	JSESSIONID	ska9q2OXv7Oh	Sessional
www.fossil.com	/	lastJVM_ID	Secondary	Sessional
www.fossil.com	/	mmlID	118799436	7 years 11 months 28 days 0 seconds
www.fossil.com	/	customer	135741968	7 years 11 months 28 days 0 seconds
www.fossil.com	/	order	105172296	29 days 0 seconds
.fossil.com	/	__utmz	193564346.1175110911.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none)	0 years 5 months 28 days 12 hours 2 seconds
.fossil.com	/	__utmc	193564346	Sessional
.fossil.com	/	__utmb	193564346	30 minutes 2 seconds
.fossil.com	/	__utma	193564346.1635159854.1175110911.1175110911.1175110911.1	30 years 9 months 19 days 3 hours 18 minutes 12 seconds

Two of the cookies, *JSESSIONID* (from the host *www.fossil.com*) and *order*, work together to maintain the state of the virtual shopping cart. The system can sustain basic shopping-cart functionality with one of these two cookies present; however, if both the *JSESSIONID* and *order* cookies are deleted, the shopping cart state is lost. The *JSESSIONID* cookie appears to do more than just maintain the shopping cart, when this cookie is deleted the application immediately (in the next HTTP header response) reinitializes the cookie and any of the other four cookies. Apart from the *JSESSIONID* cookie, only the *lastJVM_ID* cookie reinitializes itself. The *JSESSIONID* and *lastJVM_ID* are related to the JSP (JavaServer Pages) application framework and are automatically reinitialized. The *JSESSIONID* cookie is the known state token for any JSP page, which explains the reinitialization of all missing cookies immediately following its removal. It appears as though the application utilizes the *JSESSIONID* cookie to primarily maintain the shopping cart state, and that the *order* cookie, while maintaining the shopping cart in the absence of

JSESSIONID, has no real function in the presence of the *JESSIONID* cookie.

```
http://fossil.com


 1: HTTP/1.1 302 Found
 2: Server: Microsoft-IIS/5.0
 3: Date: Thu, 19 Apr 2007 17:27:01 GMT
 4: Location: http://www.fossil.com/jump.jsp?itemID=0&itemType=HOME_PAGE
 5: Content-Length: 97
 6: Set-Cookie: JSESSIONID=lyBJchFIJ5pg; Path=/
 7: Content-Type: text/html

http://www.fossil.com/jump.jsp?itemID=0&itemType=HOME_PAGE


 1: HTTP/1.1 200 OK
 2: Server: Microsoft-IIS/5.0
 3: Date: Thu, 19 Apr 2007 17:27:29 GMT
 4: ETag: "AAAARIK4msa"
 5: Last-Modified: Thu, 19 Apr 2007 17:27:29 GMT
 6: Pragma: No-cache
 7: Expires: Thu, 01 Jan 1970 00:00:00 GMT
 8: Cache-Control: no-cache
 9: Set-Cookie: order=106378988; Path=/; Expires=Sat, 19-May-2007
      17:27:29 GMT
10: Set-Cookie: customer=137490762; Path=/; Expires=Fri, 17-Apr-2015
      17:27:29 GMT
11: Set-Cookie: mmlID=120287474; Path=/; Expires=Fri, 17-Apr-2015
      17:27:29 GMT
12: Set-Cookie: lastJVM_ID=secondary; Path=/
13: Set-Cookie: lastJVM_ID=secondary; Path=/
14: Set-Cookie: JSESSIONID=rLEX5MI49kpb; Path=/
15: Content-Type: text/html
```

Figure 3-9. HTTP Headers from the Response to the Request for *http://fossil.com*

The *customer* cookie, unlike the *order* cookie, will recreate itself when it is needed. The primary role of this cookie appears to be the identification of user-specific profiles within the site. The cookie is recreated at two points in execution, during the checkout procedure and when a user logs into the site. The *customer* cookie, if deleted, is recreated upon entering the checkout procedure or viewing user account pages. This cookie is used to identify a user profile within a session, granting access to user specific information such as a product wish-list, address book, order history and most critically, stored credit-card and billing information. As with the *order* cookie, once the initial identity is established (in this case at the login or checkout pages) the *customer* cookie is no longer necessary and will not be recreated except in the absence of the *JSESSIONID* cookie.

Unlike the *order* cookie, the *customer* cookie does not provide the ability to sustain a user session without the presence of *JSESSIONID*; if the *JSESSIONID* cookie is deleted all customer identification state is lost regardless of the presence of the *customer* cookie. This behavior further suggests that the *JSESSIONID* is the primary state identification mechanism and that the other cookies needlessly add increased complexity and limited redundancy to the system. The function of the other first-party cookie *mmlID* was unidentifiable, as the cookie was never individually recreated at any point, and in the absence of the cookie the application continued to function without error.

3.3.1.3 Cookie Theft Testing: Knowing the Security Risks

Due to the prevalence of the *JSESSIONID* cookie and its role within the web application, a simple security test was undertaken regarding the theft of this seemingly omniscient cookie:

1. A user accesses the application through *http://www.fossil.com* and clicks the "View Your Account" link.
2. The user is then presented with a login form and logs into the system. Once logged in, the system is verified to be inside the user account page where sensitive private information is available.
3. The value of the *JSESSIONID* cookie is *stolen*, i.e. the value is stored to a temporary value.
4. All cookies are removed from the user-agent.
5. Steps 1-3 are repeated to ensure that user account information is no longer available, providing a base-case against which the final assertions are validated.
6. All cookies are removed from the user-agent and the stolen *JSESSIONID* cookie is reintroduced.
7. Steps 1-3 are repeated to verify that the sensitive information is not available (result of step 5).

If correctly executed, this scenario verifies that an unauthorized party is not allowed to access user information via a stolen *JSESSIONID* cookie.

When the above scenario was implemented, the test case failed at step 7, revealing the possibility of unauthorized access to user information. This finding was repeatable, and further investigation revealed that this vulnerability was present for connections coming from various IP addresses and user agents. If a *JSESSIONID* cookie was to be stolen, this vulnerability exists from any remote host with the stolen cookie. This is particularly disconcerting, considering that cookies are subject to numerous cross-site scripting vulnerabilities present in applications across the Internet (Cgisecurity.com, 2002; Cook, 2003). This vulnerability hinges explicitly upon the ability for the *JSESSIONID* cookie to be stolen, and the fact that the user has not properly logged out of the system. This risk must be properly understood and acknowledged by the stakeholders of the application, and deemed to be within acceptable risk criterion (Tappenden, Huynh, Miller, Geras, & Smith, 2006). If the risk is deemed to be too great, steps must be taken to minimize the risk, such as greatly restricting the period of time over which any such cookie is valid, and ensuring that both the user agent (browser version) and IP address are validated against the cookie. These suggestions, although not completely eliminating the risk, greatly reduce the risk of vulnerability exploitation. This type of testing involving cookies needs to be undertaken to ensure that the principle risks and implications of using such technology are fully understood and mitigated.

3.3.1.4 First-Party Surrogates & Third-Party Analytics

The final five first-party cookies, not set as part of the HTTP headers, share more in common with the third-party cookies than with the previous six first-party cookies. These cookies are set by two embedded third-party JavaScript scripts and are the product of Google Analytics (2007) and Yahoo! Search Marketing (2006), the web-analytics branches of the two

most popular search engines on the Internet (comScore Inc., 2007a; Nielsen//NetRatings, 2007). The script attributed to Yahoo! was responsible for setting the single cookie with the name *ysm_CKAJTAOCVD4V5RMM2B78CVG78KJ*. From examination of the script that was used to deposit the cookie, it appears that the latter portion of the cookie name is a reference to the ID of the client, in this case identifying the domain *www.fossil.com*. This cookie is repeatedly set on each page and is most likely used to track a user's browsing habits creating a customer profile based on the merchandise viewed or purchased.

Google Analytics, on the other hand, is responsible for setting four first-party cookies: *__utmz*, *__utmc,* *__utmb*, and *__utma*. These cookies are set by the embedded third-party JavaScript and appear to work in conjunction to track a user's actions across a site. The four cookies share the same sequence of numbers, *193564346*, within their respective values. This sequence most likely represents the ID of the host *www.fossil.com*, as this number remains constant across multiple sessions, regardless of client browser and IP address. The cookies *__utmb* and *__utmc* both contain the same value, but appear to have different functions. The *__utmb* cookie appears to be the primary cookie used to track the time a user spends viewing any one page, as it is the only cookie of the four that is continually reset on every page request. With the *_utmb* cookie having a 30 minute expiry time, the third-party JavaScript can identify whether a user has viewed the page for a period of less than or greater than 30 minutes. The *__utmc* cookie appears to have a similar function, as it is the only sessional cookie, and its presence would alert Google Analytics to the fact that the user has not closed the browser window between browsing pages.

The other two cookies *__utmz* and *__utma* have much more complicated values. The two cookies do share a second commonality, the number *1175110911*, which appears to be an identifier unique to the user. These two cookies, containing similar information, have very different

expiry dates. The __*utmz* cookie is set to live for just under 6 months, and the __*utma* cookie is set to live for over thirty years. It appears that these expiration values were deliberately selected as they do not share any commonality with the dates presented in Section 3.2.8. One possible explanation for the dates is to avoid the deletion of the cookies based upon a time-to-live policy.

The cookies set by Google Analytics and Yahoo! Search Marketing raise a number of interesting questions about the nature of first-party cookies. These cookies, although set for the domain *www.fossil.com*, are explicitly used as part of a third-party analysis system and clearly illustrate the increased complexity novel cookie use adds to an application with respect to privacy, security, functionality and testing perspectives. This increased complexity, and usage of surrogate first-party cookies by third-party agents, suggests that third-party cookies are becoming less desirable to advertisers. This is a puzzling phenomenon, especially in the light of recent studies that suggest that the majority of users who habitually delete cookies delete both first- and third-party cookies equally (comScore Inc., 2007b).

3.3.2 Error, Fault, Failure: Examples of Incorrect Cookie Assumptions

Cookies, as the primary state mechanism for any web application, require careful consideration during the design, implementation, and testing phases of the software development lifecycle. Frequently web applications are initially developed with a cavalier attitude that elevates time-to-market pressures above application robustness, resulting in an ever-increasing number of security and privacy faults. Due to the extreme time-to-market pressures thrust upon web applications, cookies are often not given adequate consideration. This section will focus upon SQL injection vulnerabilities present within many web applications through the malicious modification of cookies. A number of real world examples will

be presented illustrating the faults found across a diverse selection of web applications leading to SQL Injection vulnerabilities (failures). From these faults a number of erroneous assumptions regarding cookies will be discussed, further strengthening the case for the inclusion of cookies within any web application testing strategy.

SQL Injection is defined as *"an attack technique used to exploit web sites by altering backend SQL statements through manipulating application input*" (Auger et al., 2005) and is inevitably tied directly to the passing of an unsanitized input into a SQL query. Although all of the failures discussed are traced to this error, analysis of the underlying faults reveal a number of insidious and incorrect assumptions regarding cookies. The faults presented may seem trivial; however, these faults are all too common and are an unfortunate reality of the software development paradigm.

Cookies, much like other inputs into a web application, are subject to malicious input manipulation and require server-side verification. Although this principle is basic, web applications continue to be plagued by input manipulation vulnerabilities, with attacks present across all types of inputs. These vulnerabilities have been discovered across the broad spectrum of web applications, both within open source and commercial applications, and across a wide array of web programming languages. Despite the heightened awareness of input manipulation vulnerabilities in the web development community, applications continue to be plagued by these types of vulnerabilities, especially with respect to input from cookies. Unfortunately, cookies are often overlooked when implementing secure web applications, as is the case with Web+Center 4.0.1, and ASP based web application (SecuriTeam, 2004). Web+Center is of particular interest because the application provides input verification for inputs for GET and POST requests, but does not filter cookie input. This oversight reveals the erroneous assumption that input from cookies is always valid. This vulnerability presents the opportunity to gain administrative privileges

within the application. Despite the efforts of the developers to verify GET and POST inputs, cookie input remains unverified and can lead to a serious breach in security. All of the subsequent examples presented in this section share this common error at their core.

```
$login_per_cookie = false;
if(isset($_COOKIE['ws_auth']) AND !isset($_SESSION['ws_auth'])) {
    $login_per_cookie = true;
    $_SESSION['ws_auth'] = $_COOKIE['ws_auth'];
}

[…]

if(stristr($_SESSION['ws_auth'], "userid")===FALSE){

    $authent = explode(":", $_SESSION['ws_auth']);
    $ws_user = $authent[0];
    $ws_pwd = $authent[1];
    $check = safe_query("SELECT userID FROM ".PREFIX."user WHERE
        userID'$ws_user' AND password='$ws_pwd'");
    while($ds=mysql_fetch_array($check)) {
        $loggedin=true;
        $userID=$ds['userID'];
}
```

Figure 3-10. Code extracted from webSPELL v4.0 (Verton, 2007)

Many web applications attempt to validate a cookie based solely on its existence. This type of check demonstrates the lack of understanding surrounding cookies, and further suggests that developers do not acknowledge the potential threat that unchecked cookies pose. A number of web applications such as myBB (SecuriTeam, 2008), webSPELL (Verton, 2007), and NukeSentinel (Vind, 2007) all have security vulnerabilities stemming from the incorrect assumption that if a cookie exists, its value is valid. Clearly this assumption is false, but these errors remain prevalent within released applications. An example of a typical fault involved with these types of failures has been extracted from the webSPELL application and is provided in Figure 3-10. The application checks only to see if the cookie exists, and fails to assess the validity of the data stored within the cookie before the value is passed into the database.

There are a number of factors that affect cookie usage within a web application. Browser version and configuration are the two most prominent client-side concerns when developing web applications, and

browser compatibility testing is often an integral component to web-application development. Server-side configuration validation is also required for any web application that is to be deployed on a third-party server. Applications that fail to validate server configurations are vulnerable to all forms of malicious input modification; such is the case for a number of vulnerabilities present in the following applications: PlaySMS (Rathaus, 2004), phpCoin (Secunia, 2005b), e107 (Secunia, 2006), PaFileDB (Secunia, 2005a), and myBB (SecuriTeam, 2008). Each of these applications rely on two server configuration options, *register_globals=off* and *magic_quotes_gpc=on*, neither of which were verified. The *register_globals* configuration provides access to any POST, GET or cookie value as though it were a global variable. If set, this configuration option represents a security vulnerability for any uninitialized variable that is passed into an SQL query, and has been linked to both SQL injection and remote code execution vulnerabilities. Such is the case for the application myBB version 1.3 (SecuriTeam, 2008). An attacker can fabricate a new cookie and use it to pass a non-sanitized input into the database. This vulnerability is truly unique from the others presented, as it provides the application with a spontaneous malicious cookie. That is, the cookie containing the payload used to compromise the system is never created by the application. Rather, the cookie is carefully crafted by the attacker to take the place of the uninitialized variable used within the application. This type of vulnerability goes far beyond simple cookie tampering, and suggests that applications are vulnerable to spurious cookie creation.

The *magic_quotes_gpc* is a server configuration option that a large number of web applications rely upon. This option filters all text input (GET, POST and cookie) into the system, and automatically escapes all ' (single-quote), " (double quote), \ (backslash) and NULL characters with a backslash (PHP Group, 2008). This option is often relied upon by developers as an input tampering security counter-measure. Despite this

reliance, a number of applications fail to verify the server configuration, leaving the applications vulnerable on servers with differing configurations. The latest version of PHP has removed support for the *magic_quotes_gpc* option, effectively disabling this security measure on all servers with the latest software, and heightening the need for server configuration verification within a web application.

Base64 encodings are a double-edged sword within web development. On one hand they provide obfuscation to sensitive values stored within cookies and can act as a deterrent to malicious users; on the other they allow un-decoded malicious strings to pass into the application through security counter-measures, such as the *magic_quotes_gpc* configuration. Such is the case in EazyPortal (Iron, 2008), NukeSentienal (Vind, 2007), and RevokeBB (BlackHawk, 2007), each of which contains an SQL Injection vulnerability due to Base64 encoded strings passing through security countermeasures and then being decoded and passed unsanitized into a query. NukeSentienal v2.5.12 is a good example of this type of vulnerability. NukeSentienal is a wrapper for the PHP-Nuke content-management application that exists to provide a subsequent layer of protection to the notoriously vulnerable PHP-Nuke. This example reveals a number of potential incorrect assumptions:

1. The author(s) assume that the cookie value is predictable and valid.
2. The author(s) assume that an existing non-empty cookie is valid.
3. The author(s) assume a false sense of security due to the use of Base64 encodings, believing that the obfuscation guarantees cookie integrity.
4. The authors(s) assume that the *magic_quotes_gpc* configuration sanitizes every incoming value; an assumption which is false due to the nature of a Base64 encoded string.

Due to the security focus of the NukeSentienal application, it is unlikely that the vulnerability stems from the first two assumptions. It is likely that the use of Base64 encodings provided the developers with a false sense of

security. Although the precise error associated with each of the presented examples cannot be pinpointed, these examples illustrate the general lack of understanding with respect to cookie deployment that exists within the web development community. As attested to in this section, the lack of understanding exists across technological platforms (PHP, ASP, JSP, etc.), and is observed amongst a wide variety of applications. Considering the faults and vulnerabilities presented, testing strategies that explicitly address cookies are required to address the potential vulnerabilities that arise from the deployment of cookie technology.

3.3.3 Case Study: A Simple eBay Bidding Scenario

To further illustrate the increased complexity cookies add to web applications, the popular online auctioneering site eBay will be considered. eBay[9] uses a number of cookies in the provision of services to both online buyers and sellers, providing a framework across which many online storefronts choose to operate. This case study will document the usage of cookies by eBay across a simple session, suggesting areas in which unique testing paradigms arise.

When first browsing to the URL *http://www.ebay.com*, 13 cookies are deposited on the client machine (See Table 3-27). These cookies originate from two unique hosts, *.ebay.com* and *.main.ebayrtm.com*. Four of the cookies associated with *.ebay.com* were set within HTTP headers; *dp1*, *nonsession* and *s* in response to the initial GET request, and *npii* in a subsequent GET request. Although the *npii* cookie is defined for the host *.ebay.com*, it was set by the host *rover.ebay.com* whose domain matches *ebay.com*; the reverse however, does not hold true. Due to this relationship, the *npii* cookie is explicitly cast to domain *.ebay.com*, highlighted in Figure 3-12. The *npii* cookie also overwrites its associated *path* value, since by default cookies are set to the path contained within the initial GET request which, in this case, is specific to the third-party

[9] http://www.ebay.com

host. Due to the third-party nature of this cookie, a P3P policy is attached to the response. This policy is included to ensure the successful deposit of the *npii* cookie and was not included in initial response from *.ebay.com* as it is not required for the deposition of first-party cookies. The two other cookies deposited by *.ebay.com—ebay* and *lucky9,* originated from embedded JavaScript scripts within the HTML document from the initial response.

Table 3-27. Cookies From Initial Request to *http://www.ebay.com*

Host	Name	Value	Expiration
.ebay.com	dp1	bspref/-1.47c48478.14b85bd80^u1p/QEBfX0BAX19AQA**49a5b7f8^	364 days
.ebay.com	ebay	%5Ejs%3D1%5Ecos%3D1%5E	Sessional
.ebay.com	npii	btguid/57a57746118oa0b583b47964ffe42a5f49a5b7f9^	364 days
.ebay.com	lucky9	6887388	1 year, and 364 days
.ebay.com	s	CgAD4ACBHxdX4NTdhNTc3NDYxMTgwYTBiNTgzYjQ3OTY0ZmZlNDJhNWZlFza0	Sessional
.ebay.com	nonsession	BAQAAARg5ShKPAAaAAMsAAUfEi4AxAPIAAUfFCnAxAM0AIFEqhfg1N2E1Nzc0NjExODBhMGI1ODNiNDc5NjRmZmU0MmE1ZroRF4vFwlUCjtxA/jfQ5zZr0B9R	364 days
.main.ebayrtm.com	HT	1204061304787%02220%0457914%0636658%03433%0460682%0633032%03245%0432658%0622908%03224%0462089%0638898%03223%0432336%0622773%03222%0458278%0635533%03221%0460969%0638009	Sessional
.main.ebayrtm.com	RUA	D1AQAAARg5ShKPAAY80Z%2FwGUeE4AwxTLQfMC8I3J%2BgNtbwLuQA17hi1vK0RMORRjKS3AhEyigomE%2Fzi3RtXz2cfqwTb8wsFh0M94DANX%2F%2FomCbt8apAAvLcGJentgyr37IyKExSfoeCASM5KeVtMEH	1 year and 364 days
.main.ebayrtm.com	TC01	gB0OeleFGBAAAFADyW7AAAAAAAAMgmhnAgeYMgUTMguDcAIzCFgA	364 days
.main.ebayrtm.com	M01	AIAQgABgAE	364 days
.main.ebayrtm.com	A01	QBUAMIbtDAAAAAAAwMDPfYMqJm7AnZhC	364 days
.main.ebayrtm.com	C01	AAAAQ57AAAAAAAAA	364 days
.main.ebayrtm.com	PS	T.0	364 days

The final seven cookies were all deposited from the host *.main.ebayrtm.com* and set by HTTP response headers, as shown in

Figure 3-11. Similar to the *npii* cookie, these headers have an associated prerequisite P3P policy ensuring that the cookies can be placed within any browser implementing a P3P agent, primarily Internet Explorer as discussed in Section 3.2.10. Despite the similarities, the P3P policy associated with the *npii* cookie is different from that associated with the seven *.main.ebayrtm.com* cookies. This difference in P3P policy creates three unique sets of cookies: those without P3P policy, those with the policy from *rover.ebay.com*, and those with the policy from *main.ebayrtm.com*. This further cookie partition enhances the complexity of testing issues surrounding cookies for the site, as cookies can be rejected on the basis of the three policies.

```
GET http://srx.main.ebayrtm.com/rtm?RtmCmd&a=json&p=223:224:220:221:222:226:245:43
3&g=61175c3b1180a0b58376cb86ffdeb19a&uf=0&c=1H4sIAAAAAAAAAFMOyShVcEtNUjCyUDA0sDKyt
DIyVvANDlEwMjCw4OUqMLQwtjW2NDDn5UrOTLE1NIopNTAwMOTlAgB1phb%2BOAAAAA%3D%3D&ord=1204
219763943&e=USC:1&z=7&bw=950&enc=cp1252&cb=parent.vjo.dsf.assembly.VjClientAssembl
er._callback0

   HTTP/1.x 302 Moved Temporarily
   Server: Apache-Coyote/1.1
   Cache-Control: no-cache
   Expires: 0
   P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC
   NOI DSP COR"
   Set-Cookie: PS=T.0; Domain=main.ebayrtm.com; Expires=Fri, 27-Feb-2009
   [...]

GET http://srx.main.ebayrtm.com/rtm?RtmCmd&a=json&p=223:224:220:221:222:226:245:43
3&g=61175c3b1180a0b58376cb86ffdeb19a&uf=0&c=1H4sIAAAAAAAAAFMOyShVcEtNUjCyUDA0sDKyt
DIyVvANDlEwMjCw4OUqMLQwtjW2NDDn5UrOTLE1NIopNTAwMOTlAgB1phb%2BOAAAAA%3D%3D&ord=1204
219763943&e=USC:1&z=7&bw=950&enc=cp1252&cb=parent.vjo.dsf.assembly.VjClientAssembl
er._callback0&r=yes

   HTTP/1.x 200 OK
   Server: Apache-Coyote/1.1
   Cache-Control: no-cache
   Expires: 0
   P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC
    NOI DSP COR"
   Set-Cookie: C01=AAAAQ57AAAAAAAAA; Domain=main.ebayrtm.com; Expires=Fri, 27-Fe
   b-2009 17:29:24 GMT; Path=/rtm
   Set-Cookie: A01=QBUAJMOeFAAAAAAAQ6hHRxpVus1o7K; Domain=main.ebayrtm.com; Expi
   res=Fri, 27-Feb-2009 17:29:24 GMT; Path=/rtm
   Set-Cookie: M01=AAIQAIAAAABCQ; Domain=main.ebayrtm.com; Expires=Fri, 27-Feb-2
   009 17:29:24 GMT; Path=/rtm
   Set-Cookie: TC01=gBwCYXEGGBAAAFQCjjXBAAAAAAAkAA9wDQRxBA1KXAAAAxWDQdXBQ; Domai
   n=main.ebayrtm.com; Expires=Fri, 27-Feb-2009 17:29:24 GMT; Path=/rtm
   Set-Cookie: RUA=D1AQAAARg5ShKPAAaQyMohDT%2Fv4Iz9g%2Bc5BWxpdhSnxOSneTtMdfSrqCd
   6VYAsOo%2Fy4%2BnZrAoHjfYr70jfl1Inb4xWsps4pKaxrjQyKxJTWTQhXnP%2BUc%2Ba8fmpQJcS
   1Nqg9Mzhw7m40spDTtrr30e%2F; Domain=main.ebayrtm.com; Expires=Sat, 27-Feb-2010
    17:29:24 GMT; Path=/rtm
   Set-Cookie: HT=1204219763943%02220%0457914%0636658%03433%0460682%0633032%0324
   5%0432658%0622908%03224%0462089%0638898%03223%0458173%0636917%03222%0458278%0
   635533%03221%0458895%0637139; Domain=main.ebayrtm.com Path=/rtm
   [...]
```

Figure 3-11. HTTP Response Headers from *http://srx.main.ebay.com*

```
GET http://www.ebay.com/

   HTTP/1.x 200 OK
   Server: Microsoft-IIS/5.0, Apache-Coyote/1.1
   Date: Thu, 28 Feb 2008 17:29:23 GMT
   Cneonction: close
   Set-Cookie: dp1=bspref/04b895673^u1p/QEBfX0BAX19AQA**49a822f3^; Domain=.ebay.
   com; Expires=Sat, 27-Feb-2010 17:29:23 GMT; Path=/
   Set-Cookie: nonsession=BAQAAARg5ShKPAAaAAMsAAUfG9nsxAPIAAUfHrXAxAMoAIFEs8PM2M
   TE3NWMzYjExODBhMGI1ODM3NmNiODZmZmRlYjE5YeQn8mxMkByfuN1NYgnkGHF+BIRw; Domain=.
   ebay.com; Expires=Fri, 27-Feb-2009 17:29:23 GMT; Path=/
   Set-Cookie: s=CgAD4ACBHyEDzNjExNzVjM2IxMTgwYTBiNTgzNzZjYjg2ZmZkZWIxOWFp6sfS;
   Domain=.ebay.com; Path=/
   [...]

…

GET http://rover.ebay.com/roversync/?site=0&mpt=1204219763771&tGuid=61175c3b1180
a0b58376cb86ffdeb19a

   HTTP/1.x 200 OK
   Server: Apache-Coyote/1.1
   Set-Cookie: npii=btguid/61175c3b1180a0b58376cb86ffdeb19a49a822f4^; Domain=.eb
   ay.com; Expires=Fri, 27-Feb-2009 17:29:24 GMT; Path=/
   P3P: CP="NOI CURa ADMa DEVa TAIa OUR BUS IND UNI COM NAV INT"
   [...]
```

Figure 3-12. HTTP Response Headers from *http://www.ebay.com* & *http://rover.ebay.com*

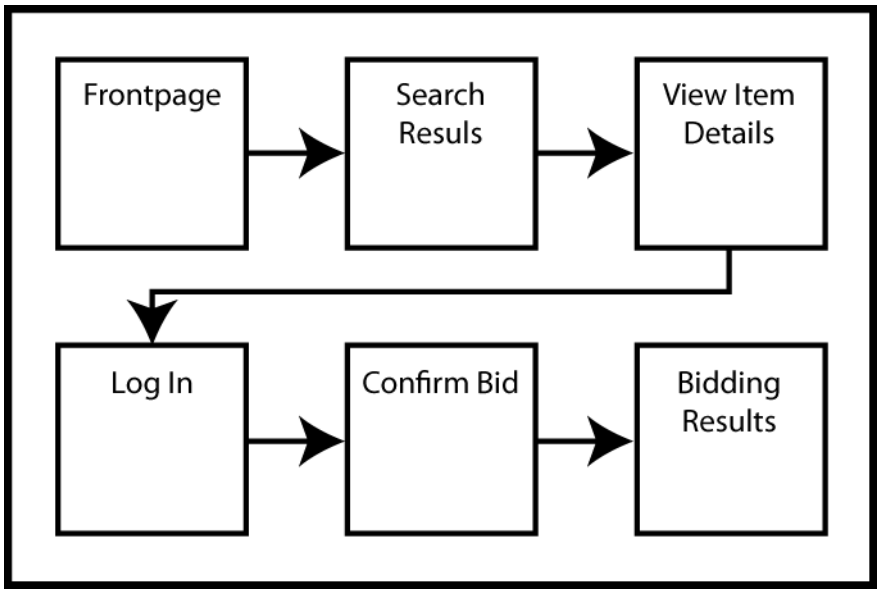


Figure 3-13. Sequence of Requests Required to Bid on a Single Item.

To further explore the use of cookies by eBay, the simple usage scenario of bidding on an item was executed and the accumulated cookies were retrieved. This simple scenario, mapped in Figure 3-13, consisted of five user-initiated requests from the initial page *http://www.ebay.com*, for a total of 6 loaded pages. After execution of this scenario, the number of cookies stored within the browser jumped to 24 due to the creation of 11

subsequent cookies presented in Table 3-28. The cookies originated from five distinct hosts, two of which were responsible for the original 13 cookies, *.ebay.com* and *.ebayrtm.com*. Third-party hosts set more than half of the cookies, and all of the third-party cookies were persistent, with expiration values ranging from 30 days to over 29 years. The exact purpose of each of these cookies could not be identified due to the techniques employed to obfuscate the values of the individual cookies; it is clear that the use of cookies by eBay goes beyond a simple state-identification token.

Table 3-28. Additional Cookies Deposited By the Bid on Item Usage Scenario

Host	Name	Value	Expiration
.ebay.com	ds1	ats/1204136160546	Sessional
.ebay.com	ds2	alss/0.foo%2bar47c6fa90^	Sessional
.ebay.com	npii	btguid/5c1a15bd1180a0b583966e06ffe33 26049a6dbf8^	1 year and 364 days
.ebay.com	secure_tick et	dXNlcmlkPTIyMjI2MTM5NXxwYWdldH lwZT0yMTQyfHRpbWU9MTIzNTY3NTc 2MHxsYXNoX21vZD0xMjAyNjg0MDkyf HVpX2xhc3RfbW9kPTExMjQyMTE4OT N8cHJlZj0xNjM4NHxtYWM9UVJXR25 XMzBxTkxSaTFyMTJOTnhGMQ**g	Sessional
.ebay.com	cid	VMuhA8d9FAEaoGDs%231462259156	1 year
.yahoo.com	B	0g323ml3sba3u&b=3&s=h2	29 years 103 days
.apmebf.com	S	83kmku-795543469-1204136208667-po	4 years and 363 days
.mediaplex.com	mojo1	s/71151818010/70	1 year 115 days
.mediaplex.com	svid	58222708638	1 year 115 days
.ebayrtm.com	CT	3.d2cf.47c5a8e1.fffffffffffd8f1.1.47c5a94c :	30 days
.ebayrtm.com	RUP	D1AQAAARg5ShKPAAY1xy5QqZ%2FoLV XoI7EivozPrgpSLkzts1C7A%2FeU72%2B BTIHBfid062tguwidZyUhYZMyaKfa	1 year and 364 days

3.4 Summary of Results and Key Findings

There are many strategies currently established in the literature for testing web applications. However, none of these strategies provide an effective means for dealing with cookies. The results of this study have shown that more than two thirds of the most popular Internet sites utilize cookie

technology. This value is much higher amongst the subsets of Internet sites which utilize dynamic web technologies and P3P polices. Furthermore, we see that this technology is utilized by web applications regardless of their popularity. Cookies are more frequently used on the Internet than JavaScript, StyleSheets Frames, Flash/Shockwave, GIFs, JPEGs, and PNGs (Security Space, 2006b). It is clear that cookie technology has become a staple of modern web applications.

The results of this study also highlight the complex role that cookies play within web applications. Although the mode of the number of cookies used per site was one, of the majority of sites surveyed that utilized cookies (72.4%) deployed more than one cookie. This suggests that web applications use cookies for more than just simple state identification. Furthermore, 52.1% of sites using multiple cookies used a mixture of both first- and third-party cookies, and 74.1% utilized a combination of persistent and sessional cookies. These numbers suggest that the usage of cookies in web applications is not as simplistic as previously believed. The survey found that the maximum number of cookies used for a single site was 76. Clearly, cookie usage within many web applications is prolific and complex, and adequate testing of these applications requires a strategy that explicitly addresses the nature of cookie usage on the Internet.

Chapter 4

Cookie Usage Amongst Nations[10]

Cookies are intertwined with modern online technologies, and are used extensively as the basic building block for state-based web applications. Despite this prevalence, very little research exists which specifically investigates cookies and their relationship with the technological aspects of web-applications. This Chapter will present the first study to directly compare cookie usage and country of origin, providing a unique picture of the technological and economic environments associated with each country. The Alexa top 100,000 list (2006b) presented in Chapter 3.1.1 was leveraged as the basis for this study. The data collected from within the cookie deployment study was augmented with geographic location information to provide insight into cookie deployment with respect to country of origin. The results of this survey establish a significant relationship between the deployment of cookies by web applications and the maturity of a country's e-commerce environment.

The remainder of the chapter will be organized as follows: Section 4.1 will present the survey methodology and specific research questions; Sections 4.2 – 4.4 will present the results of the investigation, and Section 4.5 will summarize the key findings, highlighting the most important results obtained within the survey.

---

[10] *A version of this chapter has been published.* Tappenden, A. F., & Miler, J. (2009.). A Survey of Cookie Technology Usage Amongst Nations. *Journal of Web Engineering, 8*(3), 211–244.

4.1 Survey Methodology

4.1.1 Research Questions

An explosion in the number of global Internet users has been observed, especially within regions in which the Internet has had little penetration (Miniwatts Marketing Group, 2008). Since 2000, Internet penetration has risen over 300% globally, and over 1000% in the least penetrated regions, Africa and the Middle East (Miniwatts Marketing Group, 2008). Given the increasingly global reach of Internet-based opportunities, many e-commerce companies are interested in targeting a truly global audience. These opportunities, unique to the Internet, present the ability to create a multi-national web presence with little overhead. Concepts such as the *Internationalisability* of a software application are gaining momentum and becoming a principle concern for any global web application (Vijayaraghavan & Kaner, 2003). This heightened interest in global web applications requires an increased knowledge of the global technological landscape, especially with regard to the technological platforms and normal usage patterns of web-specific technologies such as cookies. To this end, a number of specific research questions were selected to provide a better understanding of the usage of cookies both globally and within specific nations.

The following research questions motivate this study:

Q1. Are cookies equally utilized across the globe?

Q2. Given the explicit privacy concerns related to third-party cookies, is the usage of this technology consistent between nations?

Q3. Are cookies set to expire as soon as possible (sessional vs. persistent), and is this constant across nations?

Q4. Which web technologies dominate, and do regional variations exist?

Q5. How prevalent are online advertisers (using third-party persistent cookies as indicators), both globally and amongst nations?

Q6. How are cookies used for tracking users' movements within a site, both globally and amongst nations?

Q7. Does cookie deployment directly correlate with the maturity of a country's e-commerce environment?

These questions directly relate to the development of current and future web applications. A thorough understanding of the best industrial practices pertaining to cookie usage globally, and within the country for which a specific product is targeted, can be of substantial benefit in the development of web applications. The knowledge of empirical cookie usage is also of benefit to the larger web engineering community who seeks to advance the current state-of-the-art for these systems, specifically those who seek to produce high-quality support systems for web software endeavors. To answer these questions a survey of the most popular 100,000 sites was overlaid with the geographical origins each site, producing a pool of over 97,000 data points containing both the cookies set through the HTTP headers, images and scripts, and the geographical location from which the site originated. This pool provided ample data from which the seven research questions could be addressed.

4.1.2 Resolution of Geographic Location

The Alexa top 100,000 list (2006b) presented in Chapter 3.1.1 was leveraged as the basis for this study. The data collected from within the cookie deployment study was augmented with geographic location information to provide insight into cookie deployment with respect to country of origin. Each site surveyed was mapped to the country from which the site was hosted. The geographic location of these websites was derived by determining their IP address via the *host* program, and then comparing this address against a database of locations (i.e. countries)

provided by IP2Location (IP2Location.com, 2006). This approach was required since the country code top-level domain is not a reliable indicator of the actual geographical location. Of the 98,004 sites surveyed, 130 nations were identified as the origin of 97,050 sites. The remaining 956 origins that could not be determined represent less than 1.0% of the total population, which is within in the error rate associated with the IP2Location mapping (IP2Location.com, 2006).

4.1.3 Analysis Tools & Statistical Tests

Due to the large pool of data gathered by the survey, SPSS was used to analyze the experimental data and generate box plots. Because of the non-parametric and ordinal nature of the data, the Mann-Whitney U test was utilized to confirm the existence of significant differences between the medians of various data populations. The associated *null* hypothesis is that the two samples are drawn from the same population, i.e. populations that have equal probability distribution functions. All null hypotheses presented within the chapter were rejected if the significance of the result was below the standard Type 1 error rate, 0.05.

To further characterize the differences verified between sample populations, an effect size was calculated as an estimate of the size of the difference between the two populations. Because the data is non-parametric, Cliff's δ (Cliff, 1993, 1996), a non-parametric effect size estimate, was used to characterize the difference between two populations. Cliff's δ examines the probability that individual observations within one group are likely to be greater than the observations in the other group, given by the following equation

$$\delta = Pr(x_{i1} > x_{j2}) - Pr(x_{i1} < x_{j2}), \quad (4\text{-}1)$$

where x_{i1} is a member of population one and x_{j2} is a member of population two. This effect size has been empirically demonstrated to be superior to Cohen's *d* and Hedges' *g* when the data is non-parametric (Hess, Kromrey, Ferron, Hogarty, & Hines, 2005; Kromrey, Hogarty, Ferron, Hines, &

Hess, 2005). Essentially, this approach considers the ordinal, rather than the interval, properties of the data. The sample estimate of this statistic, Cliff's delta is obtained by comparing each of the values within in one group to each in the other. The calculation of this sample statistic is given by

$$\delta = \frac{\#\left(x_1 > x_2\right) - \#\left(x_1 - x_2\right)}{n_q n_2}, \qquad (4\text{-}2)$$

where x_1 and x_2 are the individual members of each sample population; and n_1 and n_2 are the number of individuals within each sample. Cliff's δ represents the degree of overlap between the two distributions and unlike Cohen's *d*, Cliff's effect size is bounded in the range [-1, 1] and takes the value of zero if the two distributions are identical. To provide a linguistic interpretation of these differences, we will adopt the suggested values of Romano et al. (2006) and interpret a $\delta > 0.147$ as *small*, $\delta > 0.330$ as *medium*, and $\delta > 0.474$ as *large* differences (equivalent to the levels suggested by Cohen (1992) of 0.20, 0.50 and 0.80 respectively).

Two alternatives exist for calculating the Cliff's delta associated variance. The consistent estimate of variance[11] (Cliff, 1996) is presented, allowing for the construction of two asymmetric confidence limits at the 95% confidence level around the δ value. This estimate of variance, as stated by Cliff (1996), produces highly conservative confidence intervals and hypothesis testing should not be based upon these estimates. Despite Cliff's recommendation, the effect size was utilized to compare the differences existing between groups and provide a metric for determining the relative size of the differences between populations. The risk that the tests did in fact measure beyond the 95% level is considered an acceptable risk given the nature of the analysis. For the remainder of this chapter, the following interpretation will be adopted: if zero is included within the confidence interval surrounding Cliff's δ, then the populations are

[11] Kromrey et al. (2005) empirically demonstrated that the choice of variance procedure is relatively unimportant across a wide range of circumstances.

considered equal; if the confidence interval is only negative then *Group 2 > Group 1*; if it is only positive then *Group 1 > Group 2*.

Finally Spearman's ρ, a non-parametric test, was selected to calculate the correlation between the ordinal ranking of each nation and the cookie usage within the nation. This correlation was chosen because of the ordinal nature of the ranking against which the dataset was compared. To further facilitate the analysis, the ρ values were converted into an equivalent Pearson's *r* correlation coefficient as prescribed by Gilpin (1993). These coefficients were then compared and attributed a measure of effect size—*small,* $0.1 < r \leq 0.3$*; medium,* $0.3 < r \leq 0.5$*;* and *large*, $0.5 < r \leq 1$; according to Cohen (1988, 1992).

4.2 Global Cookie Usage

The survey outlined in Section 4.1 was quite prolific in the generation of usable data from which the research questions could be effectively answered. The survey generated the cookie-usage and geographical origins of over 97,000 sites with the identification of 130 nations as the origins of these sites. Given the sheer volume of data extracted by this investigation, a smaller subset of data was selected for in-depth analysis within this chapter; however, a complete summary of the survey data is provided in Appendix A.

Appendix A presents the data from two perspectives: Table A-1 provides a breakdown of the number of sites surveyed for each country, and Table A-2 focuses on the number of cookies set by sites originating from each country. Each appendix looks at six distinct cookies classifications: all, first-party, third-party, sessional, persistent and third-party persistent cookies. It is clear from the appendices that the dataset is quite large and a large discrepancy exists in the number of sites surveyed between nations. Due to this discrepancy and the sheer size of the dataset, smaller subsets were selected and analyzed. The remainder of this section will focus upon the nations from which at least 500 sites were surveyed.

This subset contains 18 nations and represents the source of over 90% of the total number of sites surveyed.

4.2.1 Number of Sites Surveyed per Country

The number of sites surveyed per nation with at least 500 sties is presented in Figure 4-1; clearly the majority of the sites surveyed (44,673) had origins within the United States (US) with the next closest country China, with less than half the number of sites (17,196). Due to the large discrepancy in populations per country, the analysis will focus upon values normalized against the number of sites surveyed for each country.

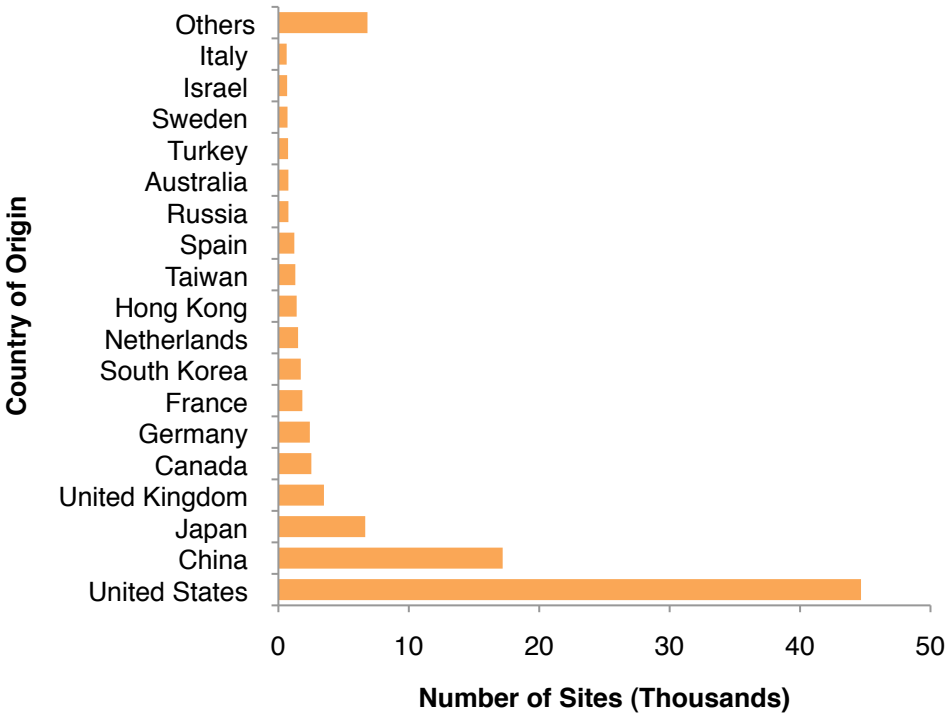


Figure 4-1. Country Frequency Histogram

4.2.2 Cookie Usage Within Each Country

The countries identified within the survey were analyzed from six perspectives, overall cookie usage, first-party, third-party, sessional, persistent, and third-party persistent cookie usage. This data is presented

with respect to the mean number of sites utilizing the specific cookie categories per country in Figure 4-2. When comparing the countries that were the origin of at least 500 sites, it is observed that Russia, Japan and Hong Kong are outliers. Russia is observed to have a higher than average number of third-party and persistent cookies; Japan and Hong Kong are observed to have a lower than average usage of cookies. Furthermore, Japan is also significantly below average for the usage of sessional cookies, whereas the mean of Hong Kong is within the 95th percentile of sessional cookies. Both Japan and Hong Kong are within the 95th percentile first-party, third-party and persistent cookies.

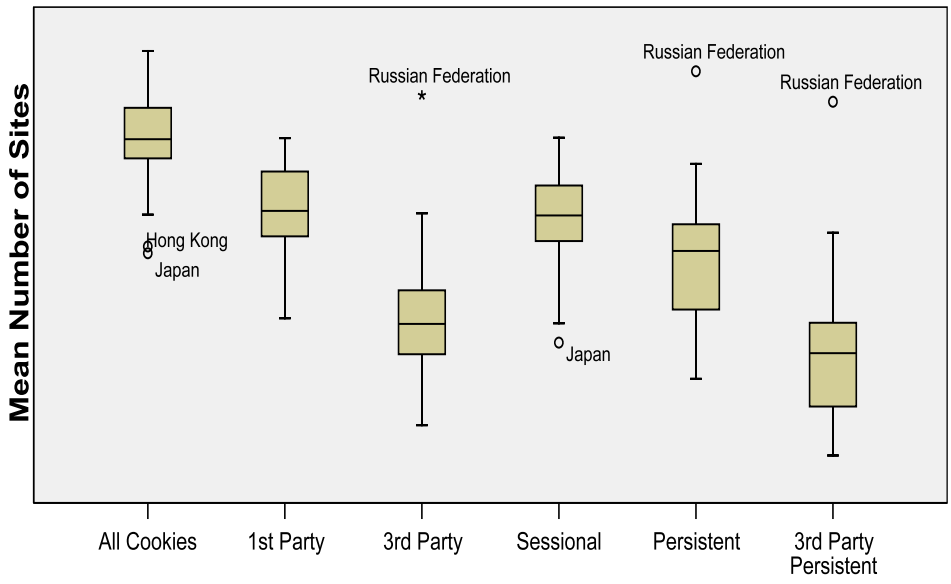
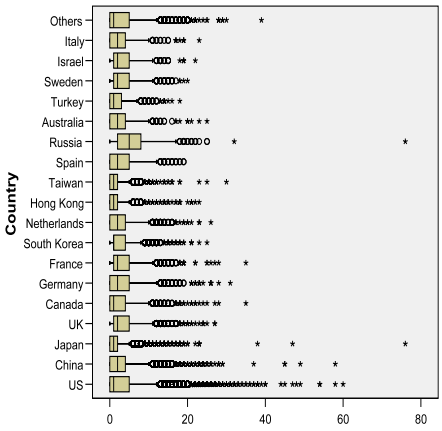


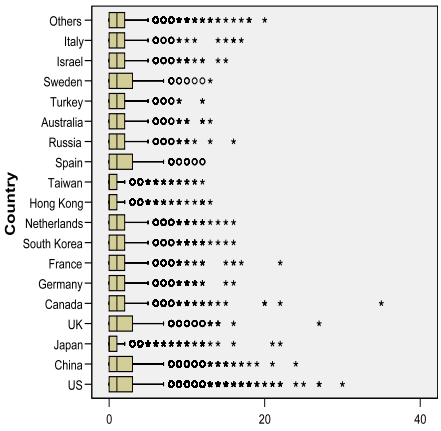
Figure 4-2. Mean Number of Sites Using Specific Types Of Cookies Per Country

To further investigate cookie usage amongst these nations box-plots for each of five types of cookies are presented in Figure 4-3. It is clear from the shape of each of the box-plots, and confirmed by the Kolmogrov-Smirnov tests presented in Table 4-1, that the distribution of number of cookies per site is not normal within each country. Due to the lack of an identifiable distribution of cookie usage per country, the remainder of the chapter will present a number of non-parametric statistical tests, as these

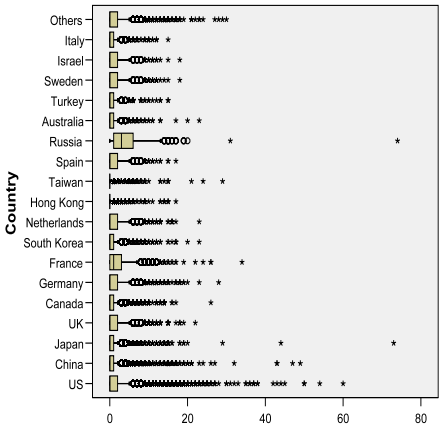
tests do not rely upon the assumption that the sample is drawn from a known distribution.



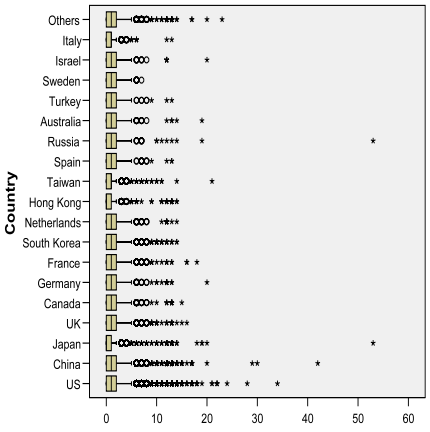
(a) Overall Cookie Usage per Country



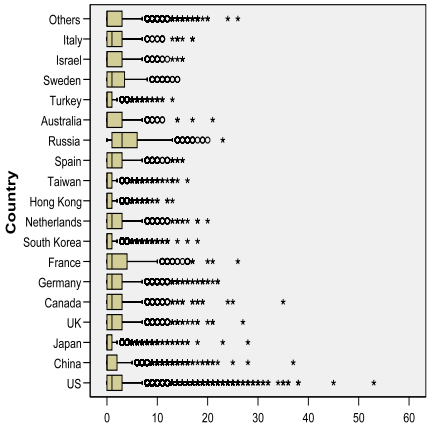
(b) First-Party per Country



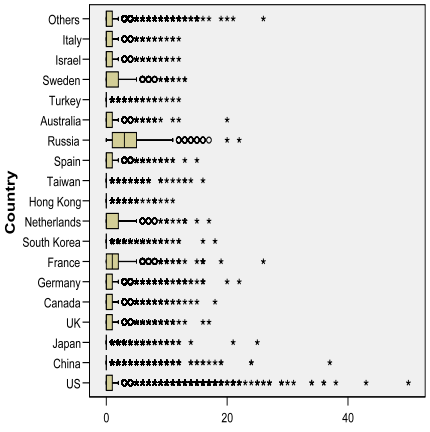
(c) Third-Party per Country



(d) Sessional per Country



(e) Persistent per Country



(f) Third-Party Persistent per Country

Figure 4-3. Cookie Usage per Country

Table 4-1. Kolmogorov-Smirnov[12] Tests For Normality

	All Cookies		First-Party Cookies		Third-Party Cookies		Sessional Cookies		Persistent Cookies		Third-Party Persistent Cookies	
	K-S	Sig.	K-S	Sig.	K-S	Sig.	K-S	Sig.	K-S	Sig.	K-S	Sig.
US	.230	.000	.259	.000	.316	.000	.267	.000	.263	.000	.352	.000
China	.211	.000	.247	.000	.336	.000	.263	.000	.289	.000	.454	.000
Japan	.281	.000	.325	.000	.376	.000	.341	.000	.311	.000	.436	.000
UK	.187	.000	.239	.000	.294	.000	.262	.000	.228	.000	.340	.000
Canada	.217	.000	.253	.000	.335	.000	.257	.000	.257	.000	.375	.000
Germany	.219	.000	.285	.000	.290	.000	.251	.000	.258	.000	.336	.000
France	.191	.000	.272	.000	.261	.000	.281	.000	.208	.000	.253	.000
South Korea	.240	.000	.271	.000	.361	.000	.269	.000	.336	.000	.423	.000
Netherlands	.211	.000	.282	.000	.286	.000	.261	.000	.240	.000	.319	.000
Hong Kong	.297	.000	.303	.000	.428	.000	.333	.000	.378	.000	.477	.000
Taiwan	.295	.000	.312	.000	.422	.000	.294	.000	.392	.000	.477	.000
Spain	.194	.000	.278	.000	.307	.000	.252	.000	.246	.000	.348	.000
Russia	.151	.000	.239	.000	.195	.000	.302	.000	.145	.000	.156	.000
Australia	.212	.000	.237	.000	.315	.000	.247	.000	.255	.000	.360	.000
Turkey	.265	.000	.315	.000	.347	.000	.267	.000	.329	.000	.412	.000
Sweden	.185	.000	.244	.000	.284	.000	.251	.000	.217	.000	.314	.000
Israel	.211	.000	.252	.000	.320	.000	.229	.000	.261	.000	.377	.000
Italy	.208	.000	.275	.000	.298	.000	.259	.000	.244	.000	.325	.000
Others	.210	.000	.278	.000	.310	.000	.256	.000	.254	.000	.356	.000

As highlighted in Figure 4-2, sites originating from Russia use significantly more cookies than sites originating from the other countries. This trend is evident in Figure 4-3, as the median number of sites from Russia using persistent cookies exceeds that of any other country, and is equal to or greater than the 75th percentile of all other countries with the exception of France. Similarly to persistent cookies, third-party cookies usage within Russia is also much more prominent, and the median of sites originating from Russia with respect to third-party cookie usage is only equaled by the 75th percentile of that of France, in fact all other countries have a 75th percentile less than the median of Russia. This increased usage of both third-party and persistent cookies can be attributed to the significant disparity in the frequency of third-party persistent cookies within sites originating from Russia, attested to by Figure 4-3f. The

[12] with Lilliefors Significance Correction

median frequency of Russian sites using third-party persistent cookies surpasses all other countries, and is above the 95th percentile of all but three countries—France, the Netherlands, and Sweden. Although Russia is observed to have a higher than average overall cookie adoption frequency, these effects are not consistent with respect to first-party and sessional cookie usage, shown in Figure 4-3b and Figure 4-3d respectively, where very little difference between Russian sites and sites from other countries exists.

Table 4-2. Mann-Whitney U Test For Russia vs. All Other Countries

	All Cookies	First-Party	Third-Party	Sessional	Persistent	Third-Party Persistent
Mann-Whitney U	2.29E+07	3.53E+07	1.80E+07	3.47E+07	2.00E+07	1.64E+07
Z	-18.952	-2.806	-28.751	-3.534	-23.876	-33.183
Asymp. Sig. (2-tailed)	0.000	0.005	0.000	0.000	0.000	0.000
Cliff's Delta	0.388	0.056	0.518	0.069	0.464	0.561
Upper Limit	0.350	0.036	0.482	0.031	0.428	0.525
Lower Limit	0.424	0.035	0.553	0.108	0.499	0.595

The Cliff's δ effect-sizes and associated confidence intervals are presented alongside the Mann-Whitney values presented in Table 4-2 and in Figure 4-4. It is clear from the effect-sizes that the largest disparity exists between Russia and the world with respect to persistent ($\delta = 0.464$), third-party ($\delta = 0.518$), and most notably third-party persistent cookies ($\delta = 0.561$). Much smaller differences exist amongst all ($\delta = 0.388$), first-party ($\delta = 0.056$), and sessional ($\delta = 0.069$) cookies. The difference with respect to sessional and first-party cookies does not warrant the classification of *small*, leading to the conclusion that the *medium* difference observed amongst overall cookie usage is due to the *large* differences existing between Russia and other countries with respect to third-party persistent cookies. A ripple effect of this difference is observed in the *medium* and *large* differences existing in persistent and third-party cookies respectively, as third-party persistent cookies is defined as *third-party* $\cap$ *persistent*. Clearly sites originating in Russia are utilizing third-

party persistent cookies at a much higher rate then other countries, these effects will be studied further in Section 4.3.2.

In direct contrast to the results observed for Russia, sites originating from Japan, Hong Kong, and Taiwan are observed to be consistently below average with respect to cookie usage. This discrepancy is most pronounced with respect to first-party and sessional cookies, where the 75th percentile frequency for Japan, Hong Kong and Taiwan are all equal to that of the median frequency for all other countries with the exception of the Netherlands. While it may be tempting to view this as a geographical issue, China does not follow this trend and South Korea posts a mixed set of results.

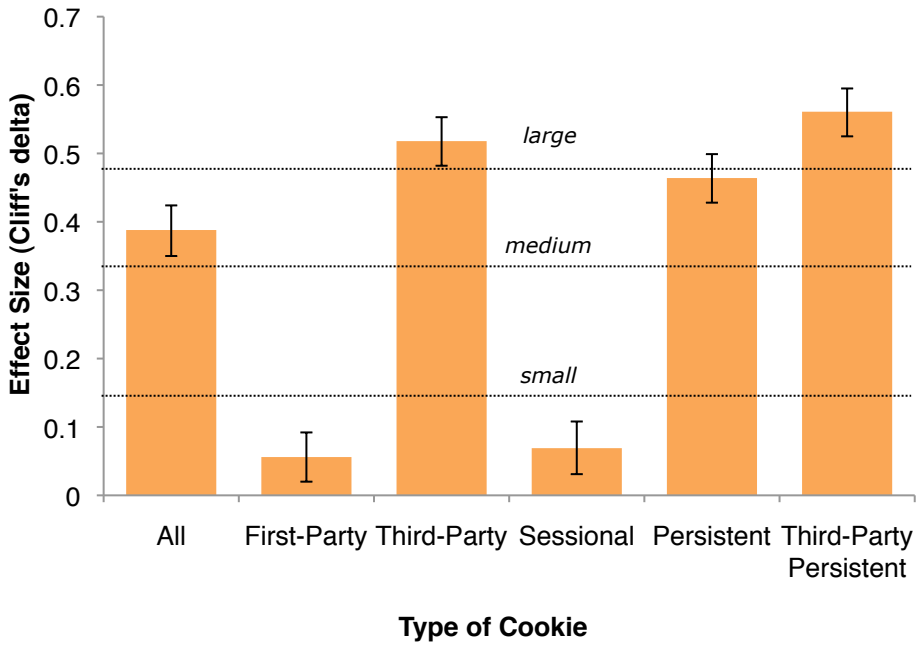


Figure 4-4. Relative Effect Sizes for Russia vs. the World

To further characterize these differences, Cliff's δ effect-sizes and associated confidence intervals were calculated for each of the eastern Asian countries[13] with the results presented in Figure 4-5. Hong Kong has

[13] Country classifications provided by the United Nations Statistical Division (2007).

the overall lowest cookie usage amongst all of the countries surveyed, as shown in Figure 4-5, this decreased overall level is due to less than average cookies usage with respect to all five categories of cookies. Despite this general decrease in cookie usage, Hong Kong is not the minimum within each category of cookies. Japan, for example, has lower sessional cookie adoption and the second lowest overall cookie adoption rate among the countries identified as the origins of at least 500 sites surveyed. Unlike Hong Kong, this *small* difference is not as widespread across all five types of cookies and is seen as the results of decreased usage of first-party and sessional cookies. Taiwan, on the other hand, is observed to have lower overall cookie adoption; however, this is primarily attributed to the *small* decrease in third-party persistent cookie adoption.

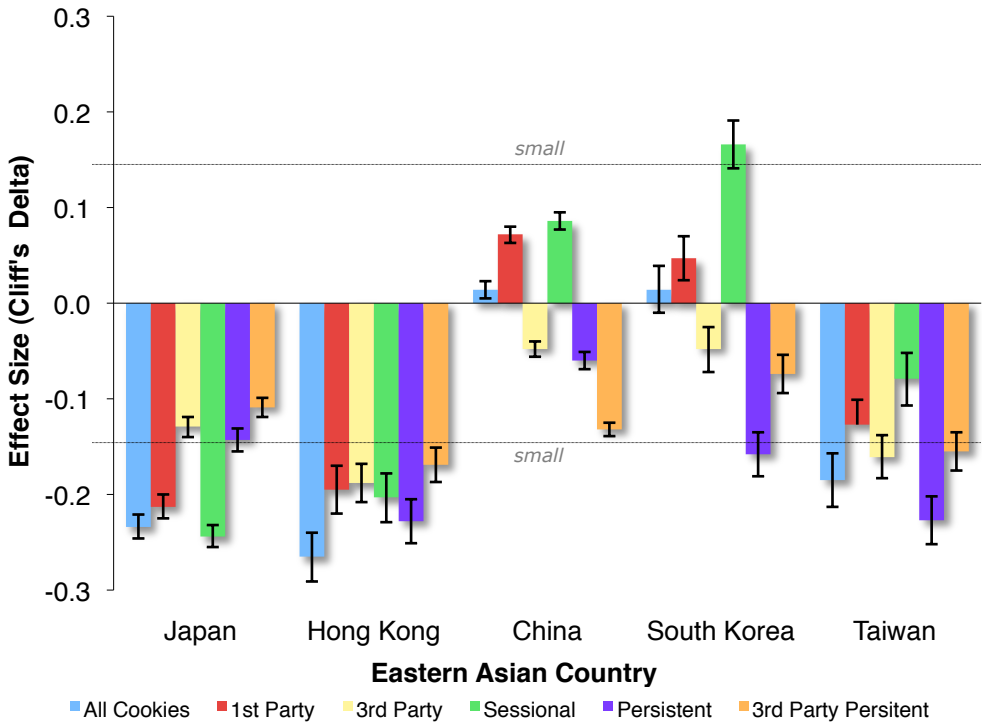


Figure 4-5. Eastern Asian Countries vs. the World

China and South Korea present very different results, neither presenting a *small* difference in overall cookie usage, in fact the Cliff's δ intervals for South Korea with respect to overall cookie usage encompass zero, indicating that with respect to overall cookie usage South Korea is

similar to that of the rest of the world. Because of the relationship between categories, *cookies = (sessional ∪ persistent) = (first-party ∪ third-party)*, the equality observed between the global distribution and that of South Korea, with respect to all cookies, manifests itself as symmetric positive and negative disparities with respect to sessional/persistent and first-/third-party cookies. China also presents a mixed set of results; however, these results are not as symmetric as those of South Korea. With respect to third-party persistent cookies, China has a larger relative disparity than South Korea; this is very interesting especially considering the relatively large disparity amongst persistent cookies in South Korea, suggesting that persistent cookie usage in South Korea is distributed more evenly between first- and third-party cookies than in China.

It is clear from these findings that the usage of cookies differs greatly based upon the geographical region in which the site originates, providing a clear answer to research question *Q1*—cookies are utilized across the globe, as cookies were found to be used in every country for which more than four sites were surveyed. Although cookies are utilized globally cookie deployment was observed to vary greatly depending on the nation from which a site originates. An answer to research questions *Q2* and *Q3* are clearly illustrated by the dichotomy of cookie usage from sites originating in Russia versus those originating in Southeast Asia, and specifically Hong Kong—the distribution of first- versus third-party cookies and sessional versus persistent cookies are not consistent across nations. While numerous reasons exist for this variance, including public perceptions of cookies, cultural sensitivities to privacy evasive technologies, the legal obligations and responsibilities of web applications, and the maturity of the e-commerce environment, it is important that anyone seeking to develop web applications within these nations understand these factors in order to correctly situate the software endeavour for success.

4.3 Commercial Off The Shelf Cookie Deployment

From the cookies collected by the survey two distinct groups emerged, those used as part of a dynamic web application framework, and those used for online tracking and advertising. Although these cookies emerged from all categories of cookies, certain specific types of cookies were strongly related to specific technologies. The discussion of these results will be partitioned into two sections: a discussion of dynamic web technologies and a discussion of online tracking and advertising technologies.

4.3.1 Dynamic Web Application Frameworks

The survey was able to identify five cookies associated directly with the state management mechanisms for five popular web application frameworks: ASP, PHP, ASP.NET, Java Servlet (JSP & J2EE), and Cold Fusion. Each of these development environments provide an internal state management mechanism relying heavily upon cookie technology. Usage of these technologies is easily identifiable by the presence of the associated technology-specific cookie: *ASPSESSIONID%*[14], *PHPSESSID, ASP.NET_SessionId*, *JSESSIONID*, and *CFID*, respectively. A closer inspection of the associated cookies, summarized in Table 4-3, reveals that the vast majority of the cookies were first-party, and all but one of the technologies (Cold Fusion) exclusively used sessional cookies. Cold Fusion uses a persistent cookie as the unique identifier for a user across a web-session. This cookie, unlike the cookies from the other frameworks, is not set to be removed when the browser is closed, rather the majority of *CFID* cookies (81.9%) were set to be stored in the browser for at least 30 years (10,950 days). The use of persistent cookies in association with dynamic web application frameworks is clearly in the minority, as the four highest occurring associated cookies were all nearly exclusively sessional.

[14] % Represents a wildcard, these cookies share the same prefix *ASPSESSION* followed by a unique identifier.

Although the majority of dynamic technology cookies were first-party, a percentage of the cookies were associated with third-party hosts, suggesting that third-party providers are using theses dynamic web technologies in the provision of third-party services. The extent to which these services are of value to the end-user and/or the service provider cannot be assessed from these findings, due to the plethora of specific third-party advertising and user-tracking services available on the Internet, it is assumed that these cookies represent more direct end-user service provision.

Table 4-3. Dynamic Web Application Frameworks Associated Cookies

Dynamic Web Technology	Cookie Name	Occurrence (Number of Sites)		First-Party (Number of Sites)		Sessional (Number of Sites)	
ASP	ASPSESSION%[14]	20875	21.3%	16917	81.0%	20874	99.9%
PHP	PHPSESSID	7379	7.5%	6028	87.1%	7052	95.6%
ASP.NET	ASP.NET_SessionID	5332	5.4%	3623	68.0%	5322	100%
JSP	JSESSIONID	5112	5.2%	3678	71.2%	5085	99.5%
Cold Fusion	CFID	1528	1.6%	1331	87.1%	1528	15.8%

All four dynamic technology cookies were found to be within the 99.9th percentile of reoccurring cookies amongst sites surveyed when ranked by cookie occurrence. In fact, ASP cookies where the most frequently retrieved cookies in this survey. PHP, ASP.NET and JSP cookies where ranked sixth, seventh, and eighth most occurring cookies. The only cookies more popular were all associated with Google Analytics[15] or ASP, making PHP, ASP.NET, and JSP the third, fourth, and fifth most widely adopted technology encountered in the survey. Cold Fusion rates where also quite high, ranking eleventh amongst the sites surveyed. Figure 4-6 presents the percentage of sites found using each of the identifiable web application frameworks. ASP was found to be the most common technology globally, with adoption rates of 21.3%, more than 13 points above the next closest competitor, PHP with 7.5%. ASP.NET and JSP were clustered together with adoption rates of 5.4% and 5.2%,

[15] Google Analytics was found to use a combination of four cookies, *__utma*, *__utmb*, *__utmc*, and *__utmz*.

respectively, and Cold Fusion was a distant fifth with a 1.6% adoption rate. These global rates will serve as a basis for comparison against which individual country adoption can be measured.

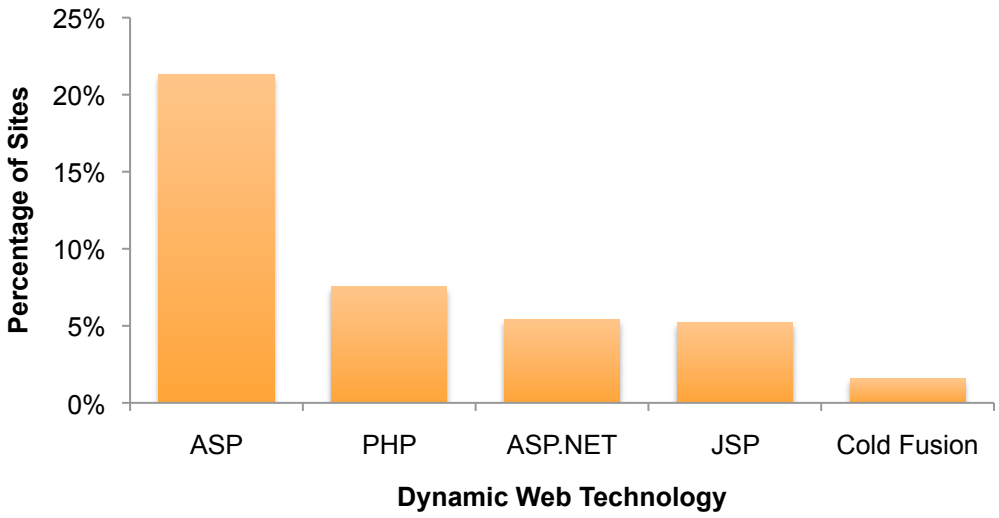


Figure 4-6. Global Dynamic Web Application Framework Adoption

A comparison of the adoption of dynamic web content frameworks amongst the 18 countries studied is provided in Figure 4-7. The figure presents the cumulative percentage of each of the four dynamic technologies. The cumulative percentages range from a low of 18.4% in Japan, to a high of 69.9% in Israel. Japan having the lowest cumulative occurrence per site is in-line with the results obtained presented in Figure 4-2, where Japan was the lower outlier with respect to sessional cookies. Although a low occurrence rate was uniform across all five technologies, Japan was found to possess the lowest occurrence rate for only ASP cookies, 7.0%, over 14 percentage points below the global average. This trend was also observed for the related ASP.NET technology for which Japan again had the lowest occurrence rate, 1.8%. Hong Kong, like Japan, had a lower than average occurrence rate across all five technologies. Despite this technology-wide depression, Russia (23.8%) was found to have a lower cumulative occurrence rate than Hong Kong (29.1%), attributed primarily to the lower-than average adoption of ASP

technology. Juxtaposed to Japan and Hong Kong, the UK presented higher than average occurrence rates for all five technologies; however, this did not translate into the UK having the highest cumulative rate amongst the countries studied, in fact six other countries posted higher rates—China, South Korea, Spain, Turkey, and Sweden. Apart for Japan, Hong Kong, and the UK, all other countries displayed mixed results, suggesting that certain technologies are more and less prevalent within specific countries.

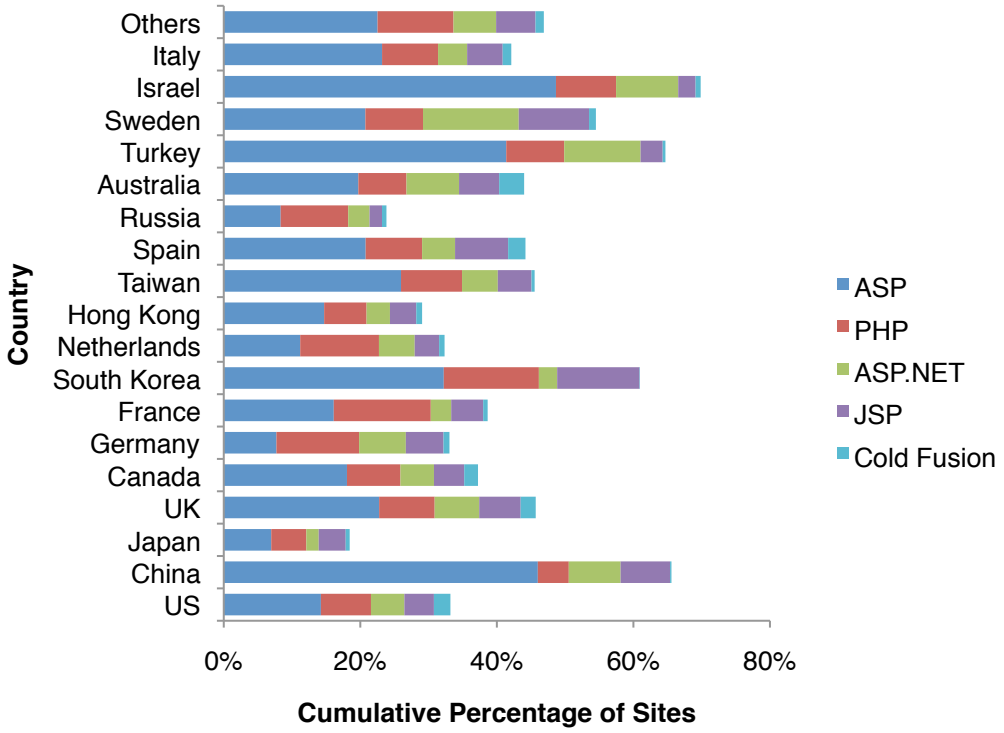


Figure 4-7. Cumulate Comparison Of Dynamic Web Framework Adoption Per Country

Israel had the highest cumulative occurrence rate, attributed to higher than average rates with respect to ASP, PHP, and ASP.NET technologies. On the other hand, the occurrence rates for JSP, and Cold Fusion cookies were below average. Although within Israel PHP and JSP were observed to be above average, ASP technology was observed to have the highest occurrence of any country—48.7%. This was over 27 percentage points above the global average representing the single largest

disparity encountered. The dominance of ASP within Israel is clear, as ASP accounts for more cookies than PHP, ASP.NET, JSP and Cold Fusion combined. Despite being the global leader in cookie occurrence, ASP had lower than average rates amongst 11 of the 18 countries analyzed; however, this only translated into a net result of 3 countries where ASP was not the dominant technology. The occurrence of this technology was most prevalent within Israel (48.7%), China (46.0%), and Turkey (41.4%), each of which posted rates more than 20 points above the global average (21.3%).

PHP, the second most encountered technology, had lower than average rates in the top three countries (ranked by number of sites surveyed), US, China, and Japan. Other than these three, PHP rates were only less than average in Hong Kong and Australia. The difference was most pronounced in China, followed closely by Japan and Hong Kong, the US appeared to be just slightly below average, less than half of one percent lower than the average. In fact, the US was observed to be below average with respect to ASP, PHP, ASP.NET, and JSP cookies, the only above average rate was with respect to Cold Fusion technology; however, it was a modest increase of less than one percentage point. The largest disparity within the US was observed amongst ASP cookies, where the US was more than 7 percentage points below the global average. All other technologies had disparities of less than 1 percent; it appears the usage of web application frameworks in the US, with the exception of ASP, is in harmony with global adoption rates.

Cold Fusion, the least occurring technology encountered by the survey, was found to be above average in only five of the countries studied: Australia, Spain, US, UK, and Canada (ordered by descending occurrence rate). Australia was observed to have the highest Cold Fusion cookie occurrence rate, more than two percentage points higher than the global average. Cold fusion usage was lowest in South Korea with an occurrence rate 0.1%. This represents the lowest occurrence rate for all dynamic web

technologies across all 18 countries. Despite having such a low occurrence rate of Cold Fusion cookies, South Korea had the highest rate of JSP cookies, 6.8 percentage points about the global average, and was the second highest with respect to PHP technology, falling closely behind France, the PHP frontrunner.

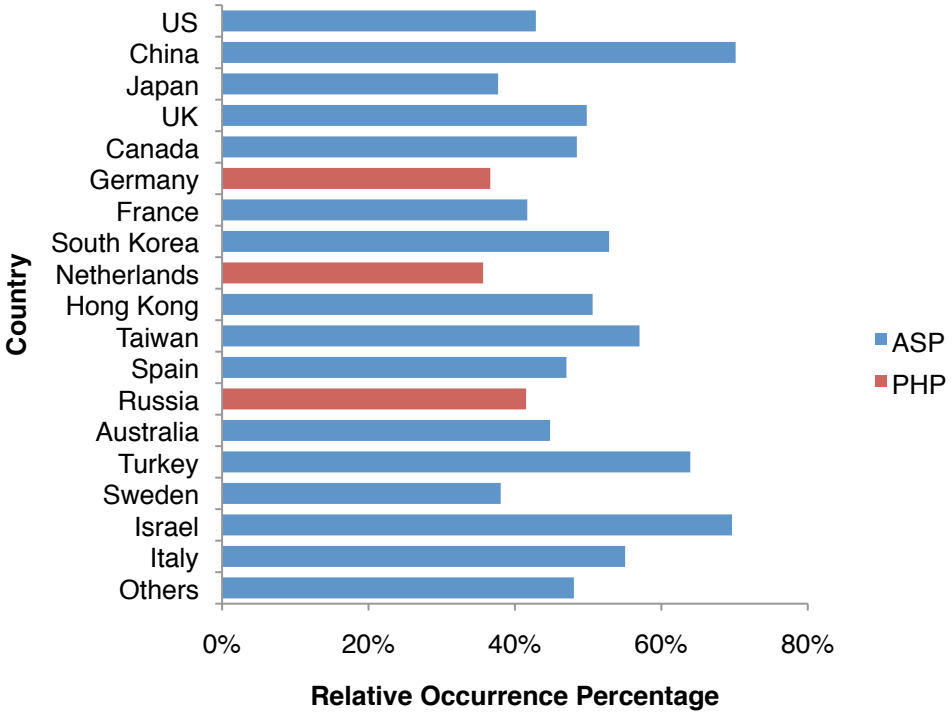


Figure 4-8. Relative Occurrence Of The Dominant Technological Platforms Per Country

To answer research question *Q4*, a breakdown of the dominant platform and relative cookie occurrence rate per country is provided in Figure 4-8. While ASP, PHP, JSP and Cold Fusion are all utilized globally, it is clear that the most dominant technology encountered was ASP. PHP, the only other technology most frequently occurring amongst the countries analyzed, was observed within 3 countries—Germany (36.6%), Netherlands (35.7%), and Russia (41.5%); two of which, Germany and Netherlands, have the lowest relative occurrence rates amongst the 18 countries. Only three countries have a relative occurrence rate outside of the range 35-60%, China (70.1%), Turkey (64.0%), and Israel (69.7%),

representing the least competitive markets with respect to dynamic web technologies.

4.3.2 Online Tracking & Advertising Technologies

Since the introduction of Cookies, online advertising providers have had a large stake in third-party cookie technology, specifically lobbying user-agents and the Internet Engineering Steering Group to include default support for third-party cookies (Kristol, 2001). Despite the lobbing, RFC 2109, and 2965 (Kristol & Montulli, 1997, 2000), the two latest cookie specifications, mandate that third-party cookies be rejected by default in any user-agent (web browser). This standard has however been generally ignored by all major user-agents—Internet Explorer, Netscape, Firefox, Opera, and Safari. Enabling third-party cookies by default allows Internet advertising networks to provided *targeted* advertisements to end-users. This *targeting* is only available due to the ability of advertisers to track Internet users across multiple sites and browsing sessions using third-party cookies. These third party cookies, often persistent in nature[16], are used extensively to track users across multiple sites and sessions, providing targeted advertisements based upon the browsing tendencies and history of end-users. Recently technologies, such as Google Analytics, have begun to use third-party embedded JavaScripts to provide tracking within a single site by setting first-party cookies. These technologies are beginning to blur the lines between first-party and third-party cookies, presenting a significant challenge for users to assess the status of their online privacy.

Third-party persistent cookies are the primary vehicle through which online advertisers have provided targeted advertisements to users across multiple sites and sessions. These advertisements are often overt; however, the user-tracking methods are not and are often concealed from the Internet user using *web bugging* or *web bugs* (Smith, 1999). This

[16] 71.8% of all third-party cookies encountered were persistent.

process involves placing links to third-party objects, often very small images with dimensions measured in single digit number pixels. These objects act to place cookies within the user-agent, without the consent of the user and are enabled by the default web-browser configurations of the majority of browsers. To study these online advertising and tracking practices, third-party persistent cookies were analyzed with respect to the repeat occurrence of similar hosts. A per-country list of the top 5 most occurring third-party persistent hosts has been compiled for each of the 18 countries selected in Section 4.2 and is presented in Table 4-4 and graphically in Figure 4-9. In the compilation of the dataset, two hosts were considered equal if the second-level domain (SLD) name matched. For example, *hitbox.com* the host responsible for the largest number of third-party persistent cookies (3969) was found to range across 503 distinct domain names all sharing the common SLD, *hitbox.com*.

Russia, as indicated in Section 4.2.2 was found to have the highest mean of sites that utilized third-party persistent cookies. This trend is clearly aspirant in Figure 4-9 as Russia records the first, second, and third largest percentages for third-party persistent hosts—*rambler.ru*, *list.ru*, and *yadro.ru,* respectively. In fact, the percentage of sites within Russia that have a cookie set by the largest occurring host *rambler.ru (48.2%)*, was larger than the cumulative percentages of the top five hosts from any other country analyzed. The two least occurring top five hosts within Russia, *spylog.com*, and *adriver.ru* were the fifth and sixth largest, falling short only of the French-specific host *xiti.com*. Third-party persistent cookie usage within Russia appears to be driven by Russian-specific advertisers, as all five were located within Russia. Although the hosts did set cookies for non-Russia sites, the majority of cookies, ranging from 73-90%, were set while visiting Russian sites.

Table 4-4. Top 5 Third-Party Persistent Cookie Hosts Per Country

Country	The Top 5 Third-Party Persistent Cookie Hosts	
	Host	**Percentage**
US	doubleclick.net	3.20%
	atdmt.com	2.79%
	207.net	2.68%
	hitbox.com	2.00%
	fastclick.net	1.95%
China	allyes.com	4.75%
	information.com	0.65%
	mozilla.org	0.65%
	revenue.net	0.60%
	admin88.com	0.56%
Japan	valuecommerce.com	3.18%
	tracer.jp	1.65%
	valueclick.ne.jp	1.11%
	207.net	1.01%
	doubleclick.net	0.92%
UK	doubleclick.net	7.70%
	atdmt.com	4.61%
	hitbox.com	2.98%
	estat.com	2.55%
	adtech.de	2.49%
Canada	doubleclick.net	3.76%
	hitbox.com	2.06%
	207.net	1.86%
	atdmt.com	1.82%
	statcounter.com	1.39%
Germany	ivwbox.de	9.08%
	doubleclick.net	4.11%
	falkag.de	3.69%
	adtech.de	2.32%
	www.etracker.de	1.87%
France	xiti.com	24.17%
	doubleclick.net	7.02%
	cybermonitor.com	4.90%
	estat.com	4.68%
	weborama.fr	4.08%
South Korea	acecounter.com	3.85%
	logger.co.kr	2.39%
	nasmedia.co.kr	1.86%
	overture.com	1.40%
	realmedia.co.kr	1.17%

Country	The Top 5 Third-Party Persistent Cookie Hosts	
	Host	**Percentage**
Netherlands	estat.com	7.22%
	doubleclick.net	5.50%
	sitestat.com	5.10%
	sexcounter.com	2.19%
	onestat.com	2.05%
Hong Kong	imrworldwide.com	0.93%
	casalemedia.com	0.71%
	zedo.com	0.71%
	doubleclick.net	0.64%
	statcounter.com	0.64%
Taiwan	hotrank.com.tw	4.16%
	cnyes.com	0.39%
	doubleclick.net	0.39%
	paypal.com	0.31%
	hitbox.com	0.23%
Spain	doubleclick.net	10.23%
	ojdinteractiva.com	8.67%
	imrworldwide.com	8.10%
	tradedoubler.com	2.45%
	atdmt.com	1.72%
Russia	rambler.ru	48.18%
	list.ru	38.15%
	yadro.ru	33.85%
	spylog.com	23.44%
	adriver.ru	17.58%
Australia	imrworldwide.com	8.22%
	doubleclick.net	7.18%
	adsfac.net	3.66%
	207.net	3.26%
	sextracker.com	1.96%
Turkey	reklam.gittigidiyor.com	3.91%
	statcounter.com	2.16%
	mediainer.net	2.16%
	iyi.net	1.62%
	hbmediapro.com	1.21%
Sweden	imrworldwide.com	12.30%
	doubleclick.net	7.15%
	tradedoubler.com	7.15%
	research-int.se	6.29%
	admeta.com	3.43%
Israel	imrworldwide.com	6.27%
	statcounter.com	3.43%
	walla.co.il	2.84%
	207.net	1.49%
	atdmt.com	1.49%

Country	The Top 5 Third-Party Persistent Cookie Hosts	
	Host	**Percentage**
Italy	imrworldwide.com	11.36%
	doubleclick.net	7.41%
	tradedoubler.com	3.79%
	shinystat.com	0.02%
	neodatagroup.com	0.02%

In strake contrast to the Russian results, Hong Kong was found to have the lowest percentage of sites setting third-party persistent cookies. This is reflected in the small percentage of sites for which third-party cookies are set. Hong Kong is the only country analyzed to have all five host recorded for less than 1% of the sites, suggesting that the use of third-party persistent cookies is spread across a wide variety of hosts, and no specific host has any substantial market share. Each of the top five hosts present in Hong Kong were found to be used in a wide variety of other countries, and no host was found to be Hong Kong specific. Taiwan, on the other hand, the county with the second lowest third-party persistent cookie rate, was found to use third-party persistent cookies at the same rate as Hong Kong. This similarity was confirmed by a Cliff's delta analysis performed upon the two populations with respect to third-party persistent cookies (δ = -0.007 [-0.033, 0.020]). Despite the similarity in third-party persistent cookie occurrence rates, Taiwan has a much different adoption rate than Hong Kong. Taiwan, like many of the other countries has a country-specific host that dominates the third-party persistent cookie landscape, *hotrank.com.tw*. This host is responsible for setting third-party persistent cookies on 4% of sites within Taiwan and is clearly the dominant host as the next closest host is only responsible for 0.4% of the cookies. Like the Russian-specific cookies, *hotrank.com.tw* is primarily used within Taiwan. The two populations, Hong Kong and Taiwan present two very different pictures of third-party persistent cookie usage, Hong Kong, a wide variety of globally established hosts, and Taiwan, a market with a clear country-specific market leader.

Like Taiwan, China, France, and Germany have a country-specific dominant host, *allyes.com*, *xiti.com*, and *ivwbox.de*, respectively. All four of the hosts more than double the next closest competitor with respect to site occurrence percentage. Other countries such as Japan, the UK, Canada, South Korea, Turkey, Sweden, Israel, and Italy have a clear front-runner, although their dominance is not a pronounced as that found in the other countries. The Netherlands, Spain, Australia appear to have a two- or three-way competition between the most occurring hosts. Finally, the US presents a much more competitive market with less than 1.5 percentage points separating the first and fifth most prevalent hosts.

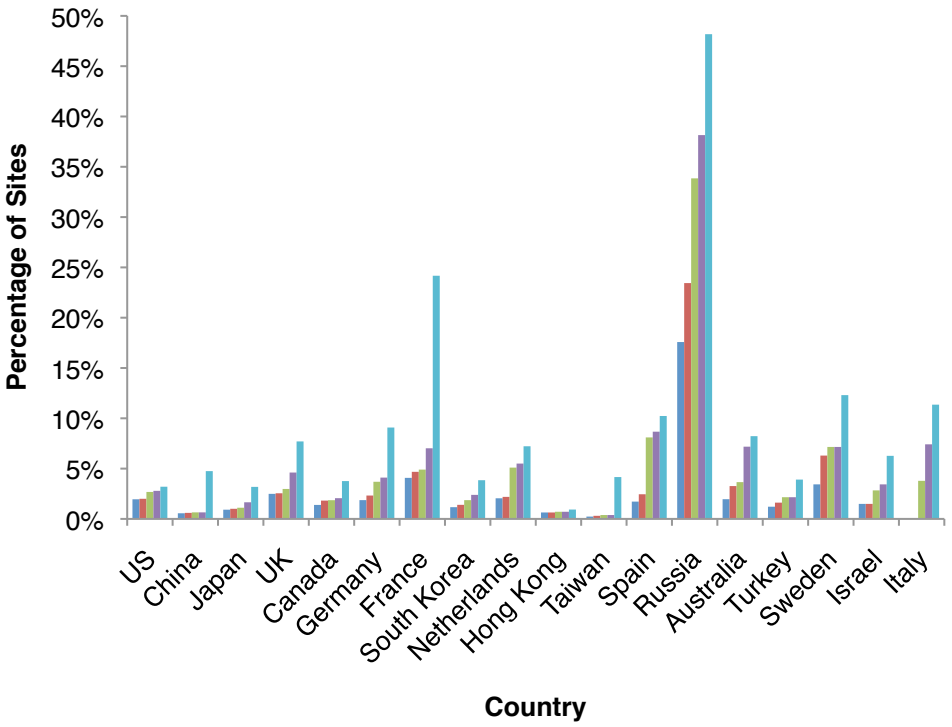


Figure 4-9. Top 5 Third-Party Persistent Hosts Per Country

Despite the number of country-specific hosts, cookies from *doubleclick.net*, were observed to be in the top five of 13 countries. These cookies prevalence was unmatched by any other host, and was set by 2,713 (2.8%) of sites encountered by the survey, the highest globally occurring third-party persistent host. Cookies set by the next closest competitor, *hitbox.com,* were encountered on less than half of the number of sites as

doubleclick.net, 1,221 (1.2%). Furthermore, *doubleclick.net* was found within the top five hosts for each of the G7 countries, in fact, it was the top technology for the US, UK, and Canada, and second behind the country-specific hosts in France and Germany. Japan was the only G7 country in which *doubleclick.net* was not one of the top two hosts.

Although third-party persistent cookies are deployed regardless of the country of origin, the usage of these cookies is not consistent between nations as a number of nation-specific cookies dominate locally. While third-party cookie usage was found to be universal, differences in the rates at which these cookies are deployed were observed. These findings answer research question Q5 providing insight into the tolerance of specific nations for this type of technological use. Clearly in Russia, third-party cookies are ubiquitous, whereas in Hong Kong, developers should generally stay away from these types of technologies, as they do not appear to have widespread acceptance.

4.3.3 Web Analytics: Third-Party Internal Site Tracking Technologies

Third-party persistent cookies and *web-bugs* are not the only techniques employed to track users and provide content tailored to specific Internet users. A growing number of sites are beginning to use third-party embedded JavaScript to set first-party cookies, which are used to collect data about users browsing habits providing the ability to produce user-centric content, assessing which pages are most profitable. Although these goals are similar to that of advertisers, the resulting cookies are difficult to identify, as they do not share a common third-party host.

An analysis, similar to that undertaken with respect to dynamic web technology was applied to the collection of over 200,000 cookies. The distinct cookie names where ranked by occurrence and then analyzed from the perspective of the number of distinct hosts that set the cookies. This analysis revealed several reoccurring cookie names, two of which were

definitively traced to third parties, namely Google Analytics (2007) and Omniture SiteCatalyst (2007). Google Analytics, traceable by the presence of four reoccurring cookies *__utmc*, *__utma*, *__utmb*, and *__utmz*, was set on 8.6% (8,411) of the sites surveyed, making it the second most widely found cookie-related technology encountered by the survey. Omniture SiteCatalyst, like Google Analytics, was traceable by the presence of two unique cookies *s_cc*, and *s_sq* and was set by 1.7% (1,690) sites surveyed, ranking Omniture seventh for most encountered cookie-related web technology.

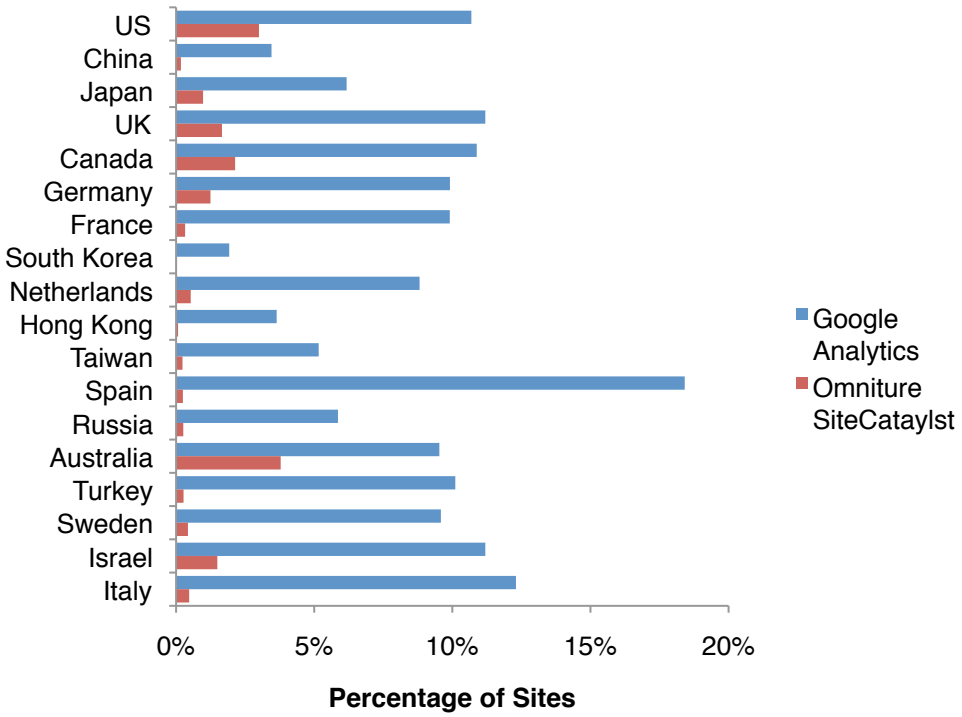


Figure 4-10. Web Analytics Technology Per Country

A per-country breakdown of Google Analytics and Omniture SiteCatalyst is provided in Figure 4-10 Google Analytics is clearly the global market leader, evident by a market share, at the lowest point, of 2.5 times that of Omniture. The market appears to be much more competitive in the Australia, Canada, Germany, Israel, Japan, UK, and US with a market share ranging from 2.5 – 8.0 times higher than that of Omniture, compared to 16.5 – 75 in the other countries. Google Analytics adoption

rate ranges from a minimum of 1.9% in South Korea to a high of 18.4% in Spain. Similarly to Google, Omniture's lowest rating was in South Korea, where the technology was not encountered. Australia posted Omniture's highest rating (3.8%) and was the country with the smallest ratio of sites with Google versus Omniture cookies.

In comparison to third-party persistent cookie based technologies presented in Table 4-4 and Figure 4-9, Google Analytics is the most used user-centric technology in all but five of the countries analyzed—China, France, South Korea, Russia, and Sweden. Within China, France, and Sweden, Google ranked second falling behind the country-specific hosts of France (*xiti.com*) and China (*allyes.com*) and *imrworldwide.com* in Sweden. In South Korea, where Google's rate was the lowest of the 18 countries studied, it ranked third behind two South Korean-specific hosts (*acecounter.com*, and *logger.co.kr*). Russia, as previously noted, was very different from the other countries, and Google Analytics ranked eighth amongst user-centric technologies encountered. All eight of the cookies that ranked higher than Google were Russian-specific, further evidence that the Russian online marketplace is driven primarily by internal interactions.

Hong Kong, despite the low adoption rate of third-party advertising cookies was found to have a Google Analytics occurrence rating of 3.6%, slightly higher than that of China. Google Analytics is the first third-party technology found used on more than 1% percent of sites from Hong Kong. Further investigation revealed a second Hong Kong-specific[17] re-occurring cookie, identifiable by the name *cdb_sid*, found occurring on 3.5% of sites. Information pertaining to the technology responsible for setting the *cdb_sid* was not readily available and a direct link to any one technology could not be established. Due to the re-occurring nature of this exclusively persistent cookie and the fact that the majority of cookies (88.9%) were first-party, it is assumed that these cookies fulfill a similar role to that of

[17] Found occurring in less than 0.6% in any other country.

Google Analytics and Omniture, however this cannot be explicitly confirmed. Further evidence suggesting that this cookie is Google's direct competitor within Hong Kong is the degree of overlap, or rather the lack there of, between sites setting these cookies. Within Hong Kong the presence of cookies from Google Analytics, Omniture, and *cdb_sid* where found to be mutually exclusive, suggesting that these cookies are dichotomous to each other. This exclusivity was also observed globally, where 98.7% of sites with these technologies were found to be mutually exclusive.

Third-party web analytics is quickly becoming an important tool within the business model of any e-commerce endeavour. Because cookies are a crucial component of these third-party software packages, the cookies are set as first-party, both sessional and persistent, to decrease the likelihood of rejection. This application raises a number of privacy questions, as these cookies are set by embedded third-party scripts; a detail that is not explicitly revealed to the user. While this study answers the research question *Q6—how are cookies used to track a users movement within a site*—they also elicit a number of further questions that are outside of the scope of this work. Similar to the privacy concerns of *unverified transactions* associated with the usage of third-party cookies (Kristol, 2001), this type of cookie usage can be seen as invasive and is currently implemented without any opt-out mechanism for the average user. Similar to the usage of third-party cookies, these issues have not crossed the tolerance threshold for the average Internet user, and as such remain in use globally.

4.4 Cookies: A Proxy for a Country's *E-Readiness?*

The annual *e-readiness* white paper published by the Economist Intelligence Unit "evaluates the technological, economic, political and social assets of 68 countries … and their cumulative impact on their respective information economies." (Economist Intelligence Unit & IBM

Institute for BusinessValue, 2006). The ranking is comprised of the weighted accumulation of scores from nearly 100 criterions spanning six distinct categories—connectivity, business environment, consumer and business adoption, legal and policy environment, social and cultural environment, and supporting e-services. Essentially, e-readiness provides a metric by which we can assess a country's e-business environment, and how fertile the environment is with respect to Internet-based endeavours. Of the 68 countries identified in the e-readiness report, 66 of these countries were the geographical origin of at least one surveyed site. 66 of 68 countries highlighted in the report represent 98% (96,365) of the sites surveyed and all 18 of the countries highlighted in Section 4.2 are ranked in the report. The e-readiness report was used to partition the survey results with respect to geographical origin of each site. These partitions are analyzed with respect to six criteria: cookie usage, first-party cookie usage, third-party cookie usage, sessional cookie usage, and persistent cookie usage, and third-party persistent usage.

To investigate the association between a country's e-readiness ranking and their cookie usage, a series of Spearman's ρ correlations were calculated and summarized in Table 4-5. Significant correlations were observed amongst all forms of cookie usage with the exception of first-party cookies. All of the correlation coefficients represented negative trends suggesting that the higher the e-readiness rank of the host country, the more likely the site to use cookies, especially third-party persistent cookies.

Table 4-5. Spearman's ρ Correlations For Cookie Usage vs. E-Readiness Ranking

	Cookies	First-Party Cookies	Third-Party Cookies	Sessional Cookies	Persistent Cookies	Third-Party Persistent Cookies
Spearman's ρ	-.309	-.208	-.462	-.254	-.468	-.499
Pearson's *r*	-.322	-.217	-.479	-.265	-.485	-.516
Sig. (2-tailed)	.012	.093	.000	.040	.000	.000
N[18]	66	66	66	66	66	66

From the ρ values in Table 4-5, we see that this correlation is strongest amongst third-party persistent cookies, and weakest for general cookie usage. A *small* relationship exists between overall cookie usage and ranking. The classification of overall cookie usage as *small* is attributed to the correlation of rank and third-party persistent cookies suppressed amongst the general population of cookies. Third-party and persistent cookies have *medium* correlations to ranking; the strongest relationship exists between third-party persistent cookies and e-readiness ranking. This *large* relationship between ranking and third-party persistent cookies suggests that the maturity of a country's e-business environment promotes the usage of third-party and persistent cookies, specifically third-party persistent cookies. These findings further support the analysis in Section 4.3.2; the majority of third-party persistent cookies are used within commercially viable environments to provide user-centric content. Based upon the correlations revealed in Table 4-5 it is clear that cookie usage, specifically third-party and persistent cookies, are related to the maturity of a nation's e-commerce environment, providing an affirmative answer to research question *Q7*—third-party persistent cookies directly correlate with the maturity of a country's e-commerce environment.

[18] N=66 because of the removal of the two countries that did not have any sites surveyed, and therefore could not be included in the Spearman's ρ correlations.

4.5 Summary of Results and Key Findings

Significant links were established by the survey with respect to the technological sources of first-party and sessional cookies. These types of cookies were observed in abundance amongst web application development platforms, specifically ASP, PHP, JSP, and ASP.NET. The clearest link was found with respect to sessional cookies comprising 99.0% of the technological-specific cookies, compared to only 28.7% of the non-specific cookies collected. This link was also observed amongst first-party cookies but not to the same degree—73.3% versus 49.0%. It is clear that there is a relationship between first-party sessional cookies and dynamic web technologies. Despite this relationship, technology-specific cookies only accounted for 37.2% of sessional cookies and 19.1% of first-party cookies, and although a clear link was found, it cannot be viewed as a definitive factor for this type of cookie usage.

The survey highlights the prevalence of vendor-specific third-party technologies both globally and within specific countries. Although the survey did find global leaders, such as *doubleclick.net* and Google Analytics, country-specific market providers such as *rambler.ru*, *allyes.com*, *xiti.com*, and *ivwbox.de* were also discovered. These providers suggest that despite a globally dominated market, there is room for country-specific competitors, as is the case within Russia, France, Germany, China, Hong Kong, and South Korea. This finding is of particular relevance to existing e-commerce ventures and those looking to break into the already saturated market.

A direct correlation was identified between third-party persistent cookie usage and the status of a country's e-business environment, as determined by the e-readiness rankings (Economist Intelligence Unit & IBM Institute for BusinessValue, 2006). This correlation, observed specifically amongst third-party persistent cookies, encapsulates the complexity of the relationship between cookie usage and country of origin.

Chapter 5

Testing Web Applications With Respect to Cookies[19,20]

The previous chapters have demonstrated that cookies are used by over two-thirds of Internet sites, and by over 80% of sites identified as providing dynamic content. With this level of adoption and the diverse user-agent implementations, testing web applications from the perspective of cookies becomes an increasingly important task. Furthermore, given the state-based nature of cookies and the reliance upon this technology within modern web programming frameworks, the verification of cookies within web applications is essential. This chapter outlines a novel cookie collection testing strategy. This strategy is based on the excess of 280,000 cookies studied in Chapters 3 and 4, and draws upon anti random testing principles to create an extendable cost-effective testing strategy for web applications.

The remainder of this chapter is organized as follows: Section 5.1 will outline the primary challenges associated with cookie testing; Section 5.2 will provide a number of testing recommendations specifically drawn from the data reported in Chapter 3; Section 5.3 will present the cookie collection testing strategy; Section 5.4 will outline the major challenges

[19] *A version of this chapter has been published. Tappenden, A. F., & Miller, J. (2009). Cookies: A Deployment Study and the Testing Implications. ACM Trans. Web, 3(3), 1–49.*

[20] *A version of this chapter has been published. Tappenden, A. F., & Miller, J. (2008). A Three-Tiered Testing Strategy for Cookies. Paper presented at the Software Testing, Verification, and Validation, 2008 IEEE International Conference on.*

overcome in relation to the automation of the testing framework, and Section 5.5 will provide a summary the chapter's key contributions.

5.1 Testing Cookies: Input-Space Explosion

The input-space for any modern software system is effectively infinite, and the primary goal of many testing strategies is to partition the infinite space into a finite number of sub-spaces or equivalence partitions. Each subspace should ideally test a different facet of the software without overlap. This partitioning is often done based upon the requirements (black-box testing), the underlying source code (white-box testing), or a hybrid of these two sources (gray-box testing). Despite the partitions methodology used, each partition should be represented by a single (or a few) test case(s). Testing cookies within a web application presents a similar challenge to that of traditional software systems: an effectively limitless input-space. This is exemplified by the cookie usage within the case studies presented in Chapter 3.3. Nebulous test cases can be fabricated by the selection of random combinations of cookies from within a cookie collection, but this is highly unlikely to be effective. The size of the input space for a collection of n cookies is given by

$$\sum_{k=0}^{n}\binom{n}{k} = C_0^n + C_1^n + \cdots + C_{n-1}^n + C_n^n = 2^n, \qquad \textbf{(5-1)}$$

assuming that each cookie is a binary variable (present or absent). The assumption of each cookie being a binary input is an oversimplification, and is useful only in demonstrating the input-space explosion. Cookies are not simple binary inputs, and this fact helps to reinforce the claim that the input-space with respect to cookies is prohibitively large. The input-space is further expanded when considering the sheer volume of subsequent pages involved within a web application. There are two ways to calculate the input-space with respect to cookie collections for a multi-paged application. The first method involves the assumption that the each individual request can be treated as an independent event; that all

previous requests and associated cookies collections remain constant. This assumption, reflecting the stateless nature of HTTP requests, results in an input-space, for an application with *p* pages, given by

$$p \cdot \sum_{k=0}^{n} \binom{n}{k} = 2^n p. \qquad \textbf{(5-2)}$$

Despite the stateless nature of HTTP requests, web applications use cookies to provide stateful sessions to users. This implementation of state across multiple HTTP requests suggests that the assumption of treating each request as independent, especially with respect to cookies, is unlikely to hold true. The value given by

$$\sum_{k=0}^{n \cdot p} \binom{np}{k} = 2^{np}, \qquad \textbf{(5-3)}$$

provides a more accurate representation of the input-space for any application that maintains state across multiple HTTP requests as it amalgamates the inputs for each individual request within a test case. It is clear from each of the equations that the input space for a web application is susceptible to exponential explosion; input-space reduction techniques are therefore mandatory for any cookie testing strategy.

To illustrate this input-space explosion, consider the eBay bidding scenario discussed in Chapter 3.3.3. The size of the input-space for cookie collection associated with the six-request usage scenario is over 16 million for a single request (5-1), over 100 million if each request is considered independent (5-2), and over 2×10^{43} if the sequence of requests are considered as a continuous input (5-3). These examples are trivial when examined from an application-wide perspective. As demonstrated in Chapter 3.2.3 and echoed within the eBay scenario, the values retrieved within the survey represent the cookies present only on the initial page; subsequent cookies and pages dramatically increase the size of the input-space. Of the 40 applications crawled in Chapter 3.2.3 and summarized in Table 3-2, the number of subsequent pages unearthed ranged between 26 and 1733 with a mean of 599 pages. The minimum collection of 5 cookies

encountered in Chapter 3.2.3 balloons to over 832 test cases using equation (5-2), and over $1x10^{39}$ with equation (5-3). This collection of cookies was associated with the 26-page website—the smallest in terms of pages encountered in Chapter 3.2.3. It must be stated that the assumption that all 26 pages actually represent differentiable units of dynamic code is very unlikely; however these crawls do represent distinct URLs most likely differentiated on the basis of the values passed into the system as part of a GET request. Values present within a GET request are appended to the URL of the request and therefore provide an explanation as to the number of distinct URLs crawled within Chapter 3.2.3. If all the URLs crawled were not differentiated on the basis of values passed via a GET request, then the URL must contain a request to a different page within the HTTP server. Although these values cannot be verified to represent distinct pages, they encapsulate the inclusion of the other variables present within the system, and illustrate the limitless extent of the input space for a web application with respect to inputs from all avenues: cookies, GET, and POST requests. In this case, it is pertinent then to examine the worst-case scenario discovered in Chapter 3.2.3. The single largest input-space encountered was *http://www.galveston.com*, with 16 cookies set and 710 URLs crawled, representing over $4x10^{7}$ possibilities via equation (5-2), and over $5x10^{3419}$ via equation (5-3). Clearly this input-space cannot be tested using brute-force methods; cookie-specific testing strategies are needed to adequately validate any application with multiple cookie deployment.

5.2 Cookie Testing Recommendations

As outlined in Chapter 2, cookies have received little attention from the academic community with respect to software testing, and are being actively exploited. As explored in Chapter 3.3.2, cookie exploitation is due to the number of cookie-related failures that permeate released software. Current conventions dictate that the cookie input space be divided into two partitions—present or absent—leading to the creation of systems that

can tolerate the rejection of all cookies. Although these are important partitions representing two of the most commonly encountered browser configurations, they are simply not sufficient to adequately validate the robustness required of a modern web application. This leads to the release of software that is riddled with cookie-related bugs and security vulnerabilities. Based on the current investigation, a number of testing partitions were elicited to form a basis for the equivalence partitioning of the cookie input space.

The first, most obvious partition is to divide cookies according to date of expiration. This partition seeks to separate sessional and persistent cookies on the basis of time. The partitioning can be done to varying degrees, either two partitions split upon sessional and persistent lines or multiple partitions based upon expiry date. This type of partitioning applied to the eBay usage scenario leads to the extraction of 9 partitions—a dramatic reduction from over 16 million. Utilizing this partitioning scheme, it quickly becomes obvious how advantageous it is, from a testing perspective, to have cookies that share the same expiration date. This trend is observed amongst the partitions expiring after 364 days, 1 year and 115 days, 1 year and 364 days, and 4 years and 363 days. Due to the likelihood of occurrence of these partitions, testing the partitions should be considered mandatory for any applications using a mixture of sessional and persistent cookies—58.3% of sites using cookies as outlined in Chapter 3.2.8.

Another useful partition is based upon third-party and first-party cookies. This partition seeks to simulate the rejection of cookies based upon the host associated with the cookie. By definition this partition is binary, and therefore provides granularity too limited for an effective testing strategy. Despite the restrictive granularity, this type of testing should also be considered mandatory for any given web application since browser configurations allow users to reject all third-party cookies. This type of testing is required for any application using a combination of first-

and third- party cookies—37.8% of sites utilizing cookies as described in Chapter 3.2.7. This testing is especially required for sites such as eBay, where third-party cookies are set from non-independent third-party hosts. Further granularity can be achieved by partitioning the cookies by distinct hosts, resulting in six partitions when applied to the eBay usage scenario. Other divisions can be elicited to provide further granularity. Cookies can be subsequently partitioned by expiration date, providing four further partitions, first-party sessional or persistent, and third-party sessional or persistent. The testing of the presence of these four partitions is of increased interest due to the continuing development of privacy protecting user-agents such as CookiePicker (Yue et al., 2007). With an increasing number of user-agents available that reject cookies on the basis of a perceived invasion of privacy, web applications can no longer assume that testing the simple dichotomy of first- versus third-party cookies is sufficient.

P3P user agents present an entirely new set of cookie partitions, based not on inherent cookie values, but rather upon the P3P policy of the host depositing the cookie. As exemplified in the eBay usage scenario, partitioning of cookies based upon P3P policy is not always binary. In the example, two distinct P3P policies were encountered, presenting the possibility of the inclusion and rejection of a specific set of cookies on the basis of P3P policy. For any site utilizing third-party cookies as part of the core business-model of the system, as third-party cookies often are (Tezinde, Murphy, Nguyen, & Jenkinson, 2001), partitioning the cookie input-space on the basis of distinct P3P policy is critical. The rejection of a subset of these cookies could have devastating financial repercussions.

As demonstrated by both the Fossil and eBay case studies, described in Chapters 3.3.1 and 3.3.3 respectively, cookies are not only set as a field within an HTTP response. Cookies can also be set through embedded JavaScript, allowing for the creation of increasingly complex user interfaces through AJAX technologies, but also allowing the theft of

cookies and user-sessions through cross-site scripting attacks. Cookies used and set within scripts must constitute another form of partition, further driving cookie-related test cases. Cookies set via embedded scripts must be set to non *HttpOnly*, an addendum to the cookie specification allowing cookies to be inaccessible from within a JavaScript environment (Microsoft Corp., 2002). These cookies are of particular vulnerability and are clearly destined for use within scripts embedded in the client-side application. Partitioning the cookies based upon JavaScript usage is required to ensure that the application functions correctly on clients with configurations that reject embedded scripts, those that do not support scripts, or those that do not support the *HttpOnly* addendum. Although the content of all cookies should be examined for the presence of sensitive information, the examination of non-*HttpOnly* cookies should be a top priority as these cookies are at the highest risk of theft.

Testing issues surrounding cookies are numerous, and the reduction of cookie input space is one of many issues that must be explored within the testing community. As attested by Chapter 3.3.2, malicious cookie input manipulation is a huge liability for any application that handles sensitive user information. Testing of cookies must be extended beyond the collection of cookies present within an application, and must address the values passed into the application through cookies. These issues remain largely unresolved, and testing strategies are needed address cookie input manipulation.

5.3 Cookie Collection Testing

As outlined in Section 5.1, an exhaustive test-suite derived from the cookie collection input-space is not feasible due to exponential explosion. Although this set is manageable for web applications that utilize a small collection of cookies, a significant number of web applications are found using upwards of 10 cookies, as discussed in Chapter 3.2.3. As web-applications use an increasing number of cookies, the selection of test case

reduction techniques that maintain testing effectiveness becomes an integral task. One potential approach is anti random test case generation, initially defined by Malaiya (1995). Anti random testing involves the selection of test cases to maximize the Cartesian or Hamming distance from all previous test cases and has been shown to be effective through a series of empirical evaluations (Malaiya, 1995; von Mayrhause et al., 1998; Yin et al., 1997).

The practice of anti random testing is extended to testing cookies by assigning a cookie to a single binary value within a test vector t_i, with 0 representing an absent cookie, and 1 representing the presence of a cookie. The two initial test vectors defined by anti random algorithms (Malaiya, 1995; von Mayrhause et al., 1998) pose two typical test cases, $t=<0,0,...,0>$ and $t=<1,1,...,1>$, representing a case where no cookies are present, and a case where all of the defined cookies are present. The two cases represent a complete rejection of cookies (browsers that are configured to reject cookies) and the total acceptance of cookies (assuming that all of the cookies are valid, accepted by the browser, and present to re-send to the server). These two test cases represent the current assumption regarding cookies; i.e. that they are either all present, or all rejected. To begin our discussion of anti random testing with respect to cookies, the question of random-seeding cannot be escaped, and is extremely pertinent. In this section, the testing recommendations will be incorporated into an anti random testing strategy as the seed for an anti random algorithm (Malaiya, 1995; von Mayrhause et al., 1998), providing an effective and efficient derivation of a robust testing suite.

The first set of test vectors is based upon a cookie's expiration date. These pairs, as mentioned previously, define the length of time for which a cookie is valid and should exist within the user-agent. With this value, a set of test vectors can be constructed representing the presence/absence of specific cookies in the collection based upon the cookie expiration. Assuming that t_i is a test vector in the test-suite $T=\{t_0,t_1,...,t_i,..,t_n\}$ and that

each bit of t_i maps directly to a single cookie within the collection, the selection of seeding test vectors based upon the expiry dates of a cookie collection is defined by the following algorithm:

1. Set the initial test vector to contain all ones, i.e. $t_0 = \langle 1, 1, \ldots, 1 \rangle$
2. Set the test-bit to zero for any sessional cookie. This vector will provide the basis (t_i) for the following steps.
3. Amongst the bits of t_i that are present (value equal to one), find the cookie(s) with earliest expiration date.
4. Set all of the bits in t_{i+1} to zero.
5. Set all of the bits in t_{i+1} whose cookie has an expiry date beyond that found in Step 3 to one.
6. Repeat Steps 3-6 for t_{i+1} until the t_i does not contain any ones.

The algorithm outlined above will create a set of test vectors that encapsulate the valid states of the cookie collection based solely upon the cookie expiration dates. These values can be extracted from the HTTP responses, a browser session, or the application source code. As an example, consider the test mapping of cookies extracted from the eBay bidding scenario presented in Table 5-1. The test-suite seed that exists after the above algorithm has been applied to this set of cookies is presented in Figure 5-1. Applying this reduction algorithm leads to the creation of 10 testing-seeds, each of which can be used to validate the application and to create a robust testing-suite using the anti random algorithm. This reduction, from over 16 million combinations down to 10 seeding vectors, allows for the development of a manageable and extendable testing-suite for cookies within web applications.

Third-party cookies comprise over 40% of the cookies used on the Internet. These cookies, as mentioned in Chapter 3.2.7, have been the subject of contentious debate between those advocating privacy concerns and those whose business model is adversely affected by third-party cookie rejection (Kristol, 2001). The result of the debate is the ability for users to configure browsers to reject third-party cookies. Given this quagmire and

the various domain-matching algorithms (Kristol, 2001), testing resources need to be allocated to these cookies to verify application robustness. From this perspective, testing-seeds should focus upon the rejection of third-party hosts, with an initial test vector reflecting the rejection of all third-party cookies, and a separate test vector for each host, reflecting host-specific rejection. An example of the seeding-vectors present within the test-suite, based upon the test mapping provided in Table 5-1 and third-party host is presented in Figure 5-2.

```
T = { <1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1>,
      <1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0>,
      <1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0>,
      <1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0>,
      <1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0> }
```

Figure 5-1. Expiration Testing-Seeds Generated From Table 5-1

```
T = { <0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,1,1,1,1,1,1>,
      <1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,0,0,0,1,0,0,0,0,0>,
      <0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0>,
      <0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0> }
```

Figure 5-2. Third-Party Host Testing-Seeds Generated From Table 5-1

```
T = { <1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,0,1,0,0,1,0,0,0,1,1,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0>,
      <0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>,
      <0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0> }
```

Figure 5-3. Compact P3P Policy Testing-Seeds Generated From Table 5-1

Table 5-1. eBay Bidding Scenario Cookie Testing Mapping

Test Bit	Cooke Name	Expiration	Host	Compact P3P[21]	HttpOnly
0	B	29 years 103 days	.yahoo.com	$P3P_1$	—
1	lucky9	4 years and 363 days	.ebay.com	No Policy	—
2	S	4 years and 363 days	.apmebf.com	$P3P_2$	—
3	RUA	1 year and 364 days	.main.ebayrtm.com	$P3P_3$	—
4	dp1	1 year and 364 days	.ebay.com	No Policy	—
5	npii	1 year and 364 days	.ebay.com	$P3P_4$	—
6	RUP	1 year and 364 days	.ebayrtm.com	$P3P_3$	—
7	mojo1	1 year 115 days	.mediaplex.com	$P3P_5$	—
8	svid	1 year 115 days	.mediaplex.com	$P3P_5$	—
9	cid	1 year	.ebay.com	No Policy	—
10	TC01	364 days	.main.ebayrtm.com	$P3P_3$	—
11	M01	364 days	.main.ebayrtm.com	$P3P_3$	—
12	A01	364 days	.main.ebayrtm.com	$P3P_3$	—
13	C01	364 days	.main.ebayrtm.com	$P3P_3$	—
14	PS	364 days	.main.ebayrtm.com	$P3P_3$	—
15	nonsession	364 days	.ebay.com	No Policy	HttpOnly
16	ns1	363 days	.ebay.com	No Policy	HttpOnly
17	CT	30 days	.ebayrtm.com	$P3P_3$	—
18	HT	Sessional	.main.ebayrtm.com	$P3P_3$	—
19	s	Sessional	.ebay.com	No Policy	HttpOnly
20	ebay	Sessional	.ebay.com	No Policy	—
21	ds1	Sessional	.ebay.com	No Policy	—
22	ds2	Sessional	.ebay.com	No Policy	—
23	secure_ticket	Sessional	.ebay.com	No Policy	—

Closely associated with third-party cookies, compact P3P policies provide a secondary layer of discretion for third-party cookies. This adds another layer of complexity to testing web applications. As observed in the

[21] Compact P3P Policy Mapping:

$P3P_1$ = CAO DSP COR CUR ADM DEV TAI PSA PSD IVAi IVDi CONi TELo OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN COM NAV INT DEM CNT STA POL HEA PRE GOV

$P3P_2$ = NOI DSP DEVo TAIo COR PSA OUR IND NAV

$P3P_3$ = CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR

$P3P_4$ = NOI CURa ADMa DEVa TAIa OUR BUS IND UNI COM NAV INT

$P3P_5$ = NOI DSP COR PSAo PSDo OUR IND UNI COM NAV

eBay usage scenario, applications are no longer contained within a single host, but can be scattered across multiple hosts. Within the scenario presented in Chapter 3.3.3, cookies were set with respect to six distinct hosts, five of which set third-party cookies and had an associated P3P policy (see Table 5-1). Similar to third-party hosts, seeding vectors should be devoted to P3P policies. Seeding vectors representing the rejection of any one specific P3P policy should be included, blanking all of the test-bits within a test vector associated with the policy. A special seeding-vector must be considered, representing the absence of all third-party cookies that do not have an associated P3P policy. This vector represents the default behavior of Internet Explorer, as it is set to reject third-party cookies without a compact P3P Policy. Figure 5-3 gives an example of the seeding-vectors present within the test-suite based on the test mapping provided in Table 5-1 and Compact P3P policy. Due to the very close relationship of the compact P3P policy and third-party cookies, reported in Chapter 3.2.10, a certain degree of overlap was observed among the P3P and third-party vectors. These three shared vectors stemmed from the inclusion of three web advertising components in the eBay application. These three hosts, among those discussed in Chapter 3.2.9, are known for providing user-centric advertisements, an essential component of the business model for many applications. Despite this overlap, both the third-party host and P3P seeding vectors were able to elicit dichotomous vectors, further enhancing the test-suite. One of the independent test vectors elicited by the P3P test case selection was the only vector found to explicitly test the *npii* cookie set by the host *rover.ebay.com* for the host *.ebay.com*, as discussed in the eBay usage scenario. This cookie, the only cookie set by *rover.ebay.com,* was found to have a different P3P policy from the other third-party cookies set by *.ebayrtm.com*. This difference in P3P policy suggests that the usage of this cookie is unique. Although P3P policy and third-party host seeding vectors produce overlapping results, it

is imperative to employ both strategies to validate any web application employing third-party cookies.

JavaScript and other embedded objects within web applications pose yet another hurdle in the verification and validation of a web application. JavaScript is used to provide end-users with a rich user-interface, greatly increasing all aspects of usability within many web applications. Although the benefits of JavaScript technology are plentiful, there are a large number of unwanted side effects associated with this technology, most notably cross site scripting (XSS) vulnerabilities (Cgisecurity.com, 2002; Cook, 2003). XSS vulnerabilities exist within an application primarily due a user's ability to post unsanitized custom content on or within a trusted page. These exploits are typically employed in an attempt to steal session cookies, like the JSESSIONID cookie discussed within Chapter 3.3.1.3 (Cgisecurity.com, 2002). To combat the security-compromising consequence resulting from the availability of cookies in the JavaScript environment, Microsoft introduced the *HttpOnly* addendum to the cookie specification (Microsoft Corp., 2002, 2007). This addendum, designed to block scripts from accessing cookies flagged with the *HttpOnly* marker, has found widespread adoption and is prevalent within the most common browsers used today. Like the other levels of discretion provided within web browsers, the *HttpOnly* addendum requires associated testing resources. The testing activities associated with *HttpOnly* cookies should focus primarily on the absence of these cookies from the collection, simulating the unavailability of these cookies to JavaScript components and the possible rejection of these cookies by any user-agent that does not accept the *HttpOnly* addendum. Depending on the fixture used for implementing testing within the application, the *HttpOnly* addendum becomes increasingly difficult to test because of the browser-specific implementations of the JavaScript environment. Due to these complications, it is recommended that at the very least the application be tested without the presence of *HttpOnly* cookies. Figure

5-4 presents an example of the seeding-vectors present in the test-suite based on the *HttpOnly* addendum and the test mapping given in Table 5-1.

```
T = { <0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0> }
```

Figure 5-4. HttpOnly Testing-Seed Generated From Table 5-1

In this section we have presented a number of test vector seeds that can be used as a basis for the creation of subsequent anti random test cases. These seeds should be amalgamated and used in combination to provide the basis for a robust cookie test-suite. The test-suite created from these seeds should be used to test each page within the web application. It is envisioned that this type of testing will be automated, using the pages present at the all cookies (<1,1,..., >) and no cookies (<0,0,....,0>) test vectors as oracles to evaluate the correctness of the application response.

5.4 Automated Cookie Collection Testing

Cookie collection testing, as described in Section 5.3, involves the testing of a web application from the perspective of modifying the collections of cookies stored within a user-agent. Given the ability of this testing strategy to define large pools of testing data for any application, this process requires a computer-assisted testing harness that is capable of providing both a framework for test-data generation and an automated test execution and evaluation environment. This section will provide a brief overview of the major challenges overcome through automation of the testing strategy. An in-depth description of the automated testing tool, CookieCruncher, is provided in Appendix B.

5.4.1 Test Case Definition

Within an automated endeavor, a definition of a test case that is accurate and precise is necessary. Test cases must accurately describe a set of pre-conditions and inputs that constitute the test. Accordingly, a test case must be able to reliably interact with the system; that is, apart from

triggering a fault, a test case should consistently execute the application to a known state.

Table 5-2. Test Case Definition

Test Case ID	
Pre-Conditions:	*A list of HTTP Requests and associated inputs required bring an application to the known testing state.*
Test Request:	*The HTTP Request and associated inputs to be evaluated.*
Cookie Collection:	*A cookie collection representing the cookies present for to be sent with the Test Request.*

In order to arrive at a definition of a test case within a cookie collection process, the underlying model of test input-space must be decided upon and explicitly stated. For the purpose of cookie collection testing, the model described in equation (5-2) has been selected as representative of cookie usage within a web application. Essentially, this model considers the cookie collection of each page within a website as an independent end-point for a test. A test case is therefore defined as a series of HTTP requests required to set the application state, the final HTTP request, and the collection of cookies present for the final request. Each of the HTTP requests in a test case includes the GET and/or POST request and any associated input data. This definition is summarize in Table 5-2 and exemplified in Table 5-3.

The example provided in Table 5-3 defines a test case in which a user browses to an application front-page, selects a link to log into the system, and inputs their credentials. The final request wherein the user provides their credentials is the request currently under test, and is the only request that is subjected to cookie modification. All other requests will be executed without modification. The output of the test case is the HTTP response of the application to the request with a modified cookie collection. It is assumed that the pre-conditions leading up to the final request are repeatable— an assumption that must be verified by the tester.

Table 5-3. An Example Test Case

System Login Test 0001		
Pre-Condition:	Request(s)	Input Data
	GET http://example.com	
	GET http://exmaple.com/login.html	
Test Request:	Request	Input Data
	POST http://example.com/main.html	username=test password=test
Cookie Collection:	loggedin = false userid=null	

5.4.2 Automated Test Oracles

An automated evaluation of each test case requires the definition of an automatable oracle that can be used to detect the presence of a fault within the underlying system under test. To fulfill this requirement, the results of two specific test-vectors were selected as the basis against which all results would be subsequently measured. A vector representing the natural state of the cookie collection for a given test request and a zeros vector were selected as test oracles. These vectors were selected as representative of two distinct and plausible cookie collections— two cookie collections that every web application should handle correctly. The vector representing the unmodified collection defines a base-case in which no cookie collection modification has occurred. The zeros vector, on the other hand, represents a direct contrast to the unmodified vector. The zeros vector presents the application with a browsing environment devoid of cookies; essentially this is equivalent to accessing the page from a browser with cookies disabled. In practice, these two vectors have been found quite effective at capturing the valid outputs related to cookie collection modification within real-world web applications.

5.4.3 The Tree, Context & Composite Similarity Coefficients

The selection of testing oracles is a fundamental component of the testing harness; of equal importance is the method by which testing results are compared against the oracles. Two similarity coefficients were selected as a basis for test result evaluation. The tree similarity and context similarity coefficients were first selected for HTML page difference detection by Yue, Xie and Wang (2007) and have been adopted for use within CookieCruncher. A brief discussion of each of the metrics will follow and the novel composite similarity coefficient will be presented.

5.4.3.1 The Tree Similarity Coefficient

Given the rise of the Document Object Model (DOM) as the basis from which HTML and XML documents are interpreted, and the nature of DOM as a hierarchical structure (Mozilla Developer Center, 2009; W3C, 2005), the *top-down tree edit distance* (Selkow, 1977) has been employed for the identification of structural differences between DOM documents (Reis, Golgher, Silva, & Laender, 2004; Yue et al., 2007; Zhai & Liu, 2005). A DOM tree is considered a *rooted labeled ordered tree*, because it has a single root node (Document), each node is defined to have a label, and the ordering of nodes is significant. The Tree Matching algorithm, presented in Figure 5-5, has been demonstrated to be effective for detecting syntactic differences between two programs (Yang, 1991), and has recently been applied to DOM trees by (Yue et al., 2007). The Tree Matching algorithm is the basis from which the tree similarity coefficient is calculated within CookieCruncher.

The Tree Matching algorithm provides a measure of the number of matching pairs within the *maximal matching* of two trees. For example, consider the two trees presented in Figure 5-6. In this example there is only one maximal matching containing 7 pairs: (N1, N15), (N2, N16), (N6, N18), (N7, N19), (N5, N17), (N11, N20), and (N12, N22).

Algorithm: Tree Matching (A,B)

1. **If** the roots of the two trees A and B contain distinct symbols **then return(0)**.
2. *m* := the number of first-level subtrees of A.
3. *n* := the number of first-level subtrees of B.
4. Initialization, $M[i,0] := 0$ for $i = 0, \ldots, m$.
   $M[0,j] := 0$ for $j = 0, \ldots, n$.
5. **for** *i* :=1 **to** *m* **do**
6. **for** *j* :=1 **to** *n* **do**
7. $M[i,j] := \max(M[i,j-1], M[i-1,j], M[i-1,j-1] + W[i,j])$
8. where $W[i,j] = Tree\ Matching(A_i, B_j)$
9. where A_i and B_j are the i^{th} and j^{th} first-level subtrees of *A* and *B*, respectively.
10. **od**
11. **od**
12. return $(M(m, n] + 1)$.

Figure 5-5. Pseudo Code for the Tree Matching Algorithm (Yang, 1991).

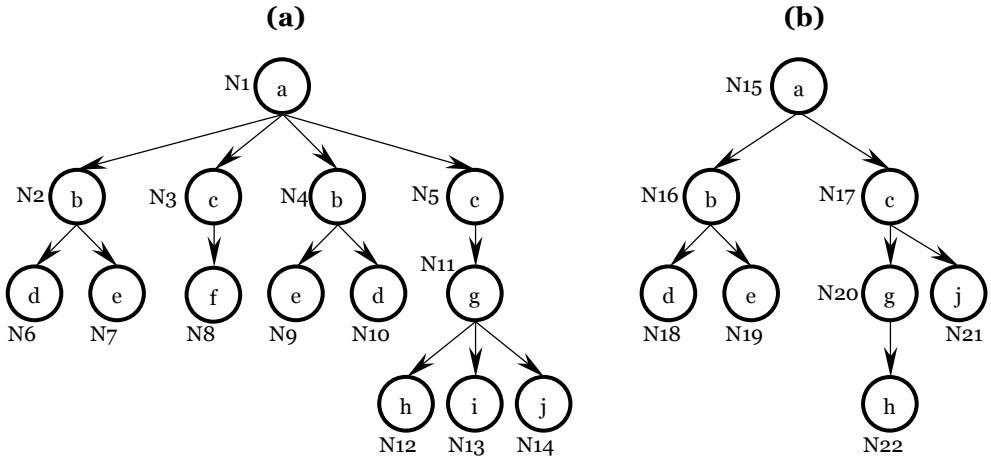


Figure 5-6. Tree Matching Example: Tree A (a) and Tree B (b) (Yang, 1991)

The maximal matching between two DOM trees can be a useful metric in determining the number of shared node pairs between two documents; however, this metric requires normalization to be useful as an oracle evaluation metric. Therefore, the Normalized DOM tree similarity coefficient was selected as the basis for all future comparisons (Yue et al., 2007). This normalized similarity metric is based upon the Jaccard similarity coefficient defined as

$$J(A,B) = \frac{(A \cap B)}{(A \cup B)}. \qquad (5\text{-}4)$$

The Jaccard similarity coefficient, or Jaccard index, measures the similarity and diversity between sets, and can be applied to measure the similarity between two DOM trees as

$$\textit{Normalized Tree Similarity } (A,\ B) \ = \ \frac{|TM(A,B)|}{\big||A| + |B| - TM(A,B)\big|}, \qquad (5\text{-}5)$$

where A and B are the two trees for which the metric is calculated, $TM(A,B)$ is the size of the maximal matching between A and B, and $|A|$ and $|B|$ are the number of nodes within A and B respectively.

5.4.3.2 The Context Similarity Coefficient

While the tree similarity coefficient provides a measure of the structural similarity between two DOM trees, the content of the DOM document, in terms of textual content present within the leaf-nodes, is independent of the tree similarity metric. Changes within the value of the leaf-nodes can dramatically alter the content of an output document and are not detectable by the tree similarity metric. In order to detect these changes a second similarity coefficient will be employed to specifically detect changes within the content of a DOM document.

The context similarity coefficient will be used as the basis from which the similarity of the content within two DOM documents will be assessed (Sachindra, Neeraj, Raghu, & Sumit, 2003; Yue et al., 2007). The context-similarity coefficient measures the similarity of DOM content within a context; that is, the metric is based upon the values stored within the textual leaf-nodes of a DOM tree and the path traversed through the tree to the leaf node. Although this metric contains a structural component—the traversed path—the metric is independent of the tree similarity. Changes within the textual leaf-nodes do not affect the tree similarity but will be reflected in the context similarity, and changes within the structural components of the DOM tree do not necessarily affect the context similarity but will be reflected in the tree similarity metric.

Algorithm: ContentExtract(A, context)

```
1.  Initialization, S := ∅; node := A.root.
2.  if node is a alpha-numeric text node then
3.     cText := context + SEPARATOR + node.value.
4.     S := S ∪ {cText}.
5.  elseif node is an element node then
6.     currentContext := context + SEPARATOR + node.name.
7.     n := the number of first-level subtrees of A.
8.     for j := 1 to n do
9.        S := S ∪ ContentExtract ( Ai, currentContext )
             where Ti is the i^th first-level subtree of A.
10.    od
11. endif
12. return (S)
```

Figure 5-7. Pseudo Code for the Content Extract Algorithm (Yue et al., 2007).

The content extraction algorithm (Yue et al., 2007) serves as the basis from which the context similarity metric is calculated, and is presented in Figure 5-7. For any tree A, the context extraction algorithm will extract the set S which contains all of the context-content paths within the document. A context-content path within the set S consists of the concatenation of the name of each of the nodes traversed, and the final textual content of the leaf-node. The context similarity of two DOM documents is defined as

$$\textit{Normalized Context Similarity } (S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}, \tag{5-6}$$

where S_1 and S_2 are the content-context sets extracted from each of the documents using the Content Extract algorithm. Like the tree similarity coefficient, the normalized context similarity is based upon the Jaccard Index. All subsequent discussions of context similarity are in reference this metric.

5.4.3.3 The Composite Similarity Coefficient

Between two DOM documents, the tree similarity metric provides a measure of the structural differences, and the context similarity metric a

measure of the content differences. Given the synergetic nature of the two metrics, the composite similarity coefficient is defined as

$$Normalized\ Composite\ Similarity\ (TSim,\ CSim) = \frac{\sqrt{TSim^2 + CSim^2}}{\sqrt{2}}, \qquad (5\text{-}7)$$

where *TSim* and *CSim* are the tree similarity and context similarity of two documents, respectively. This metric is derived from the Euclidian distance of a (tree similarity, context similarity) data-point from the origin, and provides an equally weighted assessment of the structural and content differences between two documents. The composite similarity coefficient is normalized and provides values in the range of [0 : 1], therefore allowing direct comparison between all three similarity metrics: tree, context and composite.

5.5 Summary

As clearly outlined in Section 5.1, input-space explosion is one of the primary challenges facing testing practitioners as they evaluate a web application. As a potential solution, an extension of anti random testing practices has been introduced. This solution outlined a number of cookie-specific attributes upon which seeding test vectors can be constructed. The definition of these vectors can be used as the basis for testing cookies within any web application, and can be used as inputs into an anti random testing algorithm to create an automatable and extendable test suite. This suite will contain the seeding vectors, ensuring the testing of the most common combinations of cookies, as well as a series of anti random generated test vectors existing to further validate the robustness of the application. This testing strategy represents a major shift in the status of novel web technologies within the literature; cookies are no longer a fringe web technology that can be ignored.

The framework outlined in Section 5.4 allows for the automation of the test execution and evaluation, two key components to any automated testing solution. Through the selection of the all present and none present

test vectors, web applications can be automatically evaluated against themselves, providing a low-cost testing framework for web application verification. Further description of the testing harness developed specifically for cookie collection testing is provided in Appendix B, including discussions of the integration of other state-of-the-art testing strategies within the testing framework.

Chapter 6

Evolutionary Adaptive Random Testing[22]

Software testing is a difficult task that involves balancing a number of competing factors such as resource utilization, time-to-market pressure, and the obligatory budget constraints. As such, getting the biggest *bang for your buck* is not always a straightforward proposition (Whittaker, 2000). A wide range of testing strategies have been established in the literature, each possessing strengths and weaknesses; one such strategy is random testing (Agrawal, 1978; Duran & Ntafos, 1984; Loo & Tsai, 1988; Ntafos, 1998; Schneck, 1979). Benefits of random testing include the low cost associated with the random generation of test cases, and the ease of automation associated with the technique. Recently, there have been a number of methods suggested to increase the effectiveness of random testing (T. Y. Chen, De Hao et al., 2007; T. Y. Chen, Kuo et al., 2007; T. Y. Chen et al., 2003; T. Y. Chen, Leung et al., 2004; T. Y. Chen & Merkel, 2006; T. Y. Chen, Merkel et al., 2004; T. Y. Chen et al., 2001). These methods, termed *Adaptive Random Testing* (ART) (T. Y. Chen, Leung et al., 2004; T. Y. Chen et al., 2001), are centered on the assumption that failure-triggering inputs occur within clusters in the input domain (Ammann & Knight, 1988; F. T. Chan et al., 1996; K. P. Chan, Chen, Kuo, & Towey, 2004; Finelli, 1991). Hence, ART methods seek to effectively

[22] *A version of this chapter has been published. Tappenden, A. F., & Miller, J. (2009). A Novel Evolutionary Approach for Adaptive Random Testing. IEEE Transactions on Reliability, 58(4), 619–633.*

detect faults by spreading the random test cases across the input domain, thereby increasing the likelihood of fault detection.

ART is not the only testing technique that attempts to evenly distribute test cases across the input domain; Anti Random testing explicitly seeks to select test cases that are as far away as possible from all previous cases (Malaiya, 1995; von Mayrhause et al., 1998). Anti random testing shares a number of similarities with ART. However, the distinct difference between the two is that anti random testing does not contain random elements. Currently, an effective, scalable implementation is not available. Furthermore, not only is the anti random testing method computationally expensive, but for any real-world testing scenario, including cookie collection testing, it is simply intractable.

Other investigations within the domain of adaptive random testing have focused upon the use of quasi-random sequences as drivers for the generation of test inputs (T. Y. Chen & Merkel, 2007; Chi & Jones, 2006). These investigations focus upon a specific group of mathematical sequences all characterized by the low-discrepancy property. These sequences deterministically produce a series of m-tuples that fill an m-dimensional hypercube more uniformly than pseudorandom numbers, as they maintain the low-discrepancy property. The use of quasi-random sequences within the testing community is a relatively recent contribution (T. Y. Chen & Merkel, 2007; Chi & Jones, 2006). These investigations have demonstrated the increased effectiveness of quasi-random sequences over pseudorandom or conventional random testing. The main benefit of quasi-random testing is the inexpensive computational cost associated with the generation of testing sequences. Limitations include the finite number of dimensions for which these algorithms are valid, and the limited number of distinct sequences that can be generated. While alleviating these limitations is an ongoing topic of research, it has not been demonstrated that quasi-random sequences are more effective than the current ART strategies (T. Y. Chen & Merkel, 2007; Chi & Jones, 2006).

A large gulf divides the computational overhead associated with the low-cost ART strategies, and the expensive, often intractable anti random algorithm (T. Y. Chen & Merkel, 2007; Chi & Jones, 2006). While anti random testing calculates the optimal test case having a maximum distance from all other tests, ART strategies employ low-cost techniques to generate testing inputs that are more evenly spaced than random testing. This chapter seeks to bridge the gap between anti random and adaptive random strategies, using an evolutionary search algorithm to find an approximation for the test case that has the maximum distance from all previous test cases. The novel application of an evolutionary algorithm to this problem dramatically lowers the cost (in terms of execution time) of finding the next anti random test case, and is within the same asymptotic runtime as the current ART methods. The increased effectiveness of the evolutionary random testing method is demonstrated when compared against both the premiere ART methods and quasi-random testing.

The remainder of this chapter is organized as follows. Section 6.1 provides a detailed explanation of the application of an evolutionary search algorithm to the simulation study testing problem; Section 6.2 outlines a simulation study undertaken to demonstrate the effectiveness of the evolutionary adaptive random testing; Section 6.3 provides a side-by-side comparison and analysis of the proposed testing methodology, and the current state-of-the-art for ART, quasi-random, and random testing techniques; Section 6.4 outlines the application of evolutionary adaptive random testing within the context of cookie collection testing as defined in Chapter 5, and Section 6.5 summarizes this chapter's key findings.

6.1 Evolutionary Adaptive Random Testing

6.1.1 Genetic Algorithms

The genetic algorithm is a well defined evolutionary search algorithm based upon the Darwinian process of natural selection (Holland, 1975). This approach, summarized by the pseudo code in Figure 6-1, involves the

```
Genetic Algorithm:

begin
t ← 0
initialize P(t)
evaluate P(t)
    while (not termination condition) do
        t ← t + 1
        select P(t) from P(t-1) according to evaluation
        crossover P(t) according to crossover rate
        mutate P(t) according to mutation rate
        evaluate P(t)
    end
end
```

Figure 6-1. Pseudo Code for the Genetic Algorithm.

initialization of a population *P*, followed by the selection of the most suitable individuals within *P* based upon a predefined *fitness function*. These individuals are then combined and mutated, creating successive *generations* of the population *P*. This process is repeated until the predefined *stoppage criterion* is met.

Within the genetic algorithm, a set of randomly generated individuals comprise the initial population, with each individual representing a possible solution. Each individual, known as a *chromosome*, can consist of a number of variables/parameters (*genes*). A ranking of the chromosomes is possible through the application of the fitness function, and a bias selection mechanism is used to determine which chromosomes will be used as *parents* in the creation of the *offspring* that will populate the subsequent generation. The creation of offspring is simulated through two processes: *crossover,* and *mutation*. The crossover operation selects the point at which the material from two parents will combine to create a new offspring. The operation is regulated by the probability of the crossover parameter to the algorithm. The mutation operation is defined for each gene within the offspring chromosome, and when triggered modifies the specific gene producing

new unique offspring. This process is regulated by the probability of the mutation parameter.

6.1.2 Genetic Algorithms & Software Testing

Genetic algorithms have been applied within a wide array of problem domains including software testing. The use of genetic algorithms for test input generation has been the focus of numerous approaches (Harman et al., 2004; Michael et al., 2001; Xiao et al., 2007). However, unlike ART, these approaches are based upon white-box code coverage metrics as an assessment of fitness. Other adaptations within the software testing domain include, but are not limited to, the prioritization of preexisting tests within a recursive test suite (Li et al., 2007), and the stress testing of distributed systems (Garousi, 2008; Garousi et al., 2008). Although the application of evolutionary algorithms to various problem domains is not unique, the application of this technique for the generation of ART input data is novel.

6.1.3 Evolutionary ART Definition

The definition of the genetic algorithm used throughout the simulation study will be provided in terms of problem encoding, fitness function, selection mechanism, crossover, mutation, stoppage criterion, and population size.

Encoding. The simulation study that will be presented in Section 6.2 approaches the testing problem of generating inputs for a two-dimensional unit hypercube, i.e. input tuples $\in [0,1]^2$. To encode this problem, each chromosome consists of two real-valued genes, each encoded as a double precision floating-point number valid in the range [0,1].

Fitness. To provide the best coverage of testing inputs across the input domain, a Euclidian distance-based fitness function is defined as

$$Fitness(x,T) = \min_{y \in T}\left(\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}\right). \quad \textbf{(6-1)}$$

The value *Fitness(x,T)* is then assigned to each chromosome, and is used to select individuals for reproduction.

Selection Mechanism. The selection of parents for the creation of each individual within the subsequent generation was performed using the k-Deterministic Tournament selection algorithm. This method selects *k*-random chromosomes from the general population, and chooses the best (the chromosome with the largest *Fitness(x,T)*) from the pool of *k* chromosomes. Within this study, the size of the tournament pool *k* was defined as 2.

Crossover. The crossover process serves to create the offspring from two parents. Single-point crossover was applied to the two chromosomes with a probability of crossover set at 0.6.

Mutation. Mutations could occur to any gene within an offspring chromosome. Because each of the genes within the chromosome is a real-valued double precision floating-point number, a subsequent parameter ε is defined to set the size of all mutations. Within this study, the probability of mutation is defined as 0.1, and the size of the mutation, ε, was set at 0.01.

Stoppage Criterion. A number of different stoppage criteria exist for genetic algorithms, and a criterion is often chosen based specifically upon the degree of complexity of the search problem. Within this study, a stoppage criterion of 100 generations was selected. This stoppage criterion was selected due to its uniformity amongst all trial runs of the search algorithm, simplifying the runtime evaluation. Other possible criteria include a successive number of generations without improvement, a target fitness goal, or an overall number of individual evaluations.

Population Size. Within this study, a constant population size of 20 individuals was used. The population size parameter was held constant to avoid the over-optimization of the parameters to the specific problem.

6.1.4 Evolutionary ART Runtime

The primary overhead associated with any genetic algorithm lies within the evaluation function. As shown in the pseudo code provided in Figure 5-5, a basic genetic algorithm has an asymptotic run time proportional to the number of executions of the loop, *G*. This runtime, with respect to the stoppage criterion outlined previously, is a constant 100 loop executions. The 2-Deterministic Tournament algorithm implemented within this study requires a constant number of executions, which is scaled linearly based upon the population size $|P|$, defined as 20 individuals. The crossover and mutation operations require a constant number of executions, and can be ignored with respect to the algorithm's asymptotic runtime.

The heaviest computational unit within the algorithm is the fitness function, as defined by (6-1). The fitness function increases linearly with *T*, the number of existing test cases supplied as an input to the algorithm. The variable T was the only value within this study which was not held constant. The resulting asymptotic runtime is $eAR(|P|,G,T) \in O(|P|G \cdot T)$. This can be further simplified to $eAR(T) \in O(T)$, recognizing that $|P|$ and *G* are constants, and as $T >> |P|G$, the values of $|P|$ and *G* become insignificant. This runtime is associated with the generation of a single test case. Similar to the repeated iterations of the FSCS algorithm, the runtime associated with the generation of a sequence of *n* test cases for fixed $|P|$ and *G* values is $eAR(n,|P|,G) \in O(|P|G \cdot n(n+1)/2)$. Because $|P|$ and *G* are constant, and become insignificant as *n* increases, the runtime can be simplified to the order of quadratic time complexity, $eAR(n) \in O(n^2)$. Both the FSCS and eAR methods can be implemented within the order of n^2 time, and RRT in the order of $n^2 \cdot log(n)$ time. Hence, all

approaches are computationally feasible in accordance with Cobham's thesis; a computational problem can be feasibly computed if it is within the order of polynomial time (Cobham, 1964).

6.2 Simulation Study

This section describes the study conducted to analyze evolutionary Adaptive Random Testing (eAR) as an alternative to FSCS, RRT, and the Sobol sequence to generate random test data for a generic software testing problem.

6.2.1 Research Questions

The following research questions motivated this study:

- ***Q1.*** *Is the application of search-based algorithms appropriate within this domain?*
- ***Q2.*** *Which algorithm generates random testing data that most effectively reveals software defects?*
- ***Q3.*** *What is the user perceived latency of the algorithms?*

These questions directly relate to the selection of a testing algorithm for software testing practitioners, and should be carefully considered when selecting a random testing strategy.

6.2.2 Experimental Design

To analyze the effectiveness of each approach, twenty-five unique testing sequences were generated for each testing algorithm. Due to the random nature of the non-deterministic testing methods analyzed, a single run would not provide an adequate sample from which to draw conclusions. Thus, twenty-five 10,000 test case sequences were generated for the eAR, FSCS, RRT, and RT algorithms. Due to the deterministic nature of the Sobol Sequence, the first 10,000 members of the Sobol Sequence were used within the study. The testing sequences were each generated from an input domain consisting of a two-dimensional unit hypercube, i.e. input tuples $\in [0,1]^2$. This simulation problem was selected due to its recurring

use within the field of ART, therefore providing the ability for comparison against past, and future research within this topic (K. P. Chan et al., 2004; T. Y. Chen, De Hao et al., 2007; T. Y. Chen et al., 2003; T. Y. Chen, Leung et al., 2004; T. Y. Chen & Merkel, 2006, 2007; T. Y. Chen, Merkel et al., 2004; T. Y. Chen et al., 2001; Mayer & Schneckenburger, 2006).

Associated with each of the ART methods are a number of algorithm-specific parameters that were held constant for all simulations. The parameter values were selected based upon the recommended values found in the respective works, and the specified values in Section 6.1.3. The value k for the FSCS algorithm defines the size of the pool from which randomly generated test cases were selected. For all simulations in this study, this value was held constant at k=10, in accordance with the recommendations of Chen et al. (T. Y. Chen, Leung et al., 2004). Similarly, for the RRT algorithm, the value of R determines the relative size of the restriction zone placed around any existing test case within the testing sequence; this value was held constant at R=1.5 as recommended in Chan et al. (K. P. Chan et al., 2004).

The quasi-random Sobol sequence was generated by the GNU Scientific Library (Free Software Foundation, 2008), and was based on the works of Antonov & Saleev (Antonov & Saleev, 1980), and Bratley & Fox & Niederreiter (Bratley & Fox, 1988; Bratley, Fox, & Niederreiter, 1994; Fox, 1986). The deterministic process of generating input data based on the Sobol Sequence was unique to quasi-random testing, as only one set of test inputs is generated by the Sobol sequence, whereas 25 unique sequences were generated for each of the other testing strategies. The parameters associated with eAR are outlined in Section 6.1.3, and are not repeated here for the sake of brevity. The parameters remained constant, and were not altered in order to reduce the likelihood of problem-specific over-optimization.

6.2.3 Effectiveness Measure

The selection of an adequate effectiveness measure was paramount to the validity of this work. Several metrics were considered; however, the *f-measure* proposed by Chen et al. (T. Y. Chen et al., 2006) was chosen due to its widespread use within the field (K. Chan et al., 2002; K. P. Chan et al., 2004; K. P. Chan, Chen, & Towey, 2006; T. Y. Chen, De Hao et al., 2007; T. Y. Chen et al., 2006; T. Y. Chen, Kuo et al., 2007; T. Y. Chen et al., 2003; T. Y. Chen, Leung et al., 2004; T. Y. Chen & Merkel, 2006, 2007; T. Y. Chen, Merkel et al., 2004; T. Y. Chen et al., 2001; Loo & Tsai, 1988; Mayer & Schneckenburger, 2006), and specific applicability to software testing. The f-measure is calculated as the number of generated test cases required to detect a randomly generated fault-causing region within the input domain, i.e. the number of test cases generated before a test case located within an error region is generated. The metric has been applied with respect to three distinct failure-causing patterns: Block, Strip, and Point. All three patters are presented graphically in Figure 6-2. Due to the inherent random nature of the metric, multiple runs of the metric were required to obtain accurate results for each of the populations. Furthermore, the failure rate θ associated with each f-measure provides the ability to analyze the testing strategy across varying failure occurrence rates. Failure rates of 0.01, 0.005, 0.002, and 0.001 were used to analyze the effectiveness of the testing methods. For each of the non-deterministic test sequences (from Section 6.2.2), 1000 f-measure statistics were calculated for each fault-pattern, and failure rate. This resulted in a population of 300,000 f-measures for each testing method, yielding a total population of 1,500,000 samples. Because of the deterministic nature of the Sobol sequence, only one testing sequence could be generated. To provide equal population sizes from which to draw statistical conclusions, 25,000 f-measure statistics were calculated with respect to the quasi-random testing sequence for each failure pattern and failure rate,

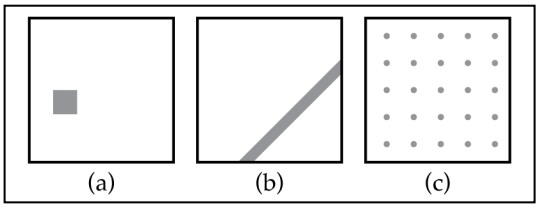


Figure 6-2. The Block (a), Strip (b), and Point (c) Failure Patterns Associated With The F-Measure (T. Y. Chen, Kuo, & Merkel, 2006).

producing an equal population size to that of the other testing methods studied.

Clearly, the selection of the f-measure metric allows for the generation of large pools of simulation data; however, that alone does not suggest that the metric is the appropriate method to use within the context at hand. One of the principle benefits of the f-measure metric is the direct correlation between it and the real-world testing effectiveness measure, namely cost. The cost associated with any testing strategy is twofold: the initial cost of creating the test suite, and the cost associated with the implementation and execution of each subsequent test within the suite. For the random testing methods, the upfront costs are very low, as the test suite is generated automatically. Therefore, when analyzing testing strategies within this domain, the cost associated with the number of tests required to trigger a fault is paramount. As it is this cost that the f-measure effectively simulates, it provides a clear indication of the effectiveness of a random testing strategy.

Although the implementation of the three failure patterns was generally straightforward, a number of interesting situations surfaced during the process. The block pattern was implemented by the selection of a random point within the unit hypercube around which a square was constructed. In the case where a square could not be constructed due to the selection of a point close to one or more boundaries, the selection process was repeated until a square could be constructed that existed completely within the border of the hypercube. The strip pattern was implemented as specified by Chen et al. (T. Y. Chen et al., 2006), whereby

two points are selected: one on a horizontal boundary, and the other on the vertical boundary. These points together create a defined region centered about the line between the two points. The point pattern was implemented by the selection of 10 random points from within the hypercube; around each point, a circular region was defined so that the sum of the constructed regions was equal to the simulation failure rate. Similar to the block pattern, the region was specified to lie completely within the boundaries of the unit hypercube. Any point whose region was not completely within the bounds of the hypercube was rejected, and reselected. Because of the restrictions placed upon the failure patterns, the area of the input domain around the boundaries of the hypercube was less likely to have any error region for both the block, and point patterns. The strip pattern as implemented cannot simulate a horizontal or vertical strip, or a strip that emanates and terminates from the two horizontal or vertical boundaries. Despite these limitations, the implementation of the failure patterns is in-line with those used in the literature, providing the ability to clearly evaluate the study results within the context of current ART research (K. P. Chan, Chen, & Towey, 2006; T. Y. Chen et al., 2006; T. Y. Chen, Leung et al., 2004; T. Y. Chen & Merkel, 2007; Mayer & Schneckenburger, 2006).

6.2.4 User-Perceived Latency Estimation

ART techniques are based on two assumptions: that faults are located within clusters within the input domain, and that idle computational resources are available. Essentially, ART methods trade off computation overhead in return for a higher quality random test input. The crucial question for any testing practitioner with choosing a RT strategy is: which method can produce the most effective test inputs without incurring considerable overhead in the testing process? Given the increasing processing power of modern computers, this question is often irrelevant as the computer-human interactions are commonly the bottleneck. For any

user-interactive testing scenario, it is unlikely that any of the ART methods will contain any human-perceivable overhead.

Given the costs associated with user-interactive testing, test automation has become an attractive alternative for many software endeavors. Automated software testing is uniquely suited to utilize ART methods. Like the test generation process, automated testing can occur without human interaction. In these cases, it is the automated test process that may be waiting for the generation of the next test input. In such a scenario, assuming that the execution of a test is concurrent with the generation of the next test input, the deciding factor in which testing strategy to employ is dependent on the execution time of the test, and the size of the desired test suite. For example, consider the system-level testing of any reasonably sized software endeavor; it is assumed that the cost of execution would greatly outweigh the computational overhead required for ART input generation. However, an empirical runtime analysis will be undertaken to confirm this assumption.

To test this conjecture, each of the ART methods were implemented as an integrated program, written in C. The program was further instrumented to measure the overhead cost (wall clock time) associated with the generation of the n^{th} test case, when provided the previous $n-1$ test cases. One hundred runtime data points were generated in successive iterations of each of the four algorithms for the range [1,000:100,000] with intervals of 1000. To minimize the possibility of noise due to interference from background processes, the benchmark was executed 100 times, resulting in a pool of 100,000 data points from which the evaluation of the overhead costs of each algorithm was conducted.

6.2.5 Analysis Tools

Due to the large pools of data generated for each testing method, SPSS was used to analyze the experimental data, and generate box plots. The ANOVA (ANalysis Of VAriance) package was used to analyze the

differences between the testing methods, and to provide a statistical basis from which to draw conclusions of testing effectiveness. The associated null hypothesis was that the means of the f-measure for each of the four methods are equal. The null hypothesis was rejected if the significance of the result was below the standard Type 1 error rate, 0.05.

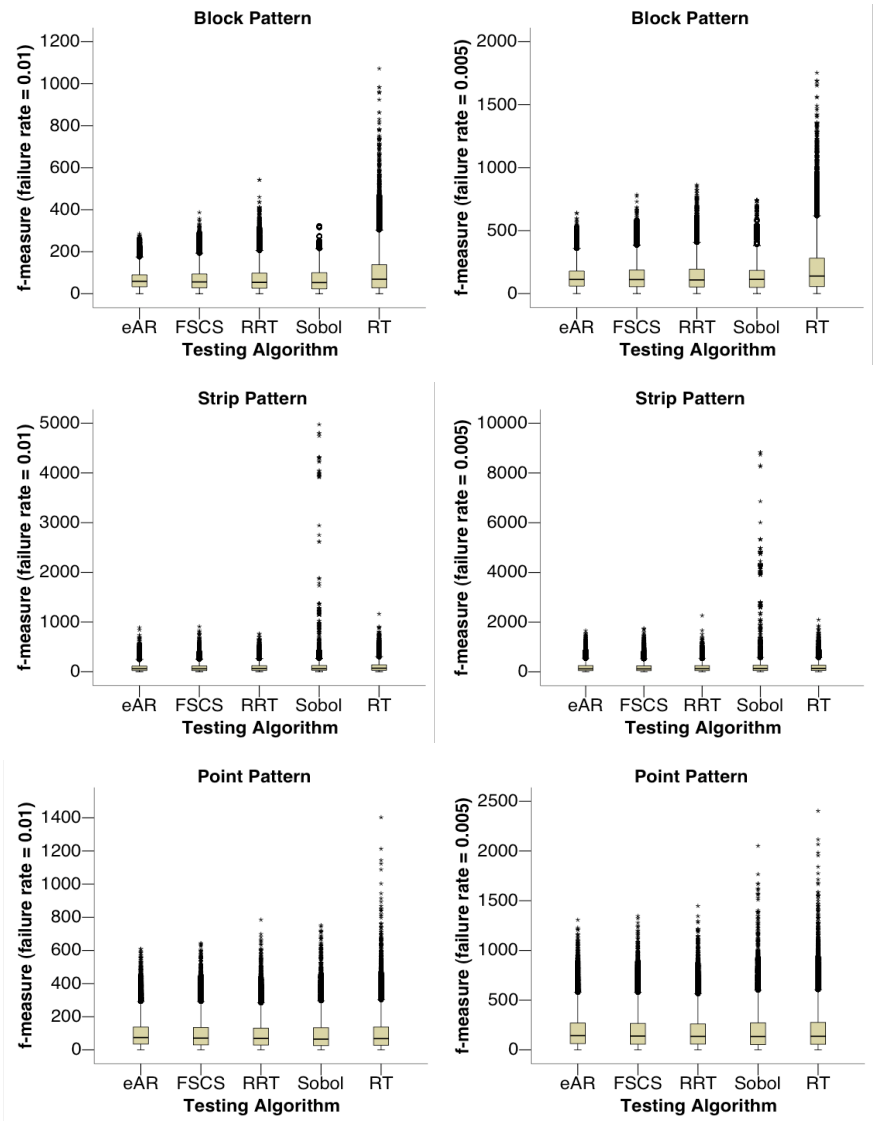
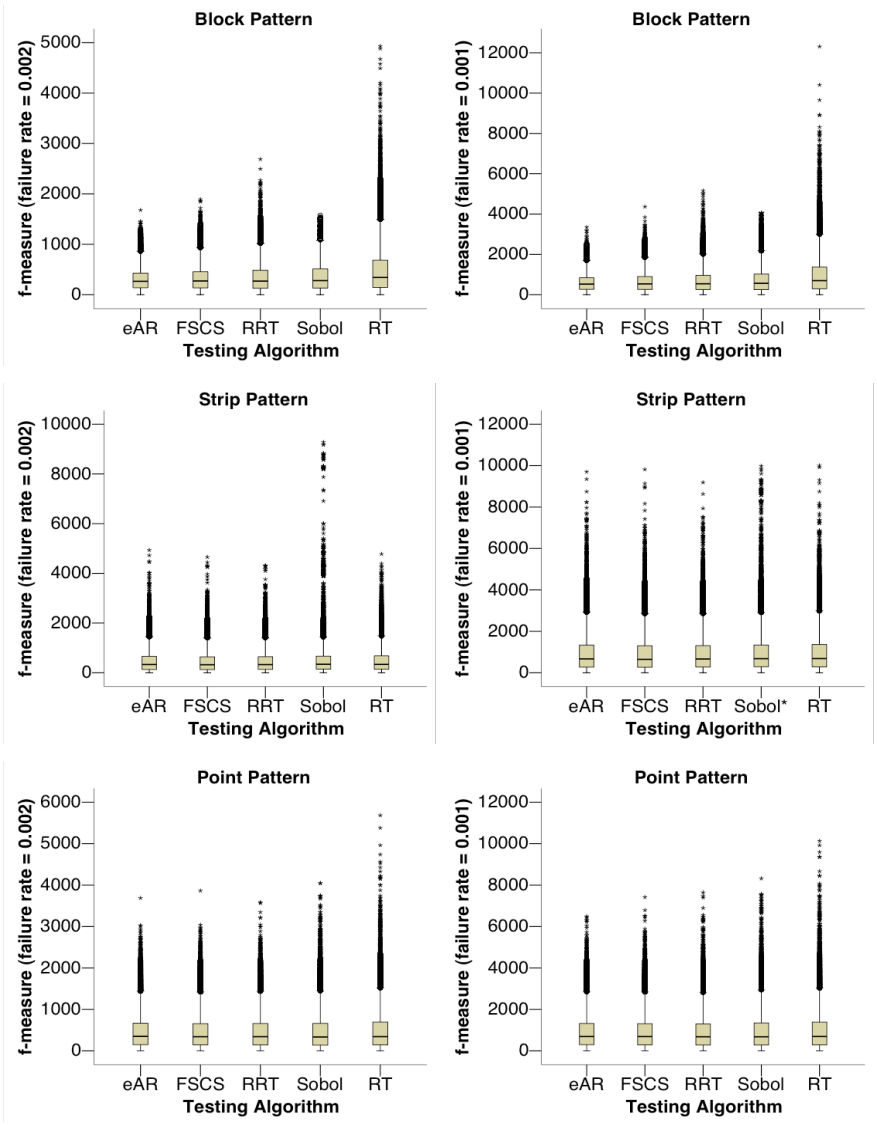


Figure 6-3. Box Plots Of Simulation Results For Block, Strip And Point Patterns With Failure Rates Of 0.01 And 0.005 For Each Of The Testing Algorithms



* *There were 11 strip failure patterns that could not be detected by the 10,000 inputs generated by the Sobol Series for failure rate $\theta = 0.001$.*

Figure 6-4. Box Plots Of Simulation Results For Block, Strip And Point Patterns With Failure Rates Of 0.002, And 0.001 For Each Of The Testing Algorithms.

6.3 Experimental Results, and Discussion

6.3.1 Worst-Case Effectiveness

The box plots presented in Figures 6-3 to 6-4 provide a graphical representation of the results of the study. Initial conclusions can be drawn from a straightforward graphical analysis of the plots. Scanning horizontally across the figures, it is clear from the shape of the plots and

the increasing scale on the vertical axis that the varying failure rate has a scaling effect on the observed f-measure. This scaling factor, while not linear, did not drastically affect the distribution of results for each of the testing methods. Scanning horizontally across the figure, a dramatic change in spread, and worst-case outliers in the f-measure statistic is apparent for the ART methods, and Sobol sequence. Conversely, RT appears unaffected by the alteration in the underlying fault pattern. A difference is observed between the strip failure pattern, and the other methods with respect to quasi-random testing, although these differences between the testing methods appear to converge as the failure rate decreases.

In terms of worst-case testing effectiveness, eAR, FSCS, RRT, and Sobol all outperform RT with respect to the block pattern simulations. Of the three ART techniques, eAR exhibited the best worst-case f-measure for the block pattern simulations. Except for a failure rate of 0.002, eAR exhibited the best worst-case f-measure of any of the techniques studied. At a failure rate of 0.002, the only other testing method with a lower worst-case f-measure was the Sobol sequence. Furthermore, the values of the 75^{th}, and 95^{th} percentiles for eAR were lower than those for the FSCS, RRT, Sobol, and RT in all block pattern simulations.

With respect to the strip pattern, the Sobol sequence exhibited the poorest worst-case testing performance. Regardless of simulation failure rate, the Sobol sequence consistently reported the highest f-measure values, and was the only testing method that was unable to detect a fault pattern with a test sequence of 10,000 test inputs. These undetected faults were present only for the strip pattern simulations with a failure rate of 0.001, and could not be presented in the box plot in Figure 6-3. Apart from the Sobol sequence, eAR, RRT, and RT each performed poorly at failure rates of 0.002, 0.005, and 0.001 respectively. While the worst-case for FSCS was never the global maximum, the worst-case fault detection was poorer than the other ART methods for failure rates of 0.01, and

0.001. In terms of worst-case performance amongst the point pattern simulations, eAR outperformed the others testing methods in three of the four simulations, and was a close second to RRT in the fourth. Similar to the results observed amongst the block patterns, the effectiveness of the ART methods in terms of worst-case, median, and the 75th percentile were observed to be lower than RT.

Table 6-1. ANOVA F-Test Results Amongst Each Of The Block, Strip, And Point Failure Patterns And For Simulation Failure Rates Of 0.01, 0.005, 0.002, And 0.001

Failure Pattern	θ = 0.01		θ = 0.005		θ = 0.002		θ = 0.001	
	F-Test	Sig.	F-Test	Sig.	F-Test	Sig.	F-Test	Sig.
Block	1259.739	.000	1445.439	.000	1403.759	.000	1402.280	.000
Strip	117.240	.000	100.389	.000	52.962	.000	17.638	.000
Point	25.039	.000	19.489	.000	30.865	.000	34.698	.000

Table 6-2Test For Homogeneity Of Variances For Each Of The Block, Strip, And Point Failure Patterns And Failure Rates Of 0.01, 0.005, 0.002, And 0.001

Failure Pattern	θ = 0.01		θ = 0.005		θ = 0.002		θ = 0.001	
	Levene Statistic	Sig.	Levene Statistic	Sig.	Levene Statistic	Sig.	Levene Statistic	Sig.
Block	3262.136	.000	3492.818	.000	3336.524	.000	3071.961	.000
Strip	166.840	.000	123.604	.000	84.542	.000	24.742	.000
Point	89.833	.000	85.801	.000	127.152	.000	117.795	.000

6.3.2 Formal Analysis: ANOVA, Games-Howell, and Effect Size

To formally analyze the differences amongst the simulation results, an ANOVA test was performed on the populations for each simulated failure pattern and rate. These results are summarized in Table 6-1, and serve to reveal significant differences between the populations. To further characterize these differences, the Levene statistic was calculated for each failure pattern, and associated failure rate to determine if homogeneity of variance between populations could be assumed. The Levene statistics,

Table 6-3 Games-Howell Comparisons Of Block Error Pattern With Simulated Failure Rates: 0.01, 0.005, 0.002, And 0.001.

Algorithm (*x*)	Algorithm (*y*)	θ = 0.01		θ = 0.005		θ = 0.002		θ = 0.001	
		Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.
eAR	FSCS	-.679	.491	-5.006*	.000	-17.716*	.000	-36.483*	.000
	RRT	-3.714*	.000	-11.027*	.000	-41.361*	.000	-85.846*	.000
	Sobol	.570	.684	-3.301*	.001	-43.711*	.000	-162.421*	.000
	RT	-33.009*	.000	-73.361*	.000	-191.455*	.000	-399.896*	.000
FSCS	eAR	.679	.491	5.006*	.000	17.716*	.000	36.483*	.000
	RRT	-3.035*	.000	-6.020*	.000	-23.645*	.000	-49.363*	.000
	Sobol	1.249	.055	1.705	.355	-25.995*	.000	-125.937*	.000
	RT	-32.330*	.000	-68.355*	.000	-173.739*	.000	-363.413*	.000
RRT	eAR	3.714*	.000	11.027*	.000	41.361*	.000	85.846*	.000
	FSCS	3.035*	.000	6.020*	.000	23.645*	.000	49.363*	.000
	Sobol	4.284*	.000	7.726*	.000	-2.350	.889	-76.575*	.000
	RT	-29.295*	.000	-62.335*	.000	-150.094*	.000	-314.050*	.000
Sobol	eAR	-.570	.684	3.301*	.001	43.711*	.000	162.421*	.000
	FSCS	-1.249	.055	-1.705	.355	25.995*	.000	125.937*	.000
	RRT	-4.284*	.000	-7.726*	.000	2.350	.889	76.575*	.000
	RT	-33.579*	.000	-70.061*	.000	-147.744*	.000	-237.475*	.000
RT	eAR	33.009*	.000	73.361*	.000	191.455*	.000	399.896*	.000
	FSCS	32.330*	.000	68.355*	.000	173.739*	.000	363.413*	.000
	RRT	29.295*	.000	62.335*	.000	150.094*	.000	314.050*	.000
	Sobol	33.579*	.000	70.061*	.000	147.744*	.000	237.475*	.000

* The mean difference is *s*-significant at the 0.05 level.

summarized in Table 6-2, yielded significant results for each failure rate, necessitating the rejection of the associated null hypothesis. Therefore, equal variances between populations could not be assumed.

Given that the variances between the populations were not equal, the Games-Howell post hoc test was selected to determine if significant differences exist between the individual testing methods. The results of the Games-Howell tests form the basis from which all subsequent comparisons are presented. A detailed summary of this analysis is presented for each failure pattern (block, strip, and point) in Tables 6-3 to 6-5. The results specifically indicate the increased testing effectiveness of the evolutionary approach compared to the other approaches studied.

To provide a standardized comparison of testing effectiveness for each of the testing methods, an effect size was calculated for each method for which a significant difference against RT was observed. The effect sizes

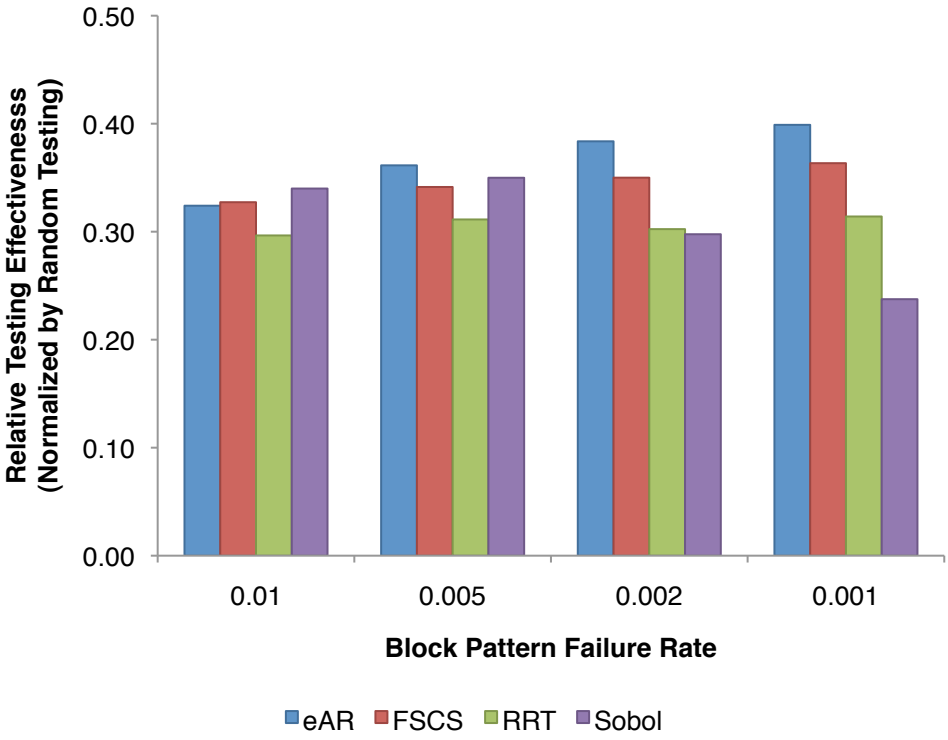


Figure 6-5. Testing Effectiveness For Block Pattern Simulations Of eAR, FSCS, RRT, And The Sobol Sequence Evaluated Against The RT Control Group.

were calculated using Cohen's d method (Cohen, 1988, 1992), and subsequently converted into a Pearson's *r* correlation for interpretation. Cohen offers the following interpretation: small effect size $r = 0.1$, medium $r = 0.3$, and large $r = 0.5$ (Cohen, 1988, 1992). All remaining results will be presented in terms of increased testing effectiveness evaluated against the RT control group established within the study.

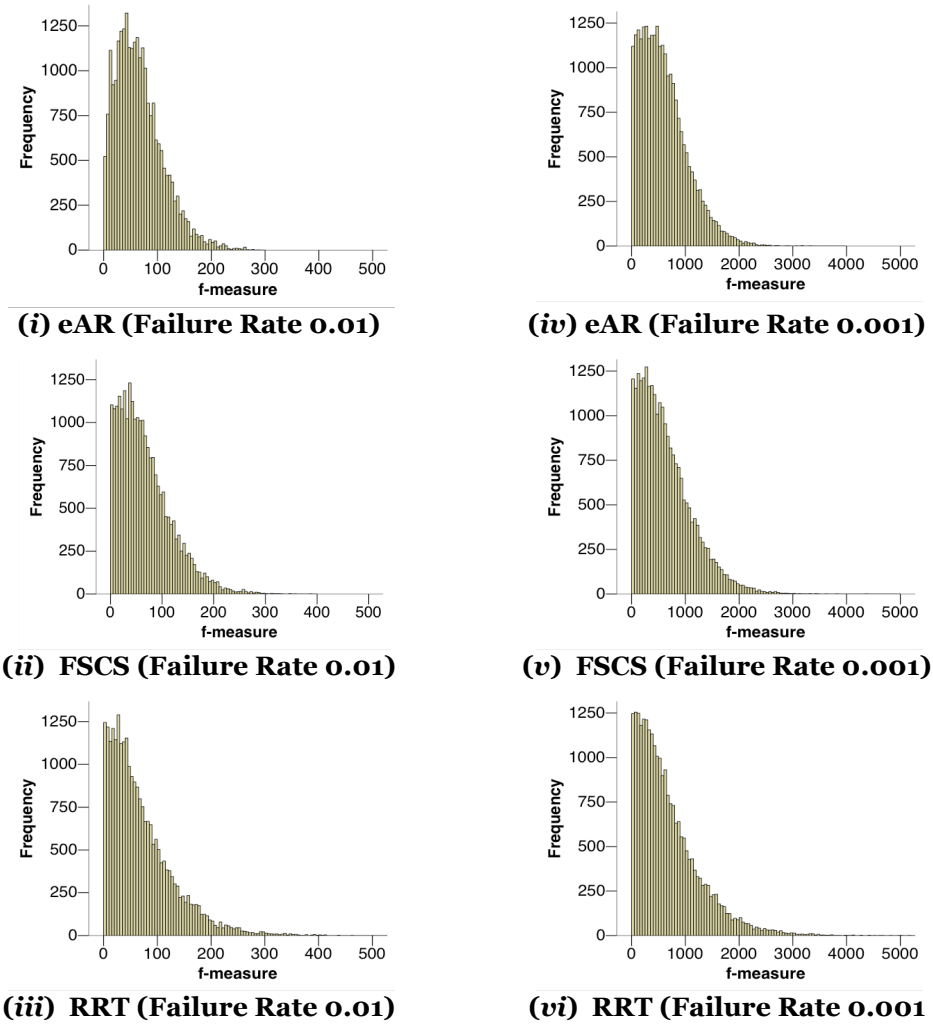
6.3.3 Block Pattern Simulation Results

As highlighted by the box plots, the largest gains in testing effectiveness were observed within the block failure pattern. All of the testing methods (eAR, FSCS, RRT, and the Sobol sequence) significantly outperformed RT with respect to the block failure pattern, as verified by the significant differences reported by the Games-Howell test in Table 6-2. The largest

observed increase in effectiveness (r = 0.178) was achieved by eAR in block pattern simulations with a failure rate of 0.001. Figure 6-5 presents the effect size of the testing effectiveness of each of the strategies in the block pattern simulations. This figure highlights the increased efficiency of eAR when contrasted against FSCS, and RRT for all block pattern simulations, and against the Sobol sequence for all simulations except for a failure rate of 0.01. For the 0.01 failure rate, eAR (r = 0.086), and Sobol (r = 0.086) where observed to have similar testing effectiveness. The Games-Howell comparisons indicated that eAR significantly outperformed all of the other methods studied for block pattern simulations at every failure rate except 0.01. In the block pattern simulations with a failure rate of 0.01, a significant difference was not identified between eAR, FSCS, and Sobol by the Games-Howell tests.

A significant difference was not observed between eAR, FSCS, and Sobol for failure rates of 0.01. This exception may be attributable to the initial propensity of the eAR method to select test inputs that exist on the input domain boundary. This tendency, a byproduct of the effectiveness of eAR in selecting values that are a maximal distance from all other test cases, produces a testing sequence that evenly spaces out test inputs across the entire input domain including the boundary values. As a result, test inputs that exist on the input boundary are less likely to detect a block pattern failure, because these failures are defined to occur completely within the input domain boundaries. For a boundary input to detect a block pattern failure, the edge of the failure region must be situated on the input boundary. This initial impeding effect is observed in side-by-side comparisons of the f-measure distributions of the three ART methods for failure rates of 0.01, and 0.001, presented in Figure 6-6. The distributions of both the FSCS, and RRT remain similar at both failure rates; however, the same cannot be said for eAR. For simulations with a failure rate of 0.01, the peak eAR f-measure frequency was observed to be in the range of 40-45 test cases, with a steep rising trend to the peak. While a similar

peak was observed within the 0.001 failure rate distribution, the pitch of the trend was not nearly as steep, suggesting that the negative effect of the initial boundary inputs decreases as the failure rate decreases.



(*i*) eAR (Failure Rate 0.01) **(*iv*) eAR (Failure Rate 0.001)**

(*ii*) FSCS (Failure Rate 0.01) **(*v*) FSCS (Failure Rate 0.001)**

(*iii*) RRT (Failure Rate 0.01) **(*vi*) RRT (Failure Rate 0.001**

Figure 6-6. Histograms Of the *f-measure* Frequency For eAR, FSCS, RRT Block Pattern Simulations With Failure Rates Of 0.01 And 0.001.

To further understand the differences between eAR, and the other ART methods studied, the frequency of the first 100 f-measure statistics were examined for each of the three ART algorithms at failure levels of 0.01, and 0.001, presented in Figure 6-7. Similar to the box plot observations, it appears that a decrease in failure rate reduces the magnitude of the f-measure frequency, effectively flattening the

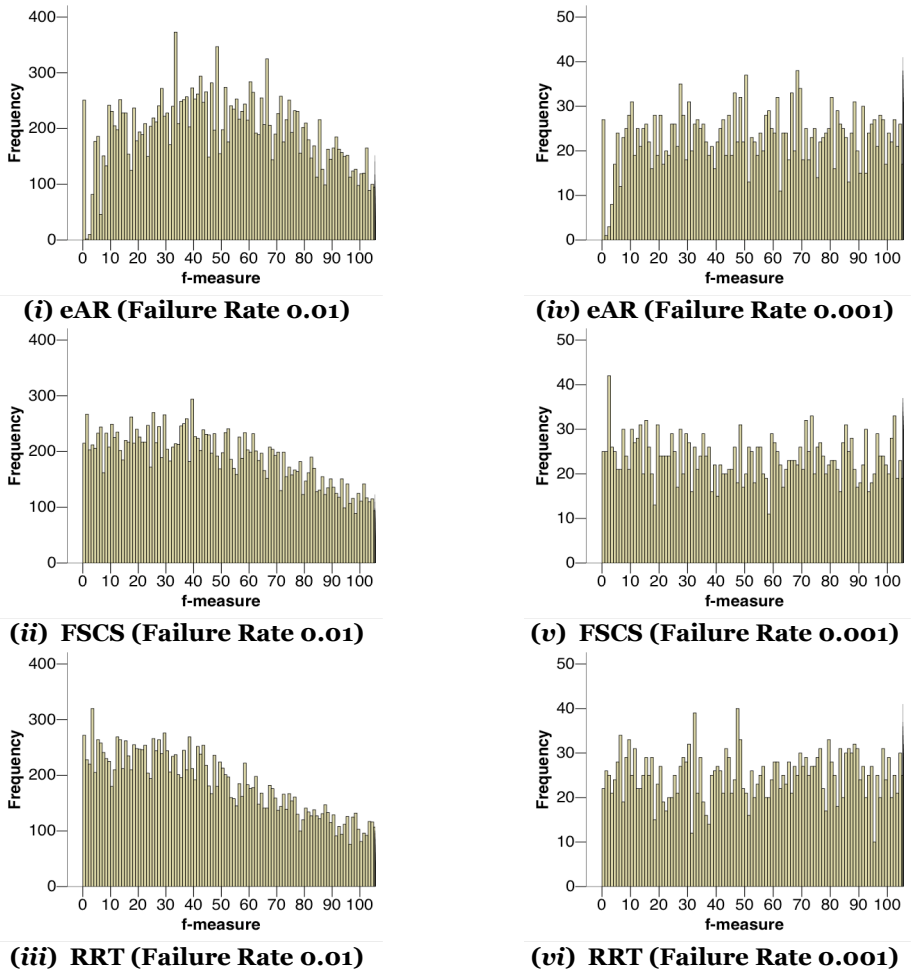


Figure 6-7. Histograms Of First 100 *f-measure* Frequencies For eAR, FSCS, RRT Block Pattern Simulations With Failure Rates Of 0.01 And 0.001.

distributions in the range [1:100] for block pattern simulations with a failure rate of 0.001.

Particular attention is drawn to the f-measure values for eAR simulations for the second, and third test cases, for both the 0.01, and 0.001 simulations. Unlike the other test cases within the testing sequence, these two inputs are dramatically lower than any of the others. These inputs were unique to eAR, and similar points were not observed within the first 100 test inputs of either FSCS, or RRT. These two specific cases represent the two points furthest from the initial random point, corresponding to two opposite corners of the input domain. These two

points, which are very unlikely to detect a block pattern failure, are the most extreme examples of the negative effects of the initial boundary inputs selected by eAR. Although these effects persist with a decreasing failure rate, the impact of these points is diminished due to the decreased number of faults found per test input.

Despite the initial tendency of eAR to produce boundary test inputs, the effectiveness of the method remains *on par* (no significant difference found) with FSCS, and the Sobol Sequence; and significantly above RRT, and RT for failure rates of 0.01. Because the tendency for the algorithm to select boundary values is most apparent in the initial inputs, the effect of this tendency is most pronounced for a failure rate of 0.01. In fact, the median f-measure for a failure rate of 0.01 was found to be an order of magnitude lower than that for a failure rate of 0.001. As the failure rate decreased, the f-measure was observed to increase, indicative of the increased difficulty of detecting smaller failure regions. As the number of tests required to detect a fault increases, the impact of the decreased detection ability of the initial boundary values decreases. This decreased impact of the initial boundary inputs is evidence by the observed results for failure rates lower than 0.01 in Figure 6-5, and by the significant Games-Howell results presented in Table 6-3.

A general upward trend was observed between the testing effectiveness, and block pattern failure rate for the ART methods. The greatest increase in ART effectiveness versus failure rate was observed for the eAR method, and the slowest rate for RRT. The increase in eAR effectiveness is most likely a direct result of suppressing the initial tendency of the algorithm to select boundary values, as previously discussed. The relationship between Sobol sequence testing effectiveness and failure rate was very different from that of the ART methods. In all simulations with a failure rate lower than 0.005 the Sobol sequence decreased in effectiveness, whereas all of the ART methods demonstrated increased effectiveness. This finding suggests that the performance of

Table 6-4. Games-Howell Comparisons Of Strip Error Pattern With Simulated Failure Rates: 0.01, 0.005, 0.002, And 0.001.

Algorithm (*x*)	Algorithm (*y*)	θ = 0.01		θ = 0.005		θ = 0.002		θ = 0.001	
		Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.
eAR	FSCS	1.129	.564	4.536*	.036	13.852*	.008	24.046*	.038
	RRT	-1.329	.395	1.335	.918	11.463*	.046	16.958	.270
	Sobol†	-14.227*	.000	-28.818*	.000	-43.330*	.000	-36.256*	.001
	RT	-13.015*	.000	-16.319*	.000	-16.832*	.001	-24.424*	.042
FSCS	eAR	-1.129	.564	-4.536*	.036	-13.852*	.008	-24.046*	.038
	RRT	-2.458*	.010	-3.202	.242	-2.389	.978	-7.088	.915
	Sobol†	-15.357*	.000	-33.354*	.000	-57.182*	.000	-60.302*	.000
	RT	-14.144*	.000	-20.855*	.000	-30.684*	.000	-48.470*	.000
RRT	eAR	1.329	.395	-1.335	.918	-11.463*	.046	-16.958	.270
	FSCS	2.458*	.010	3.202	.242	2.389	.978	7.088	.915
	Sobol†	-12.898*	.000	-30.152*	.000	-54.793*	.000	-53.214*	.000
	RT	-11.686*	.000	-17.653*	.000	-28.295*	.000	-41.382*	.000
Sobol†	eAR	14.227*	.000	28.818*	.000	43.330*	.000	36.256*	.001
	FSCS	15.357*	.000	33.354*	.000	57.182*	.000	60.302*	.000
	RRT	12.898*	.000	30.152*	.000	54.793*	.000	53.214*	.000
	RT	1.213	.865	12.499*	.000	26.498*	.000	11.832	.697
RT	eAR	13.015*	.000	16.319*	.000	16.832*	.001	24.424*	.042
	FSCS	14.144*	.000	20.855*	.000	30.684*	.000	48.470*	.000
	RRT	11.686*	.000	17.653*	.000	28.295*	.000	41.382*	.000
	Sobol†	-1.213	.865	-12.499*	.000	-26.498*	.000	-11.832	.697

** The mean difference is s-significant at the 0.05 level.*
† There were 11 strip failure patterns that could not be detected by the 10,000 inputs generated by the Sobol Series for failure rate θ = 0.001.

quasi-random testing strategies decreases with the failure rate. However, this result cannot be considered definitive, and a more in-depth study of quasi-random testing is required to fully understand this trend.

6.3.4 Strip Pattern Simulation Results

With respect to the strip pattern, the Games-Howell comparisons presented in Table 6-4 confirm that the ART methods significantly outperformed RT for all failure rates. Figure 6-8 presents the testing effectiveness in terms of effect size *r* for the strip pattern simulations. This figure highlights an interesting situation; it represents the only failure pattern for which the effectiveness of the ART methods decrease with the failure rate. It may be observed from Figure 6-8 that FSCS was the best performer, performing significantly better than RRT for failure rates of

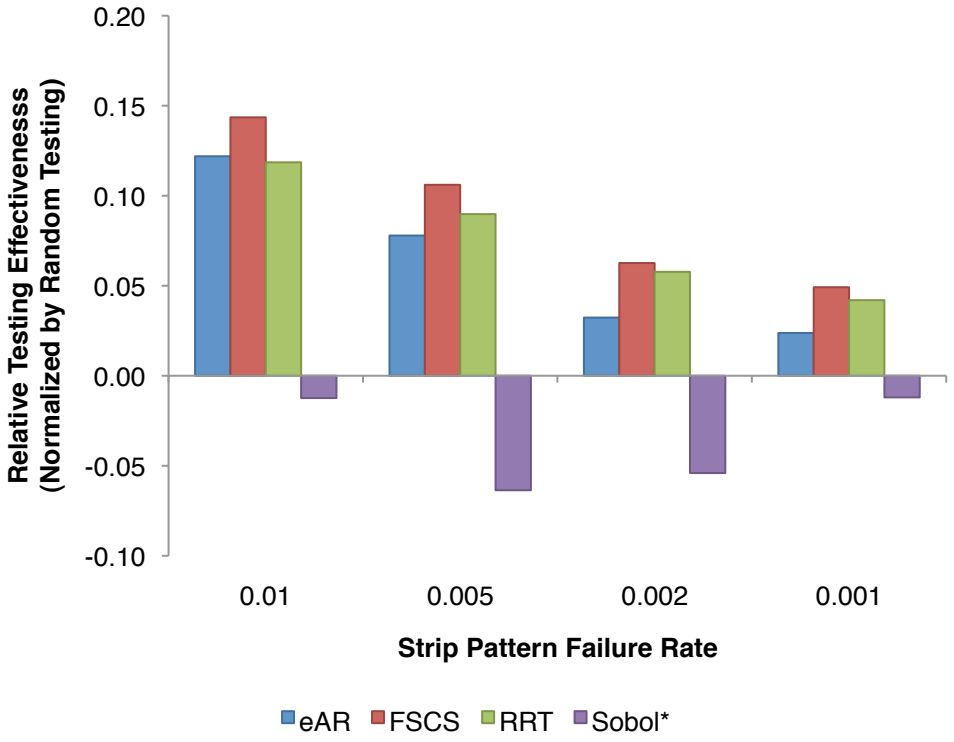


Figure 6-8. Testing Effectiveness For Strip Pattern Simulations Of eAR, FSCS, RRT, And The Sobol Sequence Evaluated Against The RT Control Group.

0.01; and significantly better than eAR for failure rates of 0.005, 0.002, and 0.001. eAR (r = 0.032) was observed to have a higher effectiveness than RRT (r = 0.029) relative to RT for the failure rate of 0.01, but the mean difference between these two populations was not found to be significant. Conversely, RRT was observed to have a greater effectiveness relative to RT for failure rates less than 0.01; however, the mean difference between the eAR and RRT populations was found to be significant in only one case, for a failure rate of 0.002.

The Sobol sequence was the only testing method found to be significantly less effective than RT, both in terms of mean difference, and effect size. For all simulated failure rates, the Sobol sequence was found to be significantly less effective than all three of the ART strategies; and for failures rates of 0.005, and 0.002, the Sobol sequence was even found to

be significantly less effective than RT. This was a surprising result due to the increased testing effectiveness observed for the Sobol sequence with respect to the block failure pattern. However, the result clearly demonstrates that a testing strategy can be adept at detecting a specific type of fault, in this case a fault characterized by its error region, while at the same time be quite inept at detecting other faults. It is important to note that, although block, strip, and point patterns do not effectively cover the enormous breadth of possibilities of error regions, they do provide in this case sufficient variation to elicit the strengths and weaknesses of various testing strategies. While the claim that ART methods are better or at least no worse than RT remains undisputed, it is clear that this claim should be rejected for the Sobol sequence. Further study is required to hone quasi-random techniques to a point at which they perform at least as well as RT for faults characterized by strip failure regions.

With respect to the testing effectiveness of the methods for strip pattern simulations, a distinct downward trend is observed in Figure 6-8 when compared against Figure 6-5, and Figure 6-9. This distinction suggests that the failure rate does not have the same effect on the testing effectiveness of ART for the strip pattern as was observed for the block and point patterns. A generally upward trend was observed amongst ART methods for the block, and point patterns with a decreasing failure rate, while amongst the strip pattern simulations the opposite effect was observed. While the block, and point patterns represent simple failure regions, each fixed about a stationary point or points, the strip pattern represents a linear relationship between two variables. This increased complexity of the relationship between the two axes could be the cause of the observed anomaly; however, the exact cause of these results is not fully understood. Despite the increased complexity encapsulated within the strip pattern, the three ART methods were found to significantly outperform RT at all failure rates.

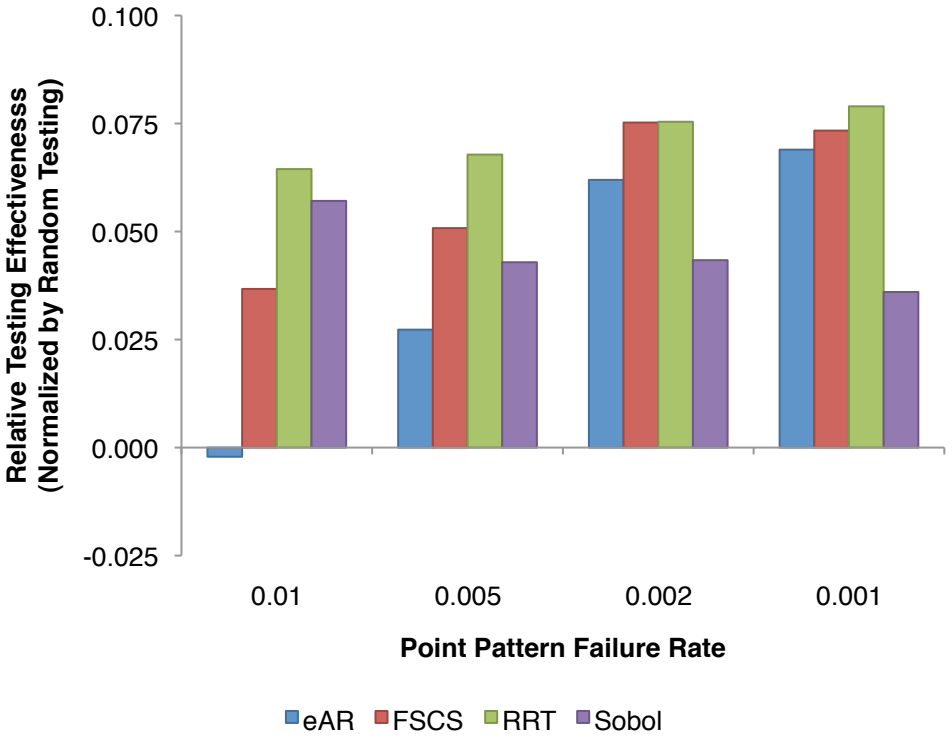


Figure 6-9. Testing Effectiveness For Point Pattern Simulations Of eAR, FSCS, RRT, And The Sobol Sequence Evaluated Against The RT Control Group.

6.3.5 Point Pattern Simulation Results

Point pattern simulation yielded results similar to the strip pattern; ART methods performed slightly better, and not worse than RT, with significant effect sizes ranging from $r = 0.009$ to $r = 0.030$. All of the testing methods significantly outperformed RT with the exception of eAR at the failure rate 0.01, where a significant difference between eAR and RT populations could not be verified by the Games-Howell test. Comparisons between the individual testing methods revealed subtle differences between the testing results, suggesting that at lower failure rates the performance of the three ART methods, while indistinguishable, were significantly better than RT, and the Sobol sequence. Of the twelve Games-Howell comparisons amongst ART methods presented in Table 6-5, only four were significant. At the failure rate of 0.01, eAR was observed to perform significantly worse than FSCS, and RRT; and at 0.005, eAR performed significantly

Table 6-5. Games-Howell Comparisons Of Point Error Pattern With Simulated Failure Rates: 0.01, 0.005, 0.002, And 0.001.

Algorithm (*x*)	Algorithm (*y*)	θ = 0.01		θ = 0.005		θ = 0.002		θ = 0.001	
		Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.	Mean Diff. (*x*-*y*)	Sig.
eAR	FSCS	2.848*	.002	3.643	.119	5.645	.566	3.409	.991
	RRT	5.598*	.000	7.007*	.000	5.740	.548	9.027	.758
	Sobol	4.870*	.000	2.089	.678	-10.313	.076	-33.947*	.000
	RT	-.797	.874	-6.386*	.001	-32.043*	.000	-69.947*	.000
FSCS	eAR	-2.848*	.002	-3.643	.119	-5.645	.566	-3.409	.991
	RRT	2.750*	.003	3.364	.176	.095	1.000	5.619	.948
	Sobol	2.022	.082	-1.555	.865	-15.958*	.001	-37.355*	.000
	RT	-3.645*	.000	-10.029*	.000	-37.688*	.000	-73.356*	.000
RRT	eAR	-5.598*	.000	-7.007*	.000	-5.740	.548	-9.027	.758
	FSCS	-2.750*	.003	-3.364	.176	-.095	1.000	-5.619	.948
	Sobol	-.728	.889	-4.919*	.016	-16.053*	.001	-42.974*	.000
	RT	-6.395*	.000	-13.393*	.000	-37.783*	.000	-78.974*	.000
Sobol	eAR	-4.870*	.000	-2.089	.678	10.313	.076	33.947*	.000
	FSCS	-2.022	.082	1.555	.865	15.958*	.001	37.355*	.000
	RRT	.728	.889	4.919*	.016	16.053*	.001	42.974*	.000
	RT	-5.667*	.000	-8.475*	.000	-21.731*	.000	-36.001*	.000
RT	eAR	.797	.874	6.386*	.001	32.043*	.000	69.947*	.000
	FSCS	3.645*	.000	10.029*	.000	37.688*	.000	73.356*	.000
	RRT	6.395*	.000	13.393*	.000	37.783*	.000	78.974*	.000
	Sobol	5.667*	.000	8.475*	.000	21.731*	.000	36.001*	.000

* The mean difference is *s*-significant at the 0.05 level.

worse than RRT. After this threshold, the results of each of the three ART methods began to converge, and significant differences between the ART populations could not be verified. This convergence suggests that, as the failure rate increases, the gains in testing effectiveness for failure regions similar to the point pattern are equal amongst the ART techniques.

The mean difference observed between RT and eAR for the point pattern with a simulated failure rate of 0.01 was the only comparison against RT that was not verified to be significant. This underachievement is attributed to the way in which the eAR method selects the initial boundary value test inputs. The point pattern as defined (T. Y. Chen et al., 2006) requires that all points exist completely within the input boundary. This definition severely restricts boundary inputs to only detect a failure in the case when the circumference of the error region just touches the boundary at the point where the test resides. Because of this definition,

test inputs that lay on the boundaries are unlikely to detect a point failure, and inputs that reside in any of the four corners simply cannot detect a failure region. Although the inability to detect point pattern failure regions using inputs that exist within the boundary corner applies to all testing methods, this limitation most severely affects the performance of eAR because the algorithm initially selects at least one of the boundary corners. As observed for the block pattern simulations, as the failure rate is decreased, the impact of the initial boundary inputs also decreases. As such, for all failure rates above 0.01, eAR was found to perform significantly better than RT. Furthermore, the significant disparity observed between eAR and the other testing methods was all but erased by a failure rate of 0.002 as the ART methods began to converge.

As previously observed amongst the block and strip patterns, the Sobol sequence exhibited different behavior than the ART methods. While the ART methods all displayed an upward trend in testing effectiveness, the Sobol sequence was the only testing method whose effectiveness did not increase with the failure rate (r = 0.014 for θ = 0.01, and r = 0.013 for θ = 0.001). This trend was also observed for the block failure pattern, and suggests that for the testing of established software endeavors, i.e. mature software projects with a low failure rate, the Sobol sequence is not the best choice when compared to the state-of-the-art offerings from the ART community.

6.3.6 Empirical Runtime Results

Based on the definitions of the algorithms, and the associated runtime analysis, it is clear that differences in terms of computation costs exist between the algorithms studied. The differences between the ART methods (eAR, FSCS, and RRT), and the Sobol sequence were directly proportional to the size of the test suite; the Sobol sequence required constant time to generate the next test case, and the ART methods increased linearly with the size of the test suite.

Table 6-6. Relative Runtime Cost For eAR vs FSCS, and eAR vs RRT.

	Minimum	Average	Maximum
eAR vs FSCS	2.1	5.3	8.4
eAR vs RRT	1.4	29.4	80.8

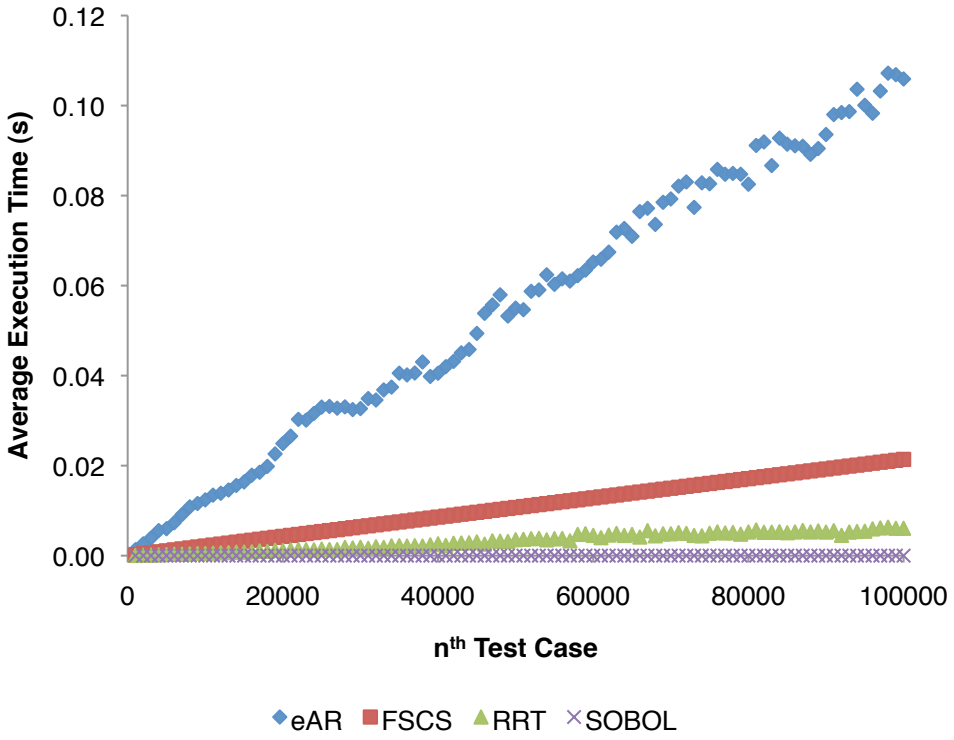


Figure 6-10. Mean Runtime Of eAR, FSCS, RRT, And The Sobol Sequence To Generate The *n*th Test Case.

To study the runtime differences between the four algorithms, an empirical analysis was conducted to demonstrate the performance costs associated with each algorithm. The mean runtimes for each of the algorithms is presented in Figure 6-10. On average, eAR and FSCS are differentiated by a factor of 5.3, while eAR and RRT are differentiated by a factor of 29.4. As shown in Table 6-6, the runtime of eAR was at best only 2.1 times worst than FSCS, and 1.4 times worst than RRT; at worst, the runtime of eAR was 8.4 times worse than FSCS, and 80.8 times worse than RRT.

The relative differences between eAR compared to FSCS and RRT are smaller than the theoretical worst-cases presented in Section 6.1.4. This discrepancy is explained by the conservative values chosen as parameters for eAR. The genetic algorithm population converges upon a suitable solution quickly; and because the algorithm only recalculates the fitness for individuals with differing values from their parents, the algorithm's performance is greatly improved. As mentioned in Section 6.1.3, parameters controlling population size, crossover possibility, mutation possibility, and the stoppage criterion were held constant to avoid problem over-optimization; thus the worst-case runtime associated with these constants is conservative.

Despite the difference in execution time between the testing algorithms, it is important to note that the runtime associated with each is marginal (less than 1 second). The instrumented application was run on an Intel Core2 Quad Processor Q6700 (2.66 GHz) with 8 GB of RAM, and was able to compute the 100,000th eAR test case in less than 0.2 seconds. In this instance, the selection of the ART algorithm from a runtime cost standpoint was inconsequential, suggesting that the choice should be driven fundamentally by demonstrated testing effectiveness. However, in practice, the costs of test generation should be weighed carefully against the cost of test case execution.

6.4 Evolutionary Adaptive Random Testing and Cookie Collection Testing

The use of the anti random testing algorithm for cookie collection testing is simply not feasible given the intractability of the anti random algorithm for test vectors of considerable length. As anti random testing requires the iteration of the entire vector space, the asymptotic runtime is in the order of exponential complexity ($anti\ random(t) \in O(2^t)$, where t is the number of elements contained within a test vector). This runtime complexity is not

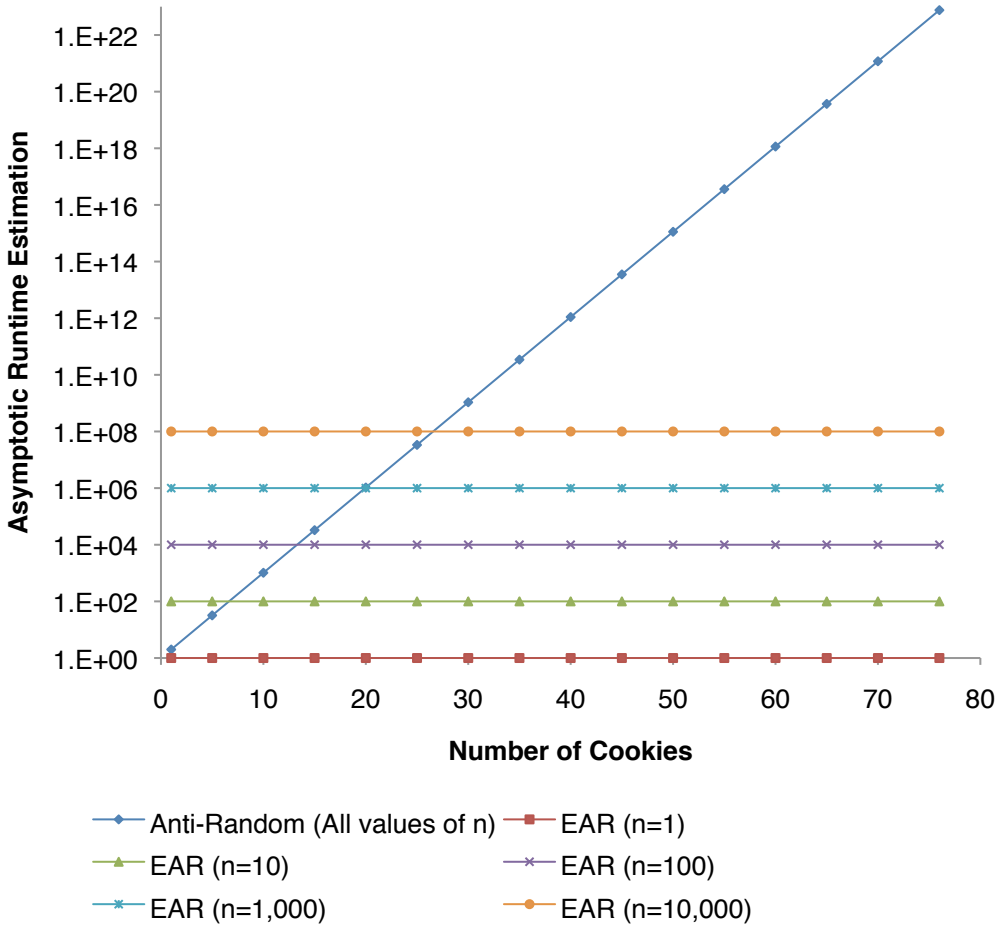


Figure 6-11. Asymptotic Runtime Comparisons

computationally feasible for any testing problem of considerable size (Cobham, 1964).

Evolutionary adaptive random testing, on the other hand, is in the order of quadratic time complexity and is not tied to the number of elements within a test vector ($eAR(n) \in O(n^2)$, where n is the number of tests within the input test suite), but rather the size of the testing suite. Figure 6-11 presents a comparison of the asymptotic runtimes of the anti random and eAR testing algorithms. This figure is based upon the range of cookies used by actual sites as documented in Chapter 3.2.6. Asymptotic runtime analysis only considers the upper bound of the growth rate of a function. As such, it may be noted that the scaling effect which n

has upon eAR also pertains to the anti random testing algorithm applied across an entire test suite. However, as displayed in Figure 6-11, the scaling effect was omitted from the asymptotic runtime analysis because it becomes insignificant in comparison to the size of the test vector for sufficiently large test vectors.

6.4.1 Evolutionary Adaptive Random Testing Definition

The definition of the eAR testing algorithm utilized within cookie collection testing will be provided in terms of problem encoding and fitness function—all other algorithm parameters including selection mechanism, crossover, mutation, stoppage criterion, and population size were consistent with Section 6.1.3.

Encoding. To encode the cookie collection testing problem, each test vector was comprised of one bit for each of the cookies present within a global cookie collection—one representing the presence of a cookie, zero representing its absence.

Fitness. To provide the best coverage of testing vectors across the input domain, the hamming distance was used as the basis for the fitness function (Malaiya, 1995). The fitness function is defined as

$$Fitness(x,T) = \min_{y \in T}\left(\sum_{i=0}^{n} |x_i - y_i|\right), \tag{6-2}$$

where x is the vector (chromosome) for which the fitness is being calculated, T is the set of pre-existing test vectors, y is a test vector within T, and n is the number of elements within the test vector x. The value *Fitness(x,T)* was then assigned to each chromosome x, and used to select individuals for reproduction.

A detailed description of the testing harness developed for the automation of cookie collection testing is documented in Appendix B.

6.5 Summary of Results and Key Findings

ART techniques demonstrated the greatest gains in testing effectiveness for failure regions that are most closely represented by the block pattern. Evolutionary adaptive random testing was found to perform significantly better than all the other testing methods studied for block pattern simulations with failure rates below 0.01.

Although the performance of eAR did not significantly exceed the other ART methods for point and strip pattern simulations, eAR was demonstrated to be of similar effectiveness, and of increased effectiveness compared to the Sobol sequence and RT. While increases in testing effectiveness were observed for the ART methods in the strip and point patterns, these increases were not of the same magnitude as those achieved in the block pattern simulations. These results are in agreement with previous claims that ART methods perform slightly better than, or at least no worse than, RT. The effect sizes associated with the differences between ART and RT for point and strip pattern simulations showed an increased testing effectiveness ranging from $r = 0.009$ to $r = 0.034$. Although these differences are significant, they are not as large as those observed for the block pattern simulations (effect sizes ranging from $r = 0.074$ to $r = 0.178$), representing a fivefold increase in testing effectiveness when comparing ART methods to RT.

The results of the simulation study clearly answer the research questions posed in Section 6.2.1. The application of an evolutionary search algorithm is applicable within the domain of ART. Evolutionary adaptive random testing was found to offer the largest increase in effectiveness in block pattern simulations, the domain in which ART has been demonstrated most effective, and was shown to generate testing inputs with a marginal increase in overhead cost. Based on the increased testing effectiveness, evolutionary adaptive random testing will be used for the generation of testing data for cookie collection testing.

Chapter 7

Cookie Collection Testing: An Empirical Evaluation

Software testing has been defined by G. J. Myers as "the process of executing a program with the intent of finding errors" *(1976, p. 6)*. Given this singular purpose, empirical evaluations of testing theories focus upon one central concept—the ability of a testing theory to detect bugs. The ability to detect defects must be evaluated against the costs associated with fault detection. It is within this tension that modern testing strategies exist—seeking to maximize fault detection while minimizing testing effort.

Cookie collection testing, as defined in Chapter 5 and implemented by the testing tool CookieCruncher in Appendix B, exists as a low-cost automated testing tool for modern web applications. The automated nature of the tool allows for testers to quickly define and create testing suites uniquely suited to the cookie collections present within a web application. Although this ability entails relatively low cost, it remains, until now, untested. This chapter will examine cookie collection testing as implemented through the adaptation of Evolutionary Adaptive Random Testing, as describe in Chapter 6.4, providing concrete empirical evidence that the testing theory is an effective testing tool, capable of finding defects in real-world web applications.

The remainder of the chapter will be organized as follows. Section 7.1 will outline the evaluation methodology used to assess the testing strategy; Section 7.2 will present empirical data drawn from a study of the testing strategy applied to six real-world web applications, with analysis

and discussion of the results; finally, Section 7.3 will summarize the results and highlight the key findings.

7.1 Evaluation Methodology

As the first testing strategy to actively evaluate a web application from the perspective of cookie collection modification, the evaluation focused primarily upon the question of fault detection. Within the testing strategy a number of comparisons were undertaken to understand which facets were the most cost-effective. In order to evaluate the cookie collection testing strategy defined in Chapter 5 and implemented using evolutionary adaptive random testing as outlined in Chapter 6 an empirical evaluation was undertaken.

7.1.1 Research Questions

The empirical evaluation addressed the following research questions:

- ***Q1.*** *Can cookie collection testing reveal defects in web applications?*
- ***Q2.*** *What is the relationship that exists between faults triggered by seeding and generated vectors? Are the two complementary, or does one supersede the other?*
- ***Q3.*** *How do web applications respond to cookie collection testing? Are the responses uniform or application-specific?*
- ***Q4.*** *Does a relationship exist between tree and context similarity coefficients?*
- ***Q5.*** *Are the tree similarity and context similarity coefficients useful at detecting web application defects? Is one superior or is a composite metric most effective?*

7.1.2 Experimental Design

Six open source web applications were chosen as test subjects to which cookie collection testing was applied. Due to the adaptive-random nature of test input generation, and the sheer size of test data generated, this

process was automated through the use of the testing tool CookieCruncher. Each of the six applications where subjected to use-case analysis from which application specific test cases were drawn.

From each of the test-suites, a smaller subset of 100 test cases were randomly selected and analyzed manually from a binary perspective—Pass or Fail. A test case was said to Pass if, and only if, the observable test result was deemed equivalent to that of the test oracle output. This process required manual inspection due to noise present within the output of the applications (non-consistent artifacts within the normal operation of the web applications). The results of the binary evaluation process were then used as a basis from which an optimal threshold-classification value could be assigned to each similarity coefficient.

7.1.3 Test Application Selection

Six open source web applications were selected for testing, based primarily on the underlying cookies each application represented. Open source applications were used because of the ease of access to the underlying application, allowing repeatability of the experiments. A summary of the applications is provided in Table 7-1.

Table 7-1. Test Applications Summary

Web Application	Tested Version	Complexity Metric (LOC)	URL
BugTracker.net	3.2.5	52,147	http://ifdefined.com/bugtrackernet.html
e107	0.7.16	175,265	http://e107.org/
GeekLog	1.5.0rc1	118,706	http://www.geeklog.net/
phpBB2	2.0.12	43,831	http://www.phpbb.com/
phpBB3	3.0.5	203,465	http://www.phpbb.com
phpMyAdmin	3.2.1	194,870	http://www.phpmyadmin.net

All of the selected applications were found to use cookies from a wide variety of categories, including first-party, sessional, persistent, httponly, and secure. The cookie usage within the web applications ranged from three to ten. Within the survey conducted in Chapter 3, it was found

that sites having between one and ten cookies accounted for over 92% of web applications. A summary of the cookie usage of each test application is presented in Table 7-2.

Table 7-2. Test Application Cookie Usage Summary

Cookie Classification	Web Application					
	BugTracker.net	e107	GeekLog	phpBB2	phpBB3	phpMyAdmin
All Cookies	5	5	8	3	4	10
First-Party	5	5	8	3	4	10
Sessional	2	3	1	2	4	1
Persistent	3	2	7	1	0	9
HttpOnly	1	0	0	0	3	9
Secure	0	0	0	0	0	7
Path Specific	2	1	1	1	1	1

Other factors were considered in the selection of the test applications. From a web-technologies perspective, the six applications employ a wide variety of platforms and methods for content delivery. The selected applications also represent the two most commonly encountered web-programming languages, ASP and PHP, as described in Chapter 4.3.1. The applications are served by two competing HTTP server platforms, Apache HTTP Server and Microsoft's Internet Information Services (IIS), two database servers, mySQL and Microsoft SQL Server, and are run on three distinct platforms—Ubuntu Linux 9.04, Windows server 2008, and Mac OSX 10.6. Furthermore, the selected applications employ a variety of underlying web-technologies and HTML elements such as AJAX, CSS, and Frames. A brief description of each web application and its installation environment and related technologies is provided in Appendix C.

7.1.4 Analysis Tools

Given that the testing process is based upon adaptive-random techniques and the use of the automated testing tool, a large pool of testing data was available for analysis. Because of the non-parametric nature of the data, the Mann-Whitney U test was selected to analyze the differences between testing results. The null hypothesis of the Mann-Whitney U test is that the two groups of test-results come from a single population. Essentially, this

test will allow for the identification of a significant difference between two populations. The null hypothesis for all statistical tests was rejected if the significance of the result was below the standard Type 1 error rate, 0.05.

Analysis of the binary Pass/Fail experiment was conducted using the receiver operating characteristic (ROC) curve to assess the optimal decision-threshold classifier (Fawcett, 2003). An ROC curve is a plot of the true-positive rate versus the false-positive rate under a variety of threshold values. Each of the threshold values can be thought of as a trade-off between false-positive and false-negative identifications. The area under the ROC curve provides an indication of the quality of the underlying diagnostic test. An area of 1.0 represents a perfect test, and an area of 0.5 represents a random guess, equivalent to making a decision based upon the flip of a coin. The ROC curve is used extensively in evaluation of medical diagnostic tests (Hanley & McNeil, 1982; Zweig & Campbell, 1993) and has recently become prevalent within the field of machine learning and data mining (Bradley, 1997; Fawcett, 2003). ROC curves will be used to select threshold values for each test suite, and the area under the curve will be used as a primary means of comparing the effectiveness of each similarity coefficient.

Finally, The Pearson product-moment correlation coefficient (Pearson's *r*) was selected to calculate the correlation between the similarity coefficients, and to deduce the relationships between these measures. Pearson's *r* was chosen for its widespread use and robustness against non-normal data (Carroll, 1961; Nefzger & Drasgow, 1957; Rodgers & Nicewander, 1988). To further enhance this analysis, the Pearson coefficients were attributed a measure of effect size—*small,* $0.1 < r \leq 0.3$*; medium,* $0.3 < r \leq 0.5$*;* and *large*, $0.5 < r \leq 1$; according to Cohen (Cohen, 1988, 1992).

7.2 Experimental Results and Discussion

7.2.1 Initial Findings – A Comparative Analysis

The use of the automated testing tool enabled the definition and execution of 29,057 individual tests spread across 675 test requests. This large test pool will serve as the basis for subsequent discussions and evaluations of the cookie collection testing strategy. A summary of the descriptive statistics of the test results for each of the six applications, based on the tree similarity, context similarity, and composite similarity coefficients, is presented in Table 7-3. Although the distribution of Jaccard similarity coefficients is known to be asymptotically normal when the underlying populations are assumed to follow equiprobable multinomial distributions (McCormick, Lyons, & Hutcheson, 1992), this is not the case for the current data set, and the distribution of the testing results remains unidentified. Because the data distribution was not identifiable, the following sections will employ robust non-parametric statistical tests and descriptions in the presentation and analysis of the testing results.

Table 7-3. Descriptive Statistics of Test Results

Web Application		N	Min.	Max.	Median	Mode	Range
BugTracker.net	Tree Similarity	1440	.014	1.000	1.000	1.000	0.986
	Context Similarity	1440	.000	1.000	1.000	1.000	1.000
	Composite Similarity	1440	.010	1.000	1.000	1.000	0.990
e107	Tree Similarity	1024	.235	1.000	0.996	1.000	0.765
	Context Similarity	1024	.203	1.000	0.957	1.000	0.797
	Composite Similarity	1024	.221	1.000	0.976	1.000	0.779
GeekLog	Tree Similarity	2232	.577	1.000	1.000	1.000	0.423
	Context Similarity	2232	.366	1.000	0.979	1.000	0.634
	Composite Similarity	2232	.507	1.000	0.990	1.000	0.493
phpBB2	Tree Similarity	580	.839	1.000	1.000	1.000	0.161
	Context Similarity	580	.609	1.000	1.000	1.000	0.391
	Composite Similarity	580	.733	1.000	1.000	1.000	0.267
phpBB3	Tree Similarity	2080	1.000	1.000	1.000	1.000	0.000
	Context Similarity	2080	.864	1.000	1.000	1.000	0.136
	Composite Similarity	2080	.935	1.000	1.000	1.000	0.065
phpMyAdmin	Tree Similarity	21701	.020	1.000	0.976	1.000	0.980
	Context Similarity	21701	.000	1.000	0.958	0.958	1.000
	Composite Similarity	21701	.014	1.000	0.967	0.967	0.986

The values of the tree and context similarity coefficients can range between [0: 1] because of the use of the Jaccard similarity coefficient in the calculation of both measurements. The composite similarity coefficient shares the same extreme values, given that is it the normalized Euclidian distance of a (tree similarity, context similarity) data point from the origin (0, 0), as described in equation (5-7). This normalized range allows for direct comparison of the test results from the similarity coefficients across all test subjects. To this end, Figures 7-1 to 7-3 provide box plots of the similarity coefficients for all six of the test applications. Content similarity was observed to exist within the range of [0: 1], whereas tree similarity ranged from [0.014: 1]. This difference exists due to the definition of the DOM tree and its implementation within the Firefox browser (Mozilla Developer Center, 2009; W3C, 2005). Because all DOM trees share a common root node (*Document*), the tree similarity coefficient always returns at least one common node. In practice this phenomenon is not only limited to the *document* node, but extends to the *HTML*, *HEAD*, and *BODY* nodes as these are present within all HTML documents, and are automatically added by Firefox if the return document is not compliant. These common nodes are responsible for the observed differences between the context and tree similarity coefficients.

While globally the similarity coefficients ranged between [0: 1], the lower bound values varied greatly between the different test subjects. BugTracker.net and phpMyAdmin were observed to have the largest spreads, with tree similarity coefficients ranging [0.014: 1.0] and [0.020: 1.0] respectively; they were also the only two test subjects with context coefficients ranging [0.0: 1.0]. The respective lower bound values for e107, GeekLog, and phpBB2 were 0.235, 0.577, and 0.839 for the tree similarity coefficient and 0.203, 0.366, and 0.609 for the context similarity coefficient. phpBB3 had the smallest data spread amongst the six applications and was observed to have a tree similarity range of [1.0: 1.0] and a context similarity range of [0.864: 1.0]. This result is due

to the maturity of phpBB3 and its robustness against cookie collection manipulation.

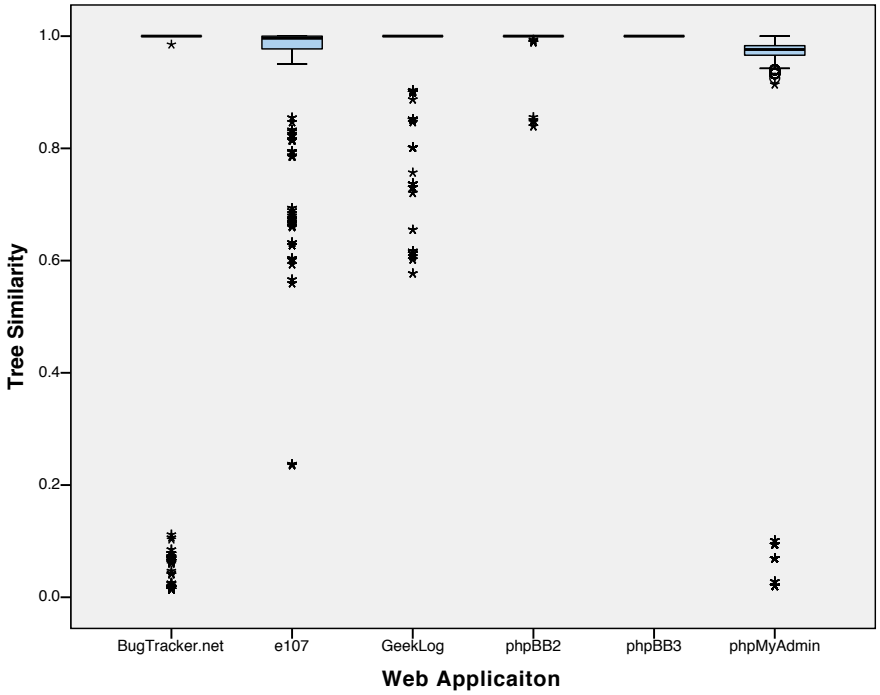


Figure 7-1. Box Plots Of The Tree Similarity Coefficient

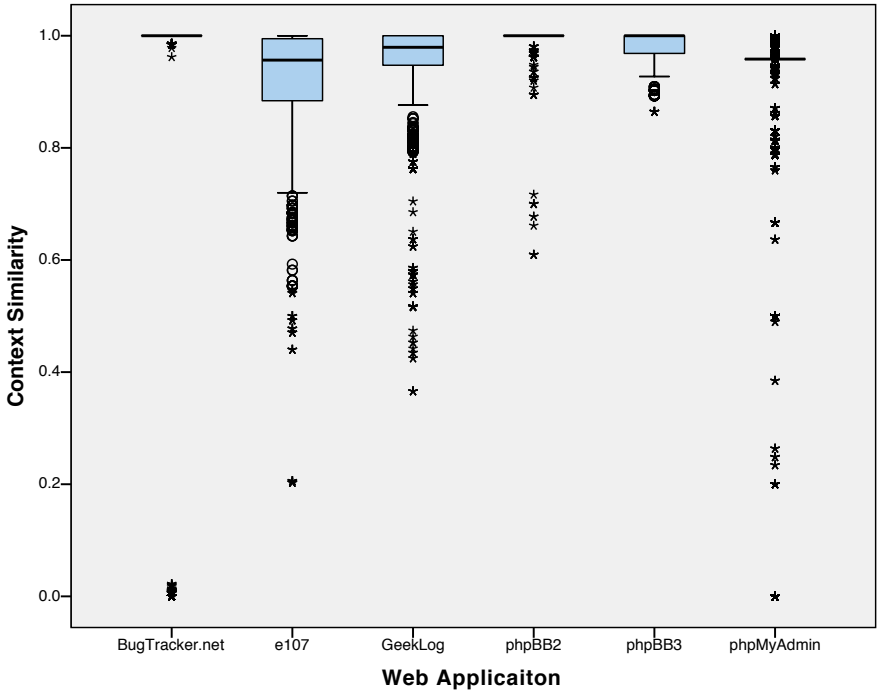


Figure 7-2. Box Plots Of The Context Similarity Coefficient

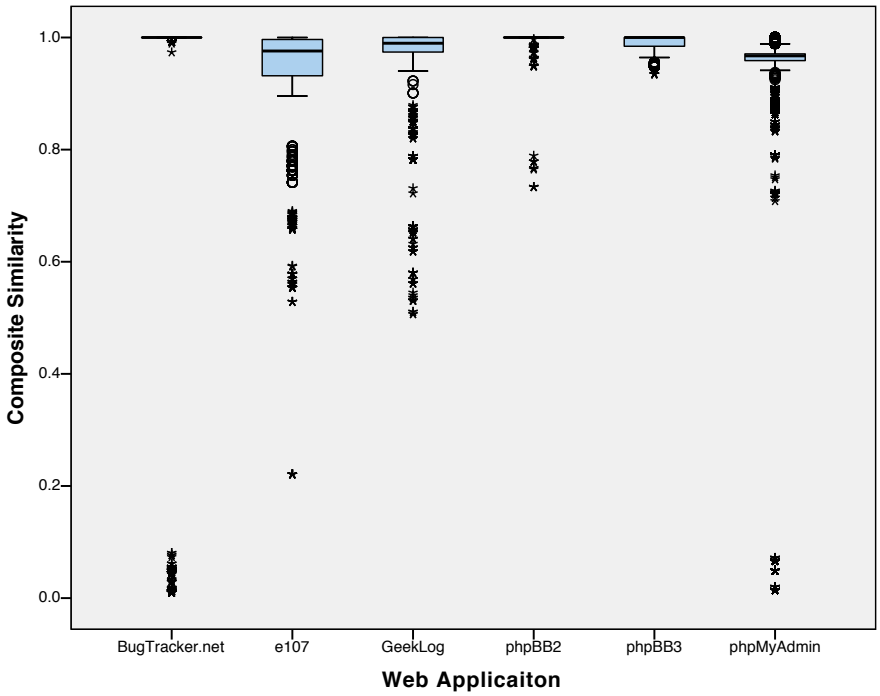


Figure 7-3. Box Plots Of The Composite Similarity Coefficient

Unlike for the lower bound values, both similarity coefficients attained the maximum upper bound value. A similarity coefficient result of 1.0, indicating that two pages are identical, was the median result for both similarity coefficients in three of the six test subjects, and was the mode for across all applications save one—the context similarity for phpMyAdmin. Although a perfect 1.0 should indicate equivalence, Section 7.2.3 will establish a testing threshold for each of the applications, providing a limit at which it is likely that a fault has occurred. Without such a threshold value, a test practitioner should simply organize the results in ascending order based on the composite similarity coefficient, and analyze system faults in the form a prioritized fault queue. An individual assessment of the each of the test programs will be provided in Section 7.2.4, after a testing threshold has been established for each of the similarity coefficients.

Table 7-4. Mann-Whitney U Tests Between Test Applications for Tree and Context Similarity

Web Application		Tree Similarity			Context Similarity		
		Mann-Whitney U	Z	Sig.	Mann-Whitney U	Z	Sig.
BugTracker.net	e107	4.801E+05	-18.530	0.000	3.392E+05	-25.837	0.000
	GeekLog	1.554E+06	-3.220	0.001	9.876E+05	-22.115	0.000
	phpBB2	3.838E+05	-5.624	0.000	4.043E+05	-1.859	0.063
	phpBB3	1.324E+06	-15.905	0.000	1.339E+06	-7.339	0.000
	phpMyAdmin	4.891E+06	-43.811	0.000	5.277E+06	-48.662	0.000
e107	BugTracker.net	4.801E+05	-18.530	0.000	3.392E+05	-25.837	0.000
	GeekLog	6.722E+05	-25.651	0.000	8.500E+05	-12.046	0.000
	phpBB2	1.524E+05	-19.044	0.000	1.076E+05	-22.461	0.000
	phpBB3	5.054E+05	-36.135	0.000	5.056E+05	-26.335	0.000
	phpMyAdmin	6.806E+06	-21.014	0.000	1.100E+07	-0.637	0.524
GeekLog	BugTracker.net	1.554E+06	-3.220	0.001	9.876E+05	-22.115	0.000
	e107	6.722E+05	-25.651	0.000	8.500E+05	-12.046	0.000
	phpBB2	6.110E+05	-4.368	0.000	3.563E+05	-17.941	0.000
	phpBB3	2.104E+06	-14.299	0.000	1.606E+06	-19.334	0.000
	phpMyAdmin	6.552E+06	-57.005	0.000	1.631E+07	-28.903	0.000
phpBB2	BugTracker.net	3.838E+05	-5.624	0.000	4.043E+05	-1.859	0.063
	e107	1.524E+05	-19.044	0.000	1.076E+05	-22.461	0.000
	GeekLog	6.110E+05	-4.368	0.000	3.563E+05	-17.941	0.000
	phpBB3	5.793E+05	-9.120	0.000	5.169E+05	-7.008	0.000
	phpMyAdmin	8.654E+05	-35.555	0.000	1.413E+06	-37.554	0.000
phpBB3	BugTracker.net	1.324E+06	-15.905	0.000	1.339E+06	-7.339	0.000
	e107	5.054E+05	-36.135	0.000	5.056E+05	-26.335	0.000
	GeekLog	2.104E+06	-14.299	0.000	1.606E+06	-19.334	0.000
	phpBB2	5.793E+05	-9.120	0.000	5.169E+05	-7.008	0.000
	phpMyAdmin	2.114E+06	-68.603	0.000	7.155E+06	-58.788	0.000
phpMyAdmin	BugTracker.net	4.891E+06	-43.811	0.000	5.277E+06	-48.662	0.000
	e107	6.806E+06	-21.014	0.000	1.100E+07	-0.637	0.524
	GeekLog	6.552E+06	-57.005	0.000	1.631E+07	-28.903	0.000
	phpBB2	8.654E+05	-35.555	0.000	9.314E+05	-35.102	0.000
	phpBB3	2.114E+06	-68.603	0.000	7.155E+06	-58.788	0.000

The box plots in Figures 7-1 to 7-3 indicate that each of the six test applications responded differently to the testing strategy. To explore these differences and confirm the existence of significant differences between the testing applications, the Mann-Whitney U test was applied to the test results. This analysis, summarized in Table 7-4, reveals significant differences between all of the test applications with respect to the tree similarity, and all except two with regard to context similarity. Significant

differences could not be indentified between the context similarity results for BugTracker.net and phpBB2, and e107 and phpMyAdmin. The significant differences established between applications suggest that the testing strategy is interacting individually with each of the underlying applications, and uncovering application-specific faults.

7.2.2 Tree vs. Context Similarity

The tree similarity and context similarity coefficients provide a metric of the similarity between two HTML documents—tree similarity with respect to the structure of the DOM tree, and context similarity with respect to the text encapsulated within the document. The composite similarity coefficient provides an equally weighted measure of the tree and context similarities. These metrics were chosen for the current study due to the specific structure and typical content of web documents. It is assumed that the two metrics are complementary; that is, generally when a context difference is detected, a tree difference should also be present, and vice-versa. Although strictly speaking this is not the case—a context difference is independent of tree differences—a correlation analysis can be performed upon the two metrics to establish or reject the hypothesis that the two coefficients are directly related.

To determine if a linear relationship exists between the tree and context similarity coefficients, the Pearson product-moment correlation coefficient (Pearson's r) analysis was applied to the test results. This analysis was applied to the whole testing population and to each of the applications individually, and is summarized in Table 7-5. phpBB3 was excluded from this analysis because the tree similarity coefficient was constant across all tests.

Table 7-5. Pearson Correlations: Tree vs. Context Similarity

Web Application	Correlations	
All Applications	Pearson Correlation *r*	0.767
	r^2	0.588
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	340.952
	Covariance	0.012
	N	29057
BugTracker.net	Pearson Correlation	1.000
	r^2	0.999
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	139.290
	Covariance	0.097
	N	1440
e107	Pearson Correlation	0.964
	r^2	0.929
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	19.145
	Covariance	0.019
	N	1024
GeekLog	Pearson Correlation	0.925
	r^2	0.856
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	13.243
	Covariance	0.006
	N	2232
phpBB2	Pearson Correlation	0.939
	r^2	0.881
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	0.599
	Covariance	0.001
	N	580
phpMyAdmin	Pearson Correlation	0.632
	r^2	0.399
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	157.110
	Covariance	0.007
	N	21701

Significant correlations were established in all cases. All of the correlations can be quantified as *large* ($r > 0.5$) (Cohen, 1988, 1992) indicating that a strong significant relation exists between tree and context similarity. Figures 7-4 to 7-10 provide a scatter plot of the tree vs. context similarity values and the corresponding coefficient of determination (r^2) of each significant relationship. The coefficients of determination ranged between [0.399 : 0.999]; the strongest relationship was observed amongst the testing results of BugTracker.net and the weakest amongst phpMyAdmin.

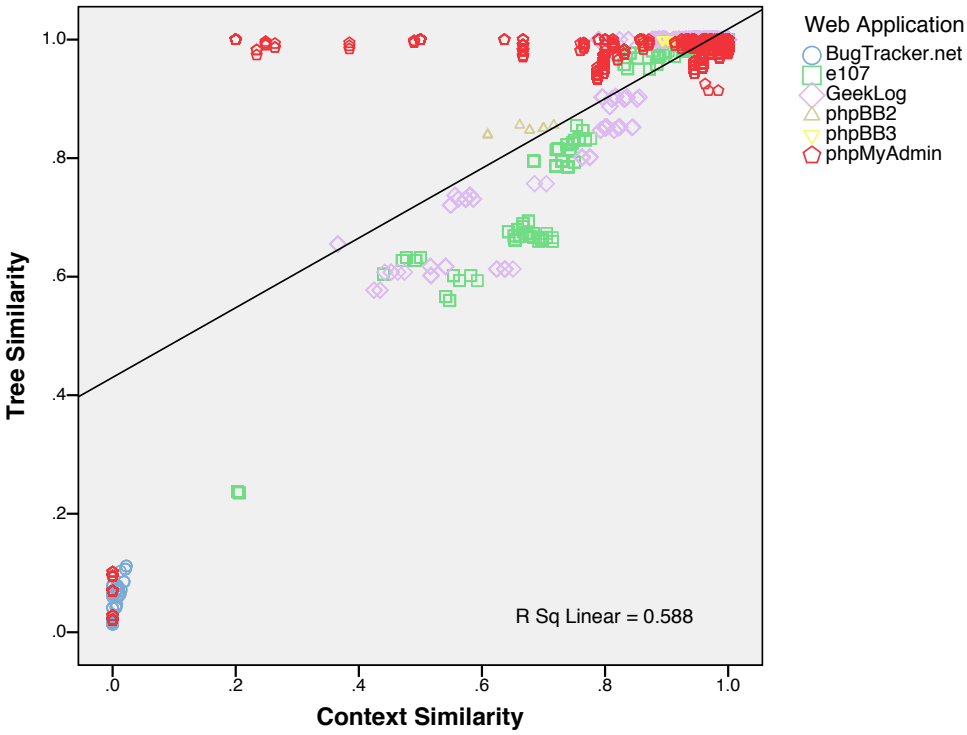


Figure 7-4. Context vs. Tree Similarity for All Applications

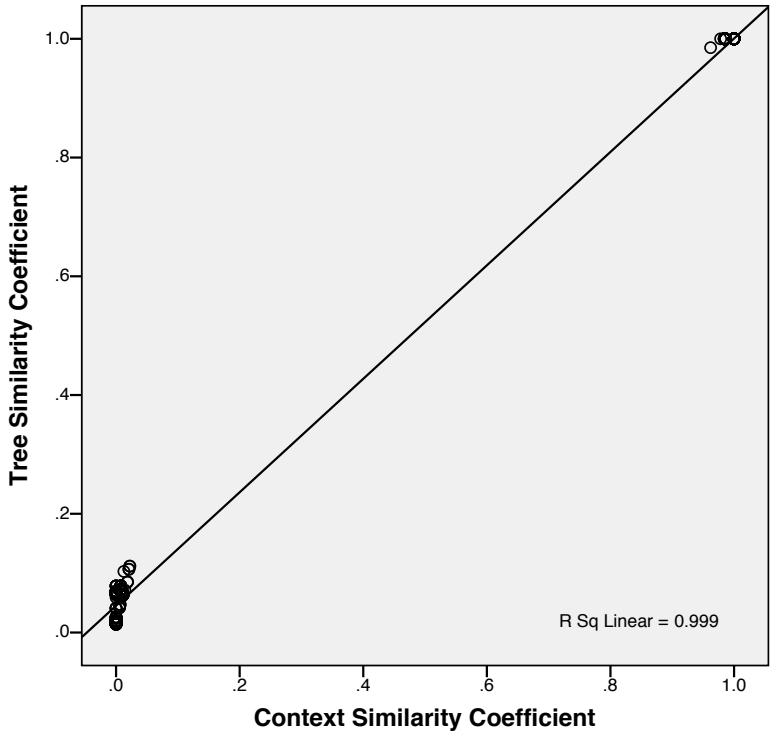


Figure 7-5. Context vs. Tree Similarity for BugTracker.net

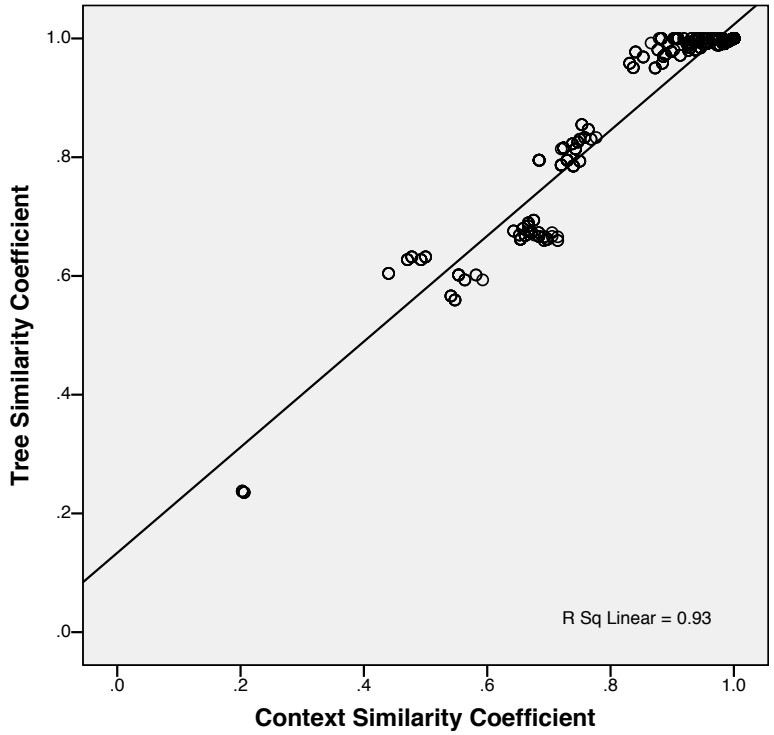


Figure 7-6. Context vs. Tree Similarity for e107

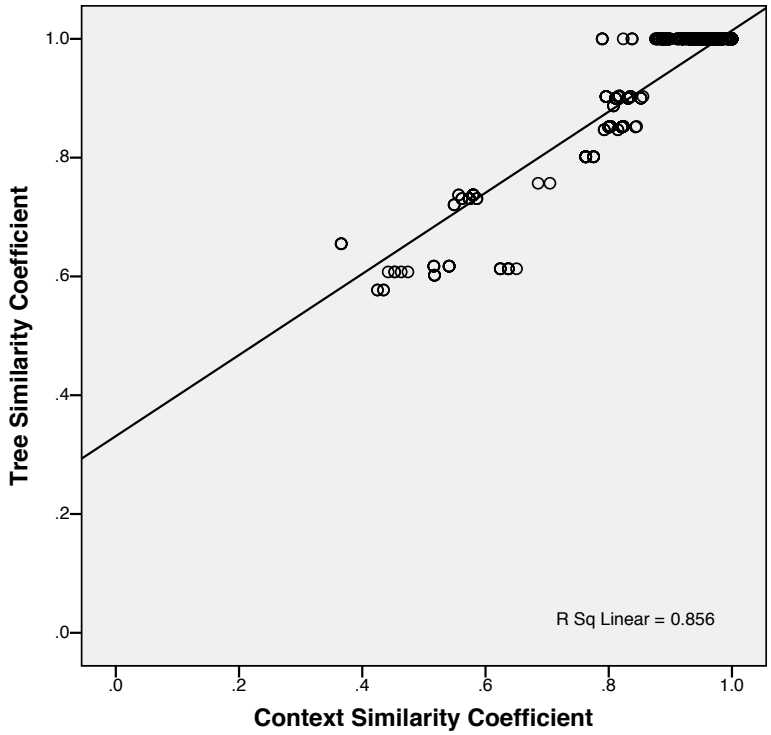


Figure 7-7. Context vs. Tree Similarity for GeekLog

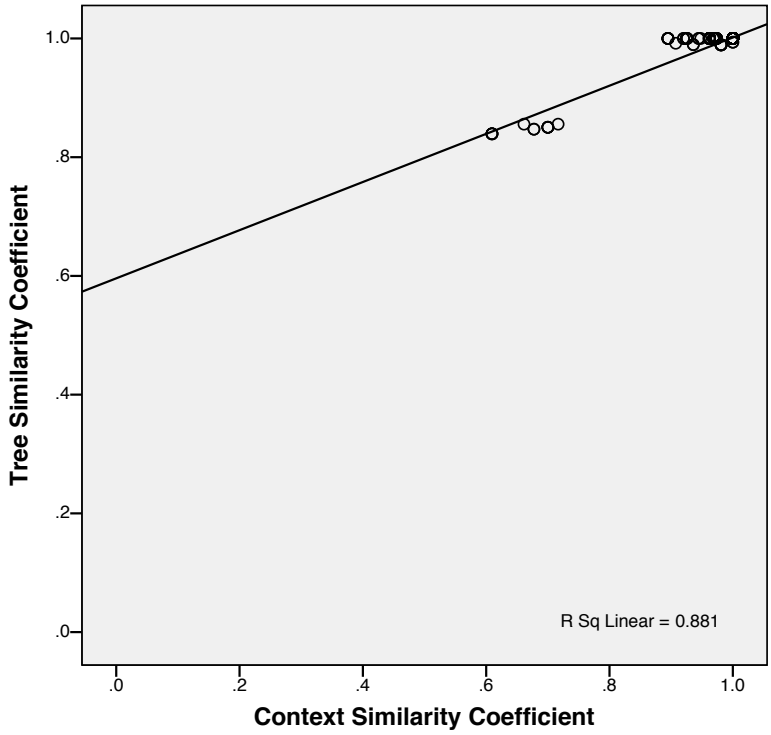


Figure 7-8. Context vs. Tree Similarity for phpBB2

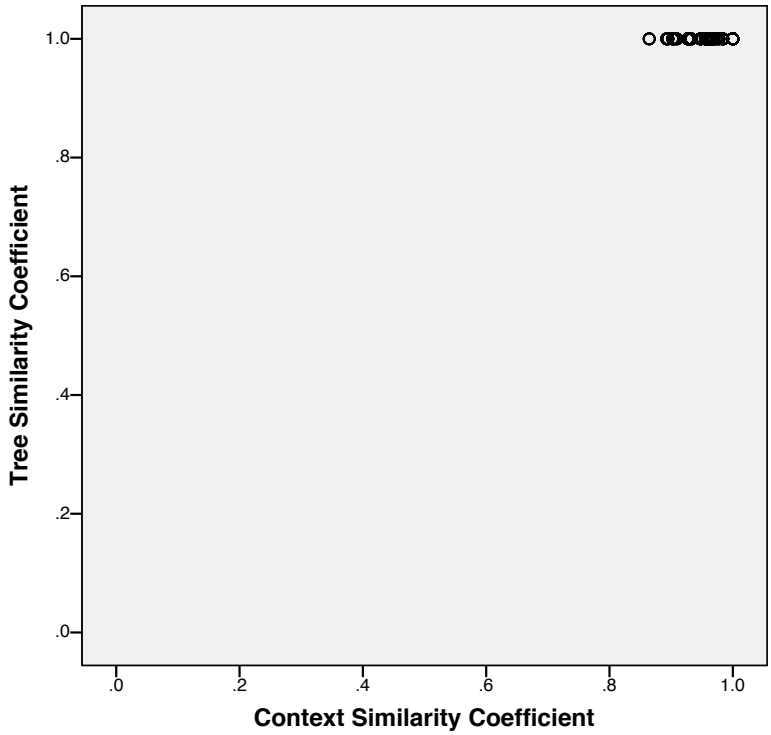


Figure 7-9. Context vs. Tree Similarity for phpBB3

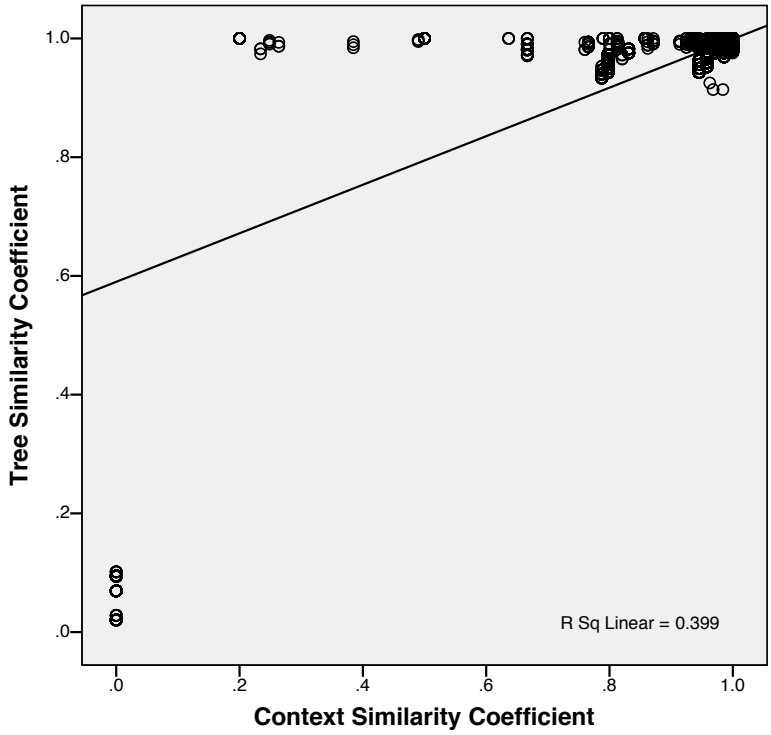


Figure 7-10. Context vs. Tree Similarity for phpMyAdmin

7.2.3 The Pass/Fail Experiment: An ROC Evaluation

It is clear from the box plots in Figures 7-4 to 7-10 and the tree vs. context similarity scatter plots presented in Figure 7-4 that the testing of the six test applications provided a spread of data, both in terms of tree and context similarity. How to classify the data remains a pertinent question—which points represent application faults? From a graphical perspective, the results presented in Figures 7-5 and 7-8 are readily amenable to classification, given the large and easily identifiable separations between clumps of data-points. In contrast, other plots such as Figures 7-9 and 7-10 do not present immediately obvious groupings and the selection of a pass/fail threshold with respect to these data points is not straightforward.

To answer the question of which data points constitute faults, a receiver operator characteristic (ROC) analysis can be applied to a smaller sub-set of the data providing a putative decision-threshold classifier for each of the testing applications (Bradley, 1997; Fawcett, 2003; Zweig & Campbell, 1993). An analysis of the area under the ROC curve (AUC) will also provide insight into the capabilities of each of the three similarity coefficients—tree, context, and composite. Essentially, the larger the AUC for a particular test, the more accurate are its decisions (Fawcett, 2003). This measure will be used to assess which similarity coefficient is the most effective in regards to detecting page differences.

For each of the applications, a sub-set of 100 test results was selected by dividing the data into two partitions based upon the value of the similarity coefficients. The result was partitioned into two sets to increase the number of values selected within the bottom half of the range of each population. Given that the median value for all of the testing results was greater than 0.95 and that the values ranged from 0 to 1, these partitions were necessary to evaluate both true positive and true negative results. Data was partitioned in two based upon the following ranges:

$$\left[\min(T) : \frac{\left(\max(T) - \min(T)\right)}{2}\right) \quad , \quad \left(\frac{\left(\max(T) - \min(T)\right)}{2} : \min(T)\right],$$

where T is the set of test results. These partitions were constructed for each of the testing applications and for both the tree and context similarity coefficients. The final subsets where constructed by the random selection of 25 test results from each of the four partitions, resulting in the selection of 100 test results on which the ROC analysis was performed.

Table 7-6. Pass/Fail Assignments Based Upon Manual Inspection

Web Application	Classification	Number of Cases
BugTracker.net	Pass	65
	Fail	35
e107	Pass	45
	Fail	55
GeekLog	Pass	50
	Fail	50
phpBB2	Pass	76
	Fail	24
phpBB3	Pass	100
	Fail	0
phpMyAdmin	Pass	26
	Fail	74

Manual inspections of each of the subsets provided the data necessary to conduct an ROC analysis on the populations—a binary classification of pass (1) or fail (0). The results of this inspection are summarized in Table 7-6. All applications, with the exception of phpBB3 provided sufficient data upon which ROC analysis could be performed. Manual analysis of phpBB3 did not yield any faults, which is congruent with the lack of data distribution present amongst the tree similarity results, and the minor fluctuations present within the context similarity [0.864:1]. Furthermore, the testing hooks, described in Appendix B.6 and Appendix C.4.2, required for the execution of phpBB3 provided sufficient system-redundancy to function unhindered despite an incomplete cookie collection. Given the popularity and maturity of the phpBB3 project, this finding is not surprising, and serves to illustrate that web applications are

being actively developed that are robust against cookie collection modifications, especially given that faults have been identified in phpBB2, a previous version of the software.

A ROC curve is the fundamental component of ROC analysis. It presents the sensitivity (true-positive rate) versus the 1-specificity (false-positive rate) under a number of varying threshold values derived from the underlying data. The ROC curve presents a cost/benefit analysis of the tradeoffs associated with each of the threshold values, facilitating the selection of an optimal decision threshold that minimizes false positive identifications while maximizing true positive assessments.

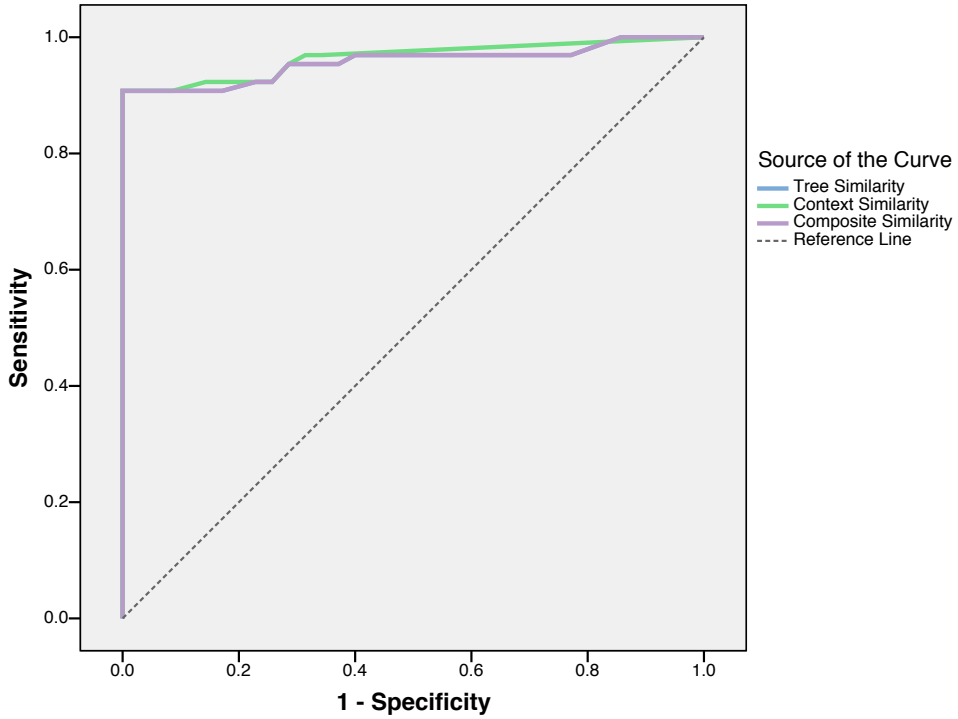


Figure 7-11. ROC Analysis: BugTracker.net

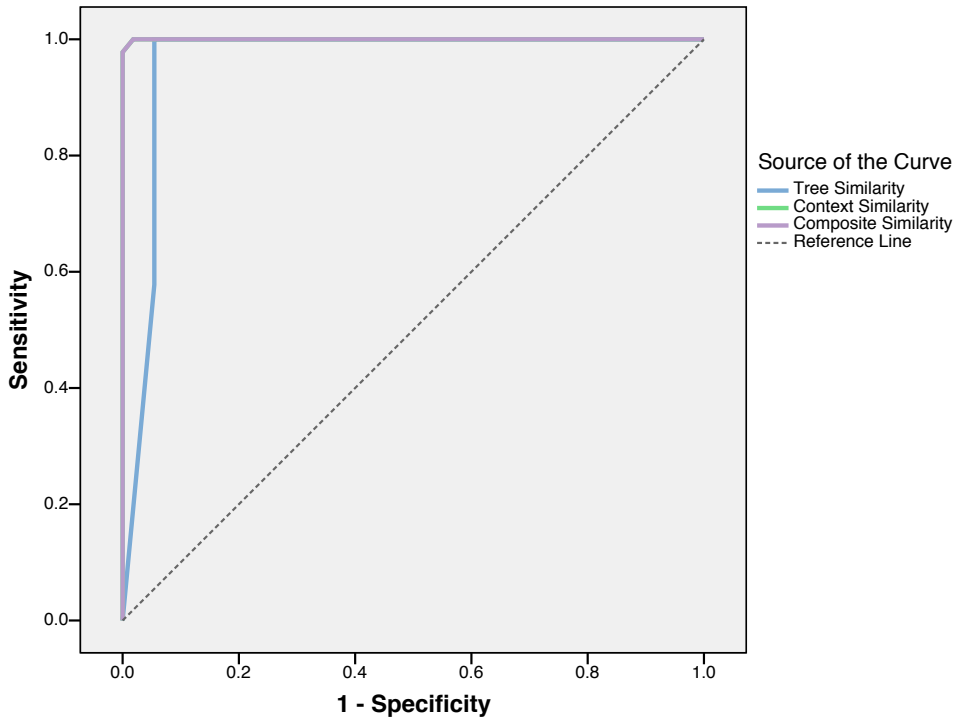


Figure 7-12. ROC Analysis: e107

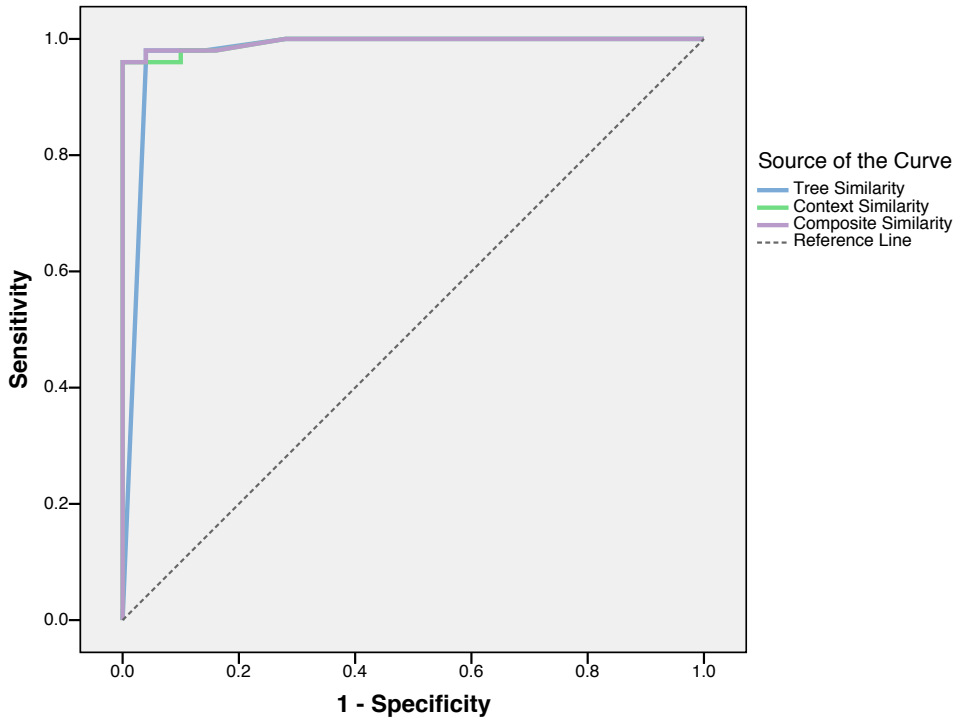


Figure 7-13. ROC Analysis: GeekLog

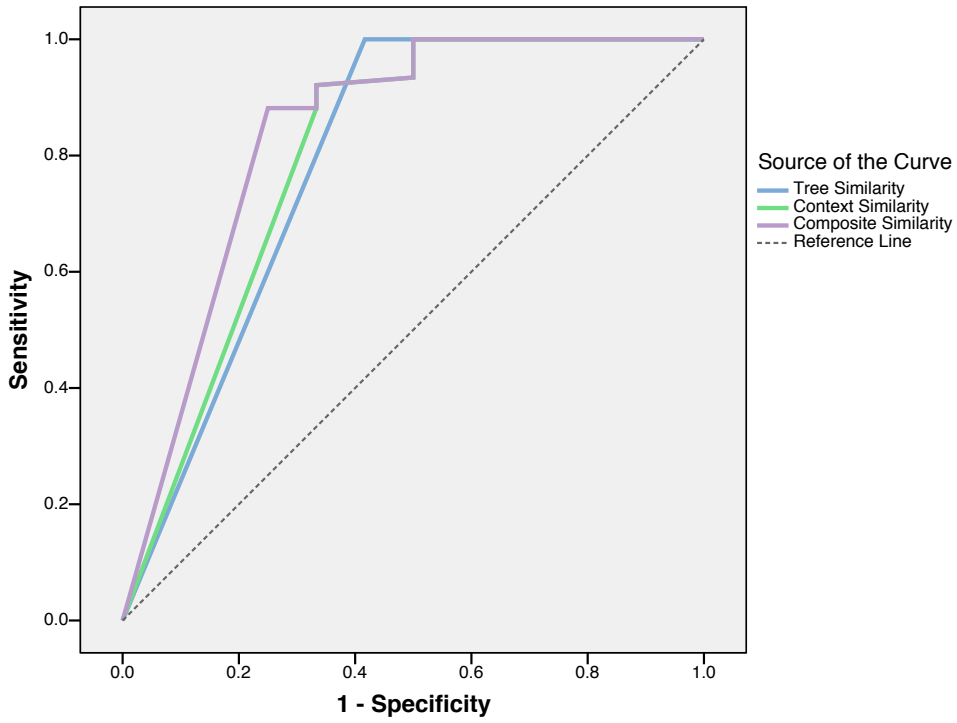


Figure 7-14. ROC Analysis: phpBB2

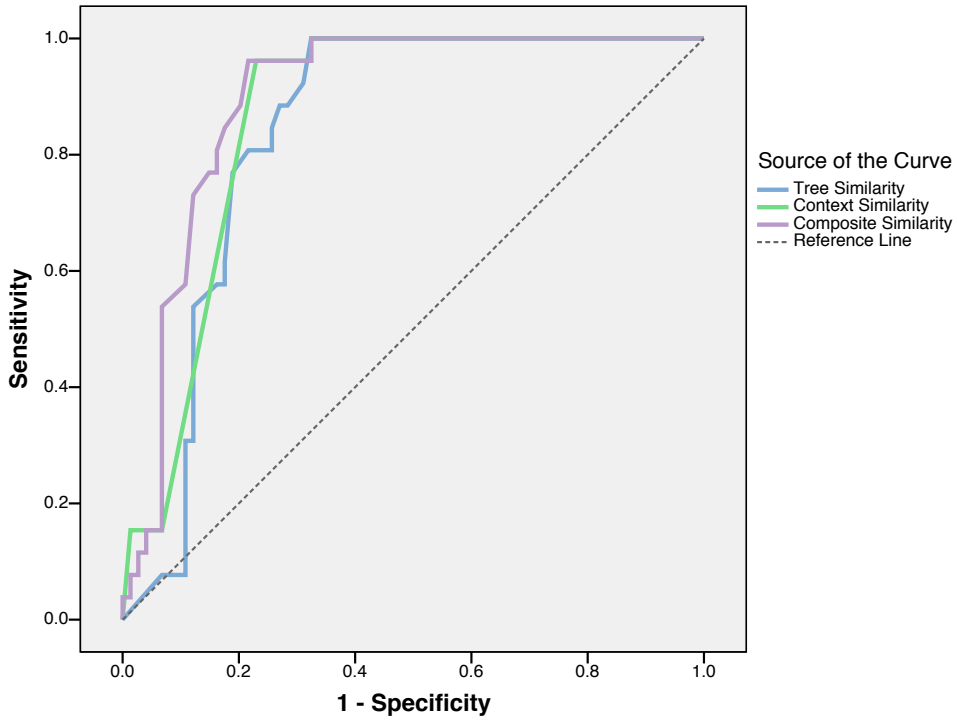


Figure 7-15. ROC Analysis: phpMyAdmin

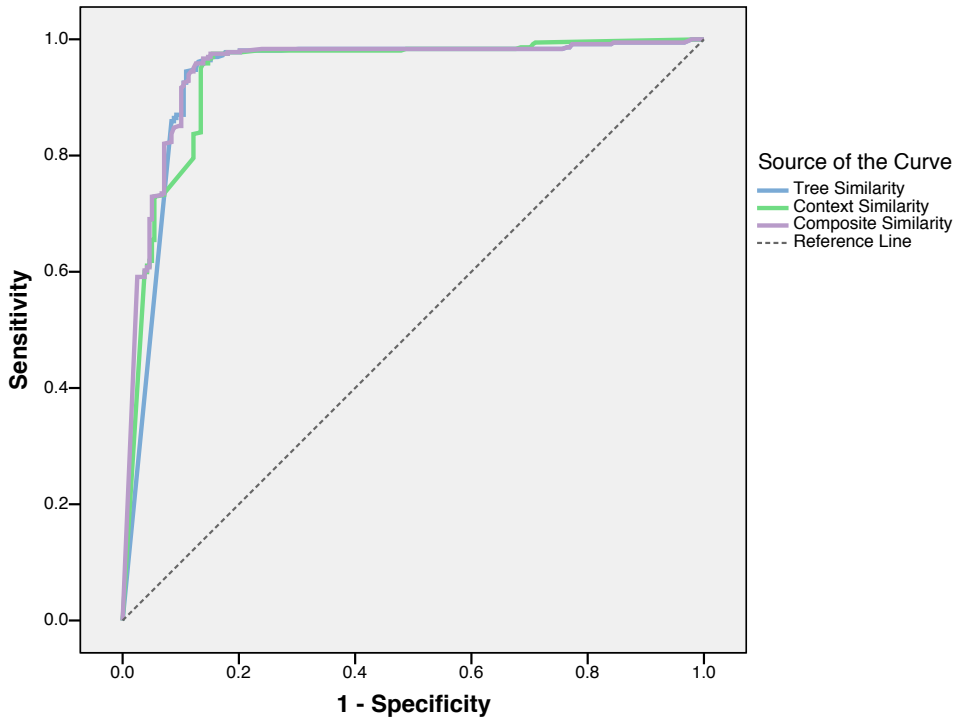


Figure 7-16. ROC Analysis: All Results

Table 7-7. ROC Area Under the Curve Summary

Web Application	Similarity Coefficient	Area	Std. Error	Asymptotic Sig.	Asymptotic 95% Confidence Interval	
					Lower Bound	Upper Bound
BugTracker.net	Tree	0.958	0.020	0.000	0.918	0.997
	Context	0.965	0.017	0.000	0.930	0.999
	Composite	0.958	0.020	0.000	0.918	0.997
e107	Tree	0.961	0.023	0.000	0.917	1.006
	Context	1.000	0.000	0.000	0.999	1.001
	Composite	1.000	0.000	0.000	0.999	1.001
GeekLog	Tree	0.976	0.017	0.000	0.943	1.009
	Context	0.994	0.005	0.000	0.983	1.004
	Composite	0.995	0.005	0.000	0.985	1.004
phpBB2	Tree	0.792	0.065	0.000	0.665	0.919
	Context	0.802	0.062	0.000	0.679	0.924
	Composite	0.838	0.056	0.000	0.728	0.948
phpMyAdmin	Tree	0.840	0.039	0.000	0.765	0.916
	Context	0.866	0.035	0.000	0.798	0.935
	Composite	0.896	0.031	0.000	0.835	0.957
All Applications	Tree	0.935	0.012	0.000	0.910	0.959
	Context	0.937	0.011	0.000	0.915	0.959
	Composite	0.947	0.010	0.000	0.927	0.968

ROC curves were generated for the five web applications in which faults were detected, and are presented in Figures 7-11 to 7-15. A composite ROC curve of all of the results is presented in Figure 7-16. Graphically, it is clear that in all cases the tree, context and composite similarity coefficients present a dramatic increase in diagnostic capability. Clearly the coefficients, as confirmed in Section 7.2.2, are linked; the ROC curves for tree and context similarity were observed to be very similar. The strongest dichotomies between true- and false-positives were observed for BugTracker.net, e107, and GeekLog, while the decision thresholds (elbows on the graphs) were not as pronounced for phpBB2 and phpMyAdmin, suggesting that dichotomies contained more overlap for these applications.

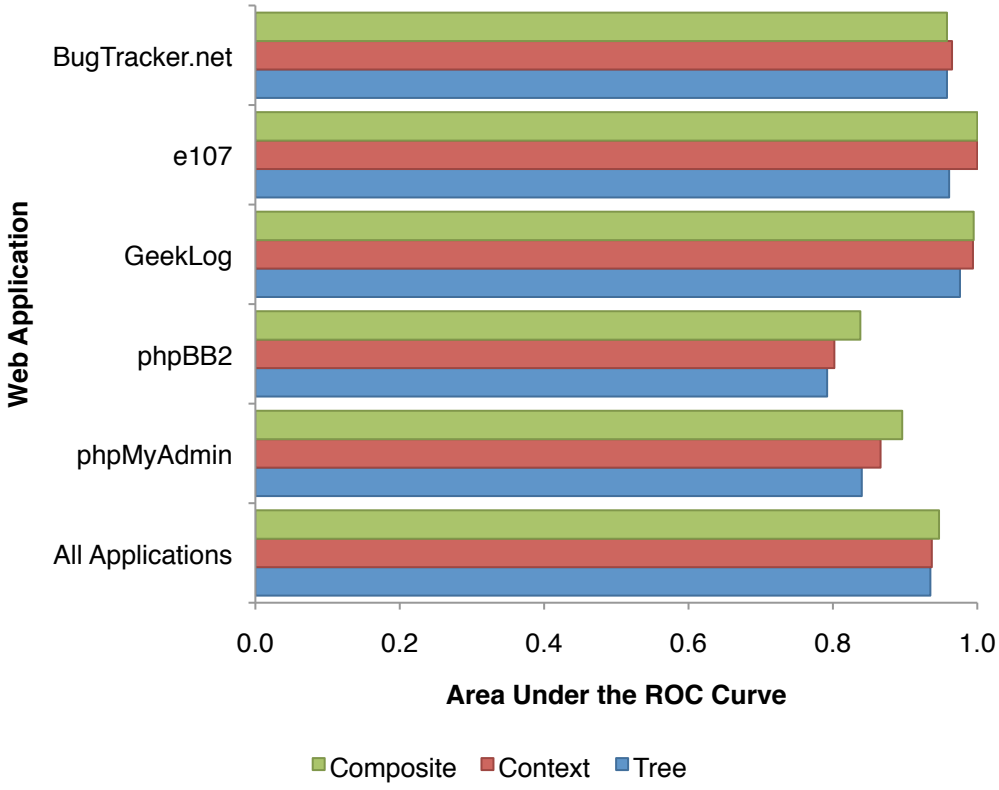


Figure 7-17. ROC Area Under the Curve Comparison

To further understand the ROC analysis, the area under the curve (AUC) was calculated for each similarity coefficient, as summarized in

Table 7-7 and presented graphically in Figure 7-17. Given the AUC values, the diagnostic ability of each metric can be assessed, whereby the higher the AUC, the more accurate the decision metric. In all cases, the tree similarity metric was smaller than the other metrics, and in all cases save one (BugTracker.net), the composite similarity metric was superior. The AUC was significant in all cases, suggesting that the selection of tree and context similarity coefficients is warranted and is an effective oracle evaluation technique.

Table 7-8. Derived Decision Threshold Classifiers

Web Application	Similarity Coefficient	Threshold Classifier	True Positive Rate	False Positive Rate
BugTracker.net	Tree	0.553	91%	0%
	Context	**0.503**	**91%**	**0%**
	Composite	0.535	91%	0%
E107	Tree	0.977	100%	6%
	Context	0.908	100%	2%
	Composite	**0.955**	**100%**	**2%**
GeekLog	Tree	0.997	100%	42%
	Context	0.879	96%	0%
	Composite	**0.942**	**96%**	**0%**
phpBB2	Tree	0.997	100%	42%
	Context	0.965	92%	33%
	Composite	**0.998**	**88%**	**25%**
phpMyAdmin	Tree	0.976	77%	19%
	Context	0.952	96%	23%
	Composite	**0.959**	**96%**	**22%**
All Programs	Tree	0.983	95%	11%
	Context	0.915	95%	13%
	Composite	**0.960**	**94%**	**11%**

Accepting the findings of the AUC analysis, Table 7-8 presents a summary of the derived decision threshold classifiers for each of the test applications with the selected threshold classifier indicated in **bold**. Based upon the analysis of the 600 test results from the six web applications, the composite coefficient can be used to classify a difference between two web pages with a threshold of 0.960. Although this value can

be used as a general guideline, it is not definitive, and an application specific ROC analysis is required to support the selection of an effective decision threshold in practice.

7.2.4 Testing Results: Seeds and Generated Testing Vectors

The values for decision classification derived in the ROC analysis can be used to evaluate the final test result populations, providing an indication of the number of defects uncovered using the cookie collection testing strategy. It must be stated that the decision classifiers have a certain amount of error associated with each classification, and these results only provide an estimate of the number of faults within each application; manual inspection and classification of each test case would be necessary for a perfect measurement.

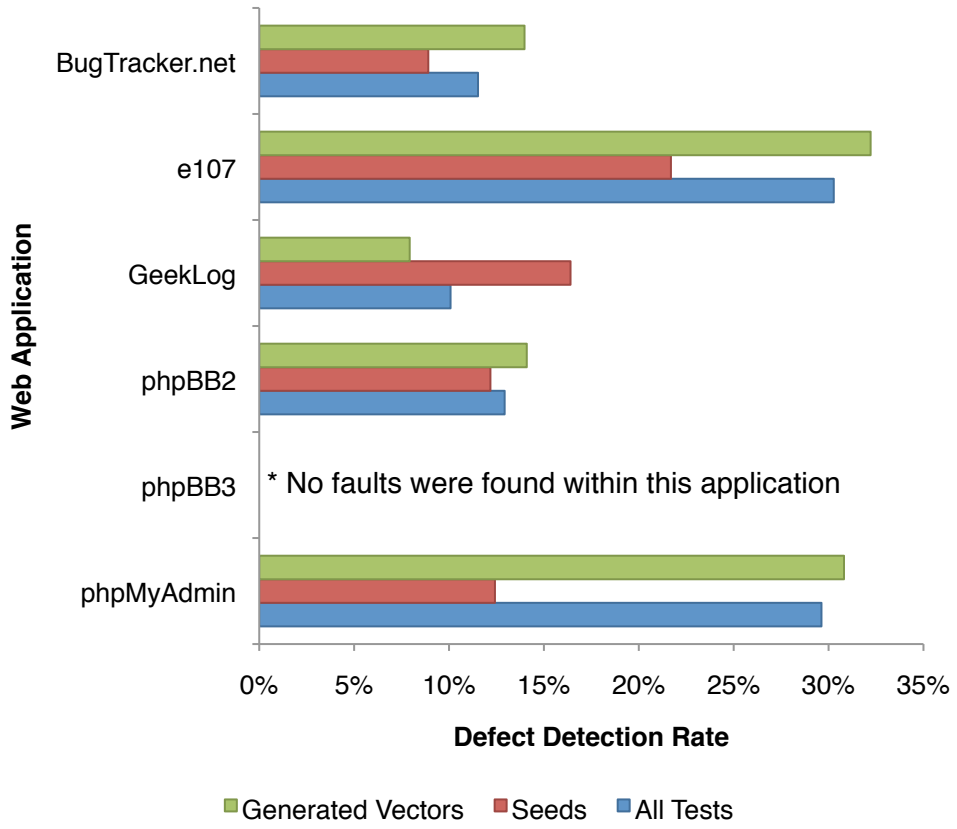


Figure 7-18. Defect Detection Rates for All, Seeding, and Generated Vectors

Table 7-9. Testing Results for All, Seeding, and Generated Vectors

Web Application	All Tests		Seed Vectors		Generated Vectors	
	Tests	Defects	Tests	Defects	Tests	Defects
BugTracker.net	1,440	166	696	62	744	104
e107	1,024	310	189	41	835	269
GeekLog	2,232	225	567	93	1,665	132
phpBB2	580	75	353	43	227	32
phpBB3	2,080	0	401	0	1,679	0
phpMyAdmin	21,701	6,429	1,401	174	20,300	6,255

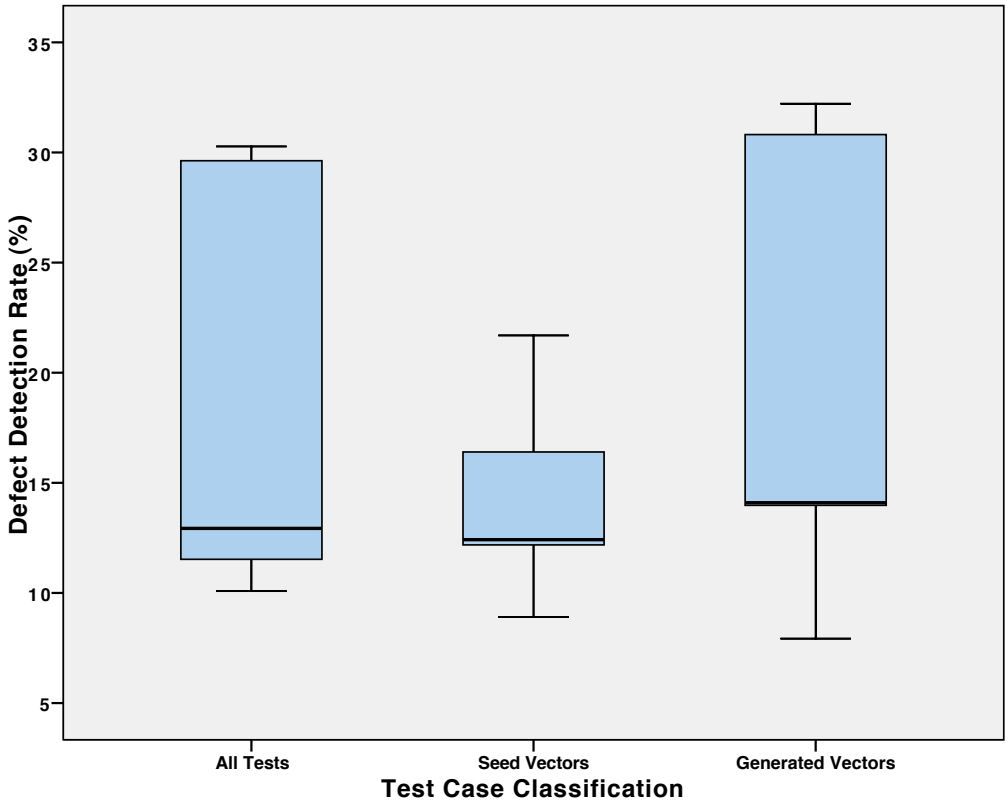


Figure 7-19. Box Plots of Fault Detection Rates for All, Seeding, and Generated Vectors

A summary of the number of test cases triggering a fault is presented in Table 7-9 and Figure 7-18 from three perspectives: all test vectors, seeding vectors, and generated vectors. It is clear from the table that the testing strategy was effective in triggering faults within five of the six web applications tested. In all of the applications except GeekLog, the fault detection rate (the percentage of test cases that triggered a fault) was higher amongst generated vectors than seeding vectors. The fault

detection rate ranged between 10% – 30% for all tests, 9% – 21% amongst seeding vectors, and 8% – 32% for generated test vectors, as shown in Figure 7-19 (excluding phpBB3 for which no defects were found). The highest fault detection rate occurred amongst generated vectors, suggesting that seeding vector testing alone does not accomplish adequate cookie collection testing.

Table 7-10. Mann-Whitney U Test: Seeding vs. Generated Test Vectors

Web Application		Tree Similarity	Context Similarity	Composite Similarity
BugTracker.net	Mann-Whitney U	246390	236078.5	235962.5
	Wilcoxon W	523530	513218.5	513102.5
	Z	-2.856	-4.701	-4.722
	Asymp. Sig. (2-tailed)	0.004	0.000	0.000
e107	Mann-Whitney U	64929.5	45874	46349
	Wilcoxon W	413959.5	394904	395379
	Z	-4.029	-9.065	-8.934
	Asymp. Sig. (2-tailed)	0.000	0.000	0.000
GeekLog	Mann-Whitney U	430957	332256	332369.5
	Wilcoxon W	591985	1.72E+06	1.72E+06
	Z	-6.131	-10.947	-10.938
	Asymp. Sig. (2-tailed)	0.000	0.000	0.000
phpBB2	Mann-Whitney U	39175.5	39326.5	39302
	Wilcoxon W	101656.5	65204.5	65180
	Z	-1.336	-0.651	-0.665
	Asymp. Sig. (2-tailed)	0.181	0.515	0.506
phpMyAdmin	Mann-Whitney U	5.49E+06	1.04E+07	6.62E+06
	Wilcoxon W	2.12E+08	2.16E+08	2.13E+08
	Z	-38.54	-20.369	-33.547
	Asymp. Sig. (2-tailed)	0.000	0.000	0.000

To further understand the relationship between seeding and generated vectors, a Mann-Whitney U test was preformed between the two populations for each application demonstrated to contain faults. This analysis, summarized in Table 7-10, reveals a significant difference between the seed and generated test results for all applications with the exception of phpBB2. These differences suggest that the evolutionary adaptive random testing vectors are providing unique test cases that are exercising the application in ways that the seeding vectors are not. Given this result, it is recommended that when performing cookie collection

testing, seeding vectors alone are insufficient, and adaptive-random vectors should be incorporated into an overall cookie testing strategy.

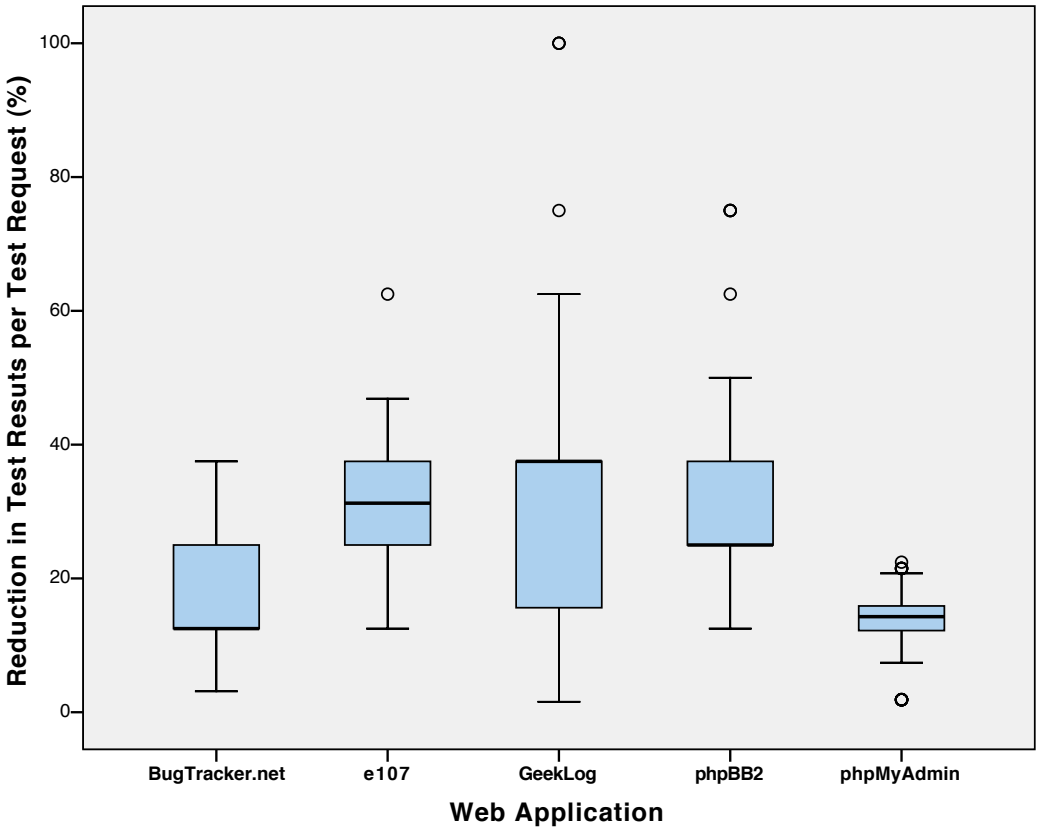


Figure 7-20. Box Plots of Test Result Reductions via Distinct Results

7.2.5 Distinct Defects — A Proxy for Defect Identification

As is customary for automated testing solutions, cookie collection testing has been demonstrated to produce large pools of testing data from which a tester has the difficult and costly task of determining the actual number of defects within a system. Given the repetitious nature of the testing strategy, it is expected that multiple test cases will trigger the same underlying defect. Techniques are therefore required to subsequently reduce the testing results into a manageable size from which conclusions about the system can be drawn. To provide this level of granularity, the grouping of test results on the basis of distinct tree and context similarity pairs is proposed. This metric, estimated by the number of distinct composite similarity values, provides a basis through which distinct test

results are identified. The use of the composite similarity coefficient for defect identification was found to dramatically reduce the workload for analyzing test results, as summarized in Table 7-11 and Figure 7-20.

Table 7-11. Test Result Reduction via Distinct Results

Statistic	BugTracker.net	e107	GeekLog	phpBB2	phpMyAdmin
Mean	16.5%	31.0%	33.6%	30.7%	13.7%
Median	12.5%	31.3%	37.5%	25.0%	14.3%
Minimum	3.1%	12.5%	1.6%	12.5%	1.9%
Maximum	37.5%	62.5%	100.0%	75.0%	22.4%

Table 7-12. Distinct Defects per Test Request

Web Application	Testing Vectors	Distinct Defects			
		Median	Min	Max	Total
BugTracker.net	All	0	0	2	57
	Seeding	0	0	2	40
	Generated	0	0	2	47
e107	All	1	0	8	92
	Seeding	0	0	3	40
	Generated	1	0	7	86
GeekLog	All	0	0	6	83
	Seeding	0	0	3	55
	Generated	0	0	4	42
phpBB2	All	0	0	4	52
	Seeding	0	0	2	38
	Generated	0	0	3	27
phpMyAdmin	All	5	0	16	1210
	Seeding	0	0	3	117
	Generated	5	0	16	1193

A summary of the number of distinct defects found per test request is provided in Table 7-12. The threshold values determined in Section 7.2.3 were used to classify each defect. The table presents a final tally of the number of distinct defects revealed within each application. To augment this analysis, Table 7-13 provides a frequency analysis of the number of distinct defects triggered per testing request. Further analysis of the data reveals a disparity between the faults uncovered by seeding vectors and generating vectors. In two of the test applications, GeekLog and phpBB2, the number of distinct defects detected for seeding vectors was larger than that of generated vectors. The opposite was true in the

case of Bugtracker.net, e107, and phpMyAdmin, where the generated vectors revealed a larger number of distinct defects.

Table 7-13. Frequency Analysis of Distinct Defects per Test Request

Web Application	Number of Distinct Failures	All Test Vectors		Seeding Vectors		Generated Vectors	
		Frequency	%	Frequency	%	Frequency	%
BugTracker.net	0	84	63.6%	99	75.0%	92	69.7%
	1	39	29.5%	26	19.7%	33	25.0%
	2	9	6.8%	7	5.3%	7	5.3%
	Total	132	100.0%	132	100.0%	132	100.0%
e107	0	27	47.4%	34	59.6%	27	47.4%
	1	7	12.3%	8	14.0%	7	12.3%
	2	8	14.0%	13	22.8%	8	14.0%
	3	5	8.8%	2	3.5%	5	8.8%
	4	4	7.0%	0	0.0%	6	10.5%
	5	2	3.5%	0	0.0%	2	3.5%
	6	2	3.5%	0	0.0%	0	0.0%
	7	0	0.0%	0	0.0%	2	3.5%
	8	2	3.5%	0	0.0%	0	0.0%
	Total	57	100.0%	57	100.0%	57	100.0%
GeekLog	0	127	75.1%	128	75.7%	154	91.1%
	1	26	15.4%	30	17.8%	0	0.0%
	2	5	3.0%	8	4.7%	7	4.1%
	3	4	2.4%	3	1.8%	4	2.4%
	4	2	1.2%	0	0.0%	4	2.4%
	5	3	1.8%	0	0.0%	0	0.0%
	6	2	1.2%	0	0.0%	0	0.0%
	Total	169	100.0%	169	100.0%	169	100.0%
phpBB2	0	71	62.3%	77	67.5%	91	79.8%
	1	36	31.6%	36	31.6%	20	17.5%
	2	6	5.3%	1	0.9%	2	1.8%
	3	0	0.0%	0	0.0%	1	0.9%
	4	1	0.9%	0	0.0%	0	0.0%
	Total	114	100.0%	114	100.0%	114	100.0%
phpMyAdmin	0	14	6.9%	121	59.6%	14	6.9%
	1	20	9.9%	49	24.1%	20	9.9%
	2	1	0.5%	31	15.3%	1	0.5%
	3	2	1.0%	2	1.0%	2	1.0%
	4	29	14.3%	0	0.0%	34	16.7%
	5	45	22.2%	0	0.0%	44	21.7%
	6	38	18.7%	0	0.0%	36	17.7%
	7	8	3.9%	0	0.0%	7	3.4%
	8	4	2.0%	0	0.0%	7	3.4%
	9	7	3.4%	0	0.0%	4	2.0%
	10	1	0.5%	0	0.0%	0	0.0%
	11	4	2.0%	0	0.0%	4	2.0%
	12	6	3.0%	0	0.0%	6	3.0%
	13	9	4.4%	0	0.0%	9	4.4%
	14	8	3.9%	0	0.0%	8	3.9%
	15	5	2.5%	0	0.0%	5	2.5%
	16	2	1.0%	0	0.0%	2	1.0%
	Total	203	100.0%	203	100.0%	203	100.0%

To further classify the ability of seeding and generated vectors to uncover distinct defects, a case-by-case analysis was performed and is summarized in Table 7-14. Generated vectors were found to trigger distinct defects not uncovered by seeding vectors in all of the applications.

Conversely, in BugTracker.net, GeekLog, and phpBB2, the seeding vectors were observed to trigger distinct faults that were not uncovered by the generated vectors. The largest disparity between faults triggered by generated and seeding vectors was observed in phpMyAdmin, where generated vectors found distinct defects in 83.3% of test requests, while seeding vectors did not uncover any defects not detected by the generated vectors. The smallest disparity was observed within GeekLog, where generated vectors were found to trigger distinct faults in only 5.9% of test requests, and seeding vectors in 16.6% of test requests. BugTracker.net was found to have the largest overlap between seeding and generated vectors, each uncovering the same number of distinct defects in 80.5% of test requests.

Table 7-14. Distinct Defect Detection Seeding vs. Generated Test Vectors

Classification	BugTracker.net		e107		GeekLog		phpBB2		phpMyAdmin	
Generated > Seeding	17	12.8%	21	36.8%	10	5.9%	8	7.0%	169	83.3%
Generated = Seeding	107	80.5%	36	63.2%	131	77.5%	86	75.4 %	34	16.7%
Generated < Seeding	9	6.8%	0	0.0%	28	16.6%	20	17.5%	0	0.0%

7.3 Summary of Results and Key Findings

To summarize the results and key findings of the Chapter, this section will present a brief answer to each of the research questions originally posed in Section 7.1.1.

Q1. *Can cookie collection testing reveal defects in web applications?*
In short, *yes*. It has been demonstrated, through the manual inspection of 100 test cases per web application, that the testing strategy triggered faults in the web applications. This initial analysis lead to the creation of application-specific decision threshold classifiers which were in turn applied to the testing results and used to make a final pronouncement of the estimated defect detection rate and the number of distinct defects detected per application. The testing strategy has not only been verified in

its ability to detect faults, but has also been shown to have a substantial defect triggering rate amongst five of the six test applications.

Q2. *What is the relationship that exists between faults triggered by seeding and generated vectors? Are the two complementary, or does one supersede the other?*

Seeding and generated vectors were found to trigger faults in five of the six test applications. In four of the five applications that were demonstrated to contain faults, the fault detection rate was higher amongst the generated vectors. In terms of the number of distinct defects detected by seeding and generated vectors, it has been demonstrated that the defects triggered by seeding and generated vectors are not identical. Furthermore, in Section 7.2.4, significant differences were established between the seeding and generated vector test results in four of the five applications, providing the strongest evidence that seeding vectors and generated vectors should both be incorporated in an overall cookie collection testing strategy.

Q3. *How do web applications respond to cookie collection testing? Are the responses uniform or application-specific?*

It has been demonstrated, through the Mann-Whitney U tests presented in Section 7.2.1, that the test result populations for each of the six applications are significantly different from one another. This is further validated by the differences in ROC curves derived for each of the test applications. Finally, the fault detection rates presented for each application further suggest that the interactions between cookie collection testing and the software under test are unique, and application-specific.

Q4. *Does a relationship exist between tree and context similarity coefficients?*

A large significant relationship (r>0.5) was found to exist between tree and context similarity metrics for each of the six applications under test in Section 7.2.2. This relationship provides the strongest evidence that a

large linear relationship exists between the tree and context differences when considering page-differences in web applications.

Q5. *Are the tree similarity and context similarity coefficients useful at detecting web application defects? Is one superior or is a composite metric most effective?*

The results of the ROC analysis performed in Section 7.2.3, specifically the area under the curve analysis, demonstrated that all three metrics are significant diagnostic tests. The context and composite similarity coefficients were found to perform better than the tree similarity, and the composite similarity was found to be superior to both tree and context measures in all but one instance.

Chapter 8

Conclusions and Recommendations

The objective of this research project was to investigate cookie usage within modern web applications and to provide an effective cookie-specific testing strategy. Towards this end, an extensive Internet survey was conducted and a novel testing theory—cookie collection testing—has been formulated in response. Furthermore, in support of cookie collection testing, the state-of-the-art of adaptive random testing has been extended through the incorporation of genetic algorithms and anti random techniques. Finally, automated cookie collection testing has been demonstrated to be an effective testing strategy for web applications. This chapter will provide a brief overview of the major contributions of this thesis and suggest avenues for future research.

8.1 Summary of Key Contributions by Chapter

8.1.1 A Precise Cookie Definition

A precise grammatical definition of a cookie was formulated in this thesis. This definition is the result of an amalgamation of the available cookie specifications and browser-specific addendums—it represents the most complete up-to-date definition of a cookie within the literature.

8.1.2 Cookie Deployment Survey

An extensive survey of the most popular 100,000 Internet sites was conducted to understand the deployment of cookies across the Internet.

The results of the investigation were analyzed from a number of distinct perspectives providing insights into the deployment of cookie technology. Cookie deployment was found to be approaching universal levels, confirming the need for relevant Web and Software Engineering processes—specifically testing strategies that actively consider cookies. The semi-automated investigation demonstrated that cookies were deployed by more than two-thirds of the sites studied. The investigation specifically examined the use of first-party, third-party, sessional and persistent cookies within web applications, and identified the presence of P3P policies and dynamic web technologies as major predictors of cookie usage. A number of real world examples confirmed the need for comprehensive testing strategies for web-based applications.

8.1.3 Cookie Usage Amongst Nations

The use of cookies and related web technologies with respect to their country of origin was studied by the augmenting the results obtained from the cookie deployment study with the geographic location of each site. A number of significant relationships were established between the origin of the web application and cookie deployment. Cookie usage amongst five popular dynamic web application frameworks was analyzed, providing a per-country breakdown of platform adoption, and a significant relationship was established between dynamic web technologies and first-party and sessional cookies. The prevalence of vendor-specific third-party technologies both globally and within specific countries was studied. Although global leaders emerged, a number of country-specific market leaders were discovered, suggesting that country-specific niche technologies are competing with the globally dominant technologies within specific markets. Finally, a large association was identified between third-party persistent cookie usage and a country's e-business environment—the strongest evidence that cookies form an integral part of the global e-commerce environment.

8.1.4 Cookie Collection Testing

In response to the cookie deployment study, a cookie-specific web testing strategy was formulated. The cookie collection testing strategy was built upon anti random testing methodologies, with consideration of the results obtained from the cookie deployment study. The collection of cookies maintained within a user-agent were explored in light of the anti random test suite reduction techniques and the grammatical definition of a cookie, culminating in the definition of seeding test vectors as the basis for a scalable test suite. The automation of the testing methodology was outlined and the definition of test oracles and evaluation criterion was discussed. The cookie collection testing strategy met the intended goal of verifying web application robustness against the modification—intentional or otherwise—of the cookies within a browsing environment.

8.1.5 Evolutionary Adaptive Random Testing

The cookie collection testing strategy envisaged anti random testing as the basis from which cookie collection testing could be applied. However, the application of anti random testing to cookie collection testing was found to be computationally infeasible for many of the applications surveyed within the cookie deployment study. In response to this reality, the novel application of an evolutionary search algorithm to the problem of adaptive random testing was proposed as a solution. An extensive simulation study was undertaken to evaluate the novel evolutionary approach against the current state-of-the-art within the adaptive random literature—fixed size candidate set, restricted random testing, quasi-random testing, and random testing. The evolutionary approach was demonstrated to be superior to the other methodologies amongst block pattern simulations. For fault patterns with increased complexity, the evolutionary approach was shown to be comparable to the current adaptive random testing strategies, and showed a modest improvement over quasi-random and

random testing techniques. A comparison of the asymptotic and empirical runtimes of the evolutionary search algorithm to the other testing approaches was also presented; the comparison provided further evidence that the application of an evolutionary search algorithm would be feasible, and within the same order of time complexity as the other adaptive random testing approaches. Finally, the application of evolutionary adaptive random testing to the problem of cookie collection testing was presented, finalizing the automation of the web testing technique.

8.1.6 Empirical Evaluations

An empirical investigation of cookie collection testing was undertaken to evaluate the effectiveness of the testing methodology. Six open source web applications were chosen as test subjects and an application-specific cookie collection testing suite was developed for each application. Cookie collection testing was demonstrated to reveal defects within five of the six test subjects, and was found to have a substantial fault-triggering rate. Furthermore, the testing strategy was demonstrated to interact with each test application independently, indicating that the testing strategy was able to access the underlying application, not just the technological platform upon which the application was implemented. Both seeding and evolutionary adaptive random generated vectors were found to be useful in triggering defects. Neither of the two sets of vectors—seeding and generated—were found to supersede the other, demonstrating a synergetic relationship between seeding and generated vectors within cookie collection testing. Finally, a large significant relationship was established between structural (tree) and content (context) similarity measures of web application responses. Due to this relationship, the composite similarity metric (a combination of the tree and context measures) was observed to be superior for the detection of a fault between two web application responses.

8.2 Recommendations for Future Research

The work presented provides a number of avenues for future research within two distinct categories: web application testing and the more generalized field of adaptive random testing.

8.2.1 Testing Web Applications

The major contribution of this thesis was the definition of a cookie-specific web application testing strategy. A number of subsequent avenues of research exist within the web testing community in relation to this work. Cookie collection testing is a viable web testing strategy given the nature of web applications, and the relationship that exists between client and server—more specifically the relationship between user-agent and server. Within this unique relationship, internal state information (cookies) are stored and implemented within the user-accessible browsing agent. This relation differs significantly from traditional client-server relationship in which the internal state exists within an application itself and is not easily accessible to the end-user. Although internal state values can be gleaned and modified through the use of reverse engineering techniques and code injection, these values have an inherent level of privacy afforded by the execution of compiled and/or obfuscated code. This level of privacy is simply not present within a web environment, and although web obfuscation techniques exist for web technologies, it is trivial to intercept the values passed between client and server. This inherently insecure environment is a reality of the client layer of web applications, and as demonstrated in the empirical evaluation, changes to the collection of cookies stored therein can trigger application defects. Essentially, cookie collection testing describes a methodology for the modification of internal state values within an insecure operating environment—a novel idea with applications within a wide range of problem domains.

Asynchronous Java and XML (AJAX) applications epitomize the execution of application components within the client layer. These

applications off-load large portions of the GUI onto the client layer in an effort to present a stateful event-driven GUI to the end-user. These applications rely on client-layer code to drive the majority of interactions with a backend server. These interactions ultimately result in HTTP requests being sent to the server which can be intercepted and modified, presenting the server with unpredictable requests. This paradigm, similar to that of a typical HTTP request, is a prime candidate for future research in a similar vein to that of cookie collection testing. How can a tester modify the internal state of an AJAX application? Are there generalities, similar to the rules defined for seeding vectors, that can be applied to a broad range of AJAX applications? These questions, and more importantly the answers to these questions, are of keen interest to future developments within the web engineering field.

The field of web services, and more generally service oriented computing, presents further opportunities for the testing of insecure internal state values. Web applications utilizing web services need to be verified against potentially damaging modifications of internal state values. Although these requests are often handled within the server layer (that is, the backend server fires a request to an independent web service), the process relies heavily on an independent component, one that could potentially introduce errors within the internal state of an application. Although a web service is assumed to act in accordance with its WSDL specification, this assumption is not necessarily verified, and any application must be robust against errors introduced within web service responses. Conversely, web services must also be able to handle a wide variety of inputs from an HTTP request. A WSDL specification cannot fully encapsulate complex relationships existing between variables specified as inputs into a web service. While these relationships may be specifically described within the documentation for a particular service, it is not wise to assume input compliance. This interface between components within the service oriented computing paradigm requires

specific testing considerations stemming from the investigations of the insecure internal state values of web applications. This research has demonstrated defects within currently deployed web applications with regard to modifications of the internal state values; do the same type of defects exist within the service-oriented paradigm? The academic community should seek the answer to this question, and any testing practitioner for a service-oriented system must verify the robustness of each computational component within the system.

8.2.2 Adaptive Random Testing

A number of avenues of future research exist in the field of adaptive random testing. Specifically in relation to evolutionary adaptive random testing, future work should focus on identification of the specific genetic algorithm parameters that can increase testing effectiveness. Although the maximum theoretical increase in testing effectiveness is bounded (T. Y. Chen & Merkel, 2008), it is clear from the simulation study that gains in testing effectiveness beyond those established in the literature are still possible.

The fitness function is another component that requires further study; specifically, the development of a more sophisticated fitness function that could further increase testing effectiveness amongst more complex failure regions is of primary interest. The observed tendency of evolutionary adaptive random testing to initially select boundary values suggests that the fitness function is not ideal, especially when detecting point pattern failure regions. There may exist more appropriate fitness functions that can be applied within both the fixed size candidate set, and evolutionary random testing. Future work within this area should include discussions on the selection of an appropriate fitness function, and possibly functions tailored for specific fault-patterns and input domains.

Finally, although there have been empirical studies—including Chapter 7 of this thesis—demonstrating the effectiveness of adaptive

random testing, further work is required to validate the use of these methodologies. Given the previous use of genetic algorithms for the selection of test inputs based upon a white-box testing criterion (Harman et al., 2004; Michael et al., 2001; Xiao et al., 2007), future research could focus upon the synergies between the varying approaches. Recent advances in genetic network programming are also of keen interest within the area of software testing. Genetic network programming has been applied within other domains in which complex relationships exist between a large number of varying factors, such as the portfolio optimization problem (Y. Chen, Ohkawa, Mabu, Shimada, & Hirasawa, 2009). Given the complex relationships that often exist within the software input domain, and the demonstrated suitability of search-based heuristics with respect to adaptive random testing, adaptations of genetic network programming could prove to be a useful tool. Such an adaptation could potentially provide increased effectiveness, leveraging both the current test suite, and the relationships that exist between input values.

Bibliography

Agrawal, V. D. (1978). When to Use Random Testing. *IEEE Transactions on Computers, C-27*(11), 1054-1055.

Alalfi, M., Cordy, J. R., & Dean, T. R. (2009). Modeling Methods for Web Application Verification and Testing: State of the Art. *Software Testing, Verification and Reliability, 19*(4), 265–296.

Alexa Internet Inc. (2006a). About the Alexa Traffic Rankings. Retrieved March 4, 2006, from http://www.alexa.com/site/devcorner/top_sites

Alexa Internet Inc. (2006b). Alexa Top Site Service. Retrieved March 4, 2006, from http://www.alexa.com/site/ds/top_sites

Alvin, T. S. C. (2004). *Cookies on-the-move: managing cookies on a smart card*. Paper presented at the Proceedings of the 2004 ACM symposium on Applied computing.

Ammann, P. E., & Knight, J. C. (1988). Data diversity: an approach to software fault tolerance. *IEEE Transactions on Computers, 37*(4), 418-425.

Andrews, A., Offutt, J., & Alexander, R. (2005). Testing Web Applications by Modeling with FSMs. *Software Systems and Modeling, 4*(3), 326-345.

Antonov, I. A., & Saleev, V. M. (1980). An economic method of computing LPτ - Sequences. *Journal of Computational Mathematics and Mathematical Physics, 19*, 252-256.

Auger, R., Currudo, C., Huseby, S. H., Newman, A. C., Pompon, R., Groves, D., et al. (2005). Web Security Glossary. Retrieved March 12, 2008, from http://www.webappsec.org/projects/glossary/

Baker, L. (2007). Alexa Bias Exposed by Top Google Engineers. Retrieved Jan, 2009, from http://www.searchenginejournal.com/alexa-bias-exposed-by-top-google-engineers/4487/

Bellettini, C., Marchetto, A., & Trentini, A. (2005). *TestUml: user-metrics driven web applications testing*. Paper presented at the Proceedings of the 2005 ACM symposium on Applied computing.

BlackHawk. (2007). RevokeBB Blind SQL Injection / Hash Extractor. Retrieved March 14, 2008, from http://archives.neohapsis.com/archives/bugtraq/2007-06/0014.html

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*(7), 1145–1159.

Bratley, P., & Fox, B. L. (1988). Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software, 14*(1), 88-100.

Bratley, P., Fox, B. L., & Niederreiter, H. (1994). Programs to generate Niederreiter's low-discrepancy sequences. *ACM Transactions on Mathematical Software, 20*(4), 494-495.

CBS News. (2002). CIA Caught Sneaking Cookies. Retrieved October 19, 2006, from http://www.cbsnews.com/stories/2002/03/20/tech/printable504131.shtml

Cgisecurity.com. (2002). The Cross Site Scripting FAQ. Retrieved May 20, 2005, from http://www.cgisecurity.com/articles/xss-faq.shtml

Chan, F. T., Chen, T. Y., Mak, I. K., & Yu, Y. T. (1996). Proportional sampling strategy: guidelines for software testing practitioners. *Information and Software Technology, 38*(12), 775-782.

Chan, K. P., Chen, T. Y., Kuo, F.-C., & Towey, D. (2004). *A revisit of adaptive random testing by restriction*. Paper presented at the

Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International.

Chan, K. P., Chen, T. Y., & Towey, D. (2002). Restricted Random Testing. In *Software Quality — ECSQ 2002* (pp. 321-330).

Chan, K. P., Chen, T. Y., & Towey, D. (2006). *Probabilistic Adaptive Random Testing*. Paper presented at the Quality Software, 2006. QSIC 2006. Sixth International Conference on.

Chen, T. Y., De Hao, H., Tse, T. H., & Zongyuan, Y. (2007). *An Innovative Approach to Tackling the Boundary Effect in Adaptive Random Testing*. Paper presented at the System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on.

Chen, T. Y., Kuo, F.-C., & Merkel, R. (2006). On the statistical properties of testing effectiveness measures. *Journal of Systems and Software, 79*(5), 591-601.

Chen, T. Y., Kuo, F. C., & Liu, H. (2007). *Enhancing Adaptive Random Testing through Partitioning by Edge and Centre*. Paper presented at the Software Engineering Conference, 2007. ASWEC 2007. 18th Australian.

Chen, T. Y., Kuo, F. C., Merkel, R. G., & Ng, S. P. (2003). *Mirror adaptive random testing*. Paper presented at the Quality Software, 2003. Proceedings. Third International Conference on.

Chen, T. Y., Leung, H., & Mak, I. K. (2004). Adaptive Random Testing. In *Advances in Computer Science - ASIAN 2004* (pp. 320-329).

Chen, T. Y., & Merkel, R. (2006). *Efficient and effective random testing using the Voronoi diagram*. Paper presented at the Software Engineering Conference, 2006. Australian.

Chen, T. Y., & Merkel, R. (2007). Quasi-Random Testing. *IEEE Transactions on Reliability, 56*(3), 562-568.

Chen, T. Y., & Merkel, R. (2008). An upper bound on software testing effectiveness. *ACM Transactions on Software Engineering and Methodology, 17*(3), 1-27.

Chen, T. Y., Merkel, R., Wong, P. K., & Eddy, G. (2004). *Adaptive random testing through dynamic partitioning*. Paper presented at the Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on.

Chen, T. Y., Tse, T. H., & Yu, Y. T. (2001). Proportional sampling strategy: a compendium and some insights. *Journal of Systems and Software, 58*(1), 65-81.

Chen, Y., Ohkawa, E., Mabu, S., Shimada, K., & Hirasawa, K. (2009). A portfolio optimization model using Genetic Network Programming with control nodes. *Expert Systems with Applications, 36*(7), 10735–10745.

Chi, H., & Jones, E. L. (2006). *Computational Investigations of Quasirandom Sequences in Generating Test Cases for Specification-Based Tests*. Paper presented at the Simulation Conference, 2006. WSC 06. Proceedings of the Winter.

Cliff, N. (1993). Dominance statistics: ordinal analysis to answer ordinal questions. *Psychological Bulletin, 114*, 494-509.

Cliff, N. (1996). *Ordinal methods for behavioral data analysis*. New Jersey: Lawrence Erlbaum Associates.

Cobham, A. (1964). *The Intrinsic Computational Difficulty of Functions*. Paper presented at the Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science.

Cohen, J. (1988). *Statistical power analysis for the behavioural sciences (2nd ed.)*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112*, 155-159.

comScore Inc. (2007a). comScore Releases March U.S. Search Engine Rankings. Retrieved May, 2007, from http://www.comscore.com/press/release.asp?id=1397

comScore Inc. (2007b). Cookie-Based Counting Overstates SIze of Web Site Audiencces. Retrieved April 18, 2007, from http://www.comscore.com/press/release.asp?press=1389

Cook, S. (2003). A Web developers guide to cross-site scripting. Retrieved July 26, 2006, from http://www.sans.org/reading_room/whitepapers/securecode/988.php

Cranor, L., Dobbs, B., Egelman, S., Hogben, G., Langheinrich, M., Marchiori, M., et al. (2006). The Platform for Privacy Preferences 1.1 (P3P 1.1) Specification. Retrieved May, 2007, from http://www.w3.org/TR/P3P11/

Di Lucca, G. A., Fasolino, A. R., Faralli, F., & De Carlini, U. (2002). *Testing Web applications*. Paper presented at the Proceedings. International Conference on Software Maintenance.

Doyle, B., & Lopes, C. V. (2008). Survey of Technologies for Web Application Development [Electronic Version]. *arXiv:0801.2618*, 43. Retrieved Jan 2009, from http://www.citebase.org/abstract?id=oai:arXiv.org:0801.2618

Duran, J. W., & Ntafos, S. C. (1984). An Evaluation of Random Testing. *IEEE Transactions on Software Engineering, SE-10*(4), 438-444.

Economist Intelligence Unit, & IBM Institute for BusinessValue. (2006). The 2006 e-readiness rankings [Electronic Version], 23,

Elbaum, S., Rothermel, G., Karre, S., & Fisher Ii, M. (2005). Leveraging user-session data to support Web application testing. *Software Engineering, IEEE Transactions on, 31*(3), 187-202.

Fawcett, T. (2003). *ROC graphs: Notes and practical considerations for researchers* (Technical Report HPL-2003-4). Palo Alto, CA, USA: HP Laboratories.

Fielding, R., Mogul, J. C., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. Retrieved Sept. 4, 2007, from http://www.ietf.org/rfc/rfc2616.txt

Finelli, G. B. (1991). NASA Software failure characterization experiments. *Reliability Engineering & System Safety, 32*(1-2), 155-169.

Fogie, S. (2006). XSS, Cookies, and Session ID Authentication - Three Ingredients for a Successful Hack. Retrieved August 14, 2006, from http://www.informit.com/articles/article.asp?p=603037&rl=1

Fox, B. L. (1986). Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators. *ACM Transactions on Mathematical Software, 12*(4), 362-376.

Free Software Foundation. (2008). GSL - GNU Scientific Library. Retrieved Sept., 2008, from http://www.gnu.org/software/gsl/

Garousi, V. (2008). *Empirical analysis of a genetic algorithm-based stress test technique*. Paper presented at the Proceedings of the 10th annual conference on Genetic and evolutionary computation.

Garousi, V., Briand, L. C., & Labiche, Y. (2008). Traffic-aware stress testing of distributed real-time systems based on UML models using genetic algorithms. *Journal of Systems and Software, 81*(2), 161-185.

GeekLog. (2010). GeekLog Documentation. Retrieved Jan., 2010, from http://www.geeklog.net/docs/english/

Gilpin, A. R. (1993). Table for Converstion of Kendall's Tau to Spearman's Rho Within the context of measures of magnitude of effect for meta-analysis. *Educational and Psychological Measurement, 53*(1), 87-52.

Google. (2007). Google Analytics. Retrieved May 7, 2007, from http://www.google.com/analytics/

Google. (2008). GMail. Retrieved Jan 5, 2009, from https://mail.google.com

Hanley, J. A., & McNeil, B. J. (1982). The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology, 143*, 29–36.

Harman, M., Hu, L., Hierons, R., Wegener, J., Sthamer, H., Baresel, A., et al. (2004). Testability transformation. *IEEE Transactions on Software Engineering, 30*(1), 3-16.

Hess, M. R., Kromrey, J. D., Ferron, J. M., Hogarty, K. Y., & Hines, C. V. (2005). *Robust Inference in Meta-Analysis: An Empirical Comparison of Point and Interval Estimates Using the Standardized Mean Difference and Cliff's Delta*. Paper presented at the Annual Meeting of the American Educational Research Association, Montreal.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan.

IP2Location.com. (2006). IP2Location. Retrieved November 20, 2006, from http://www.ip2location.com/

Iron. (2008). EazyPortal <= 1.0 SQL Injection Exploit. Retrieved March 14, 2008, from http://milworm.com/exploits/5196

ISO/IEC. (1996). *Information technology -- Syntactic metalanguage -- Extended BNF* (ISO/IEC 14977:1996).

Juels, A., Jakobsson, M., & Jagatic, T. N. (2006). *Cache cookies for browser authentication*. Paper presented at the IEEE Symposium on Security and Privacy.

Kals, S. (2007, 2006). SecuBat. Retrieved July, 2006, from http://www.secubat.org/

Kristol, D. (2001). HTTP Cookies: Standards, privacy, and politics. *ACM Transactions on Internet Technology, 1*(2), 151-198.

Kristol, D., & Montulli, L. (1997). RFC 2109: HTTP State Management Mechanism. Retrieved March 1, 2006, from http://www.ietf.org/rfc/rfc2109.txt

Kristol, D., & Montulli, L. (2000). RFC 2965: HTTP State Management Mechanism. Retrieved March 1, 2006, from http://www.ietf.org/rfc/rfc2965.txt

Kromrey, J. D., Hogarty, K. Y., Ferron, J. M., Hines, C. V., & Hess, M. R. (2005). *Robustness in Meta-Analysis: An Empirical Comparison of Point and Interval Estimates of Standardized Mean Differences and Cliff's Delta*. Paper presented at the American Statistical Association Joint Statistical Meetings. from http://luna.cas.usf.edu/~mbrannic/files/meta/Robust%20Estimates.pdf

Kung, D. C., Liu, C., & Hsia, P. (2000). *An Object-Oriented Web Test Model for Testing Web Applications*. Paper presented at the Proceedings of the The First Asia-Pacific Conference on Quality Software (APAQS'00).

Li, Z., Harman, M., & Hierons, R. M. (2007). Search Algorithms for Regression Test Case Prioritization. *IEEE Transactions on Software Engineering, 33*(4), 225-237.

Loo, P. S., & Tsai, W. K. (1988). Random testing revisited. *Information and Software Technology, 30*(7), 402-417.

Malaiya, Y. K. (1995). *Antirandom testing: getting the most out of black-box testing*. Paper presented at the Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on.

Mayer, J., & Schneckenburger, C. (2006). *An empirical analysis and comparison of random testing techniques*. Paper presented at the Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering.

McCormick, W. P., Lyons, N. I., & Hutcheson, K. (1992). Distributional properties of Jaccard's index of similarity. *Communications in Statistics - Theory and Methods, 21*(1), 51–68.

Michael, C. C., McGraw, G., & Schatz, M. A. (2001). Generating software test data by evolution. *IEEE Transactions on Software Engineering, 27*(12), 1085-1110.

Microsoft Corp. (2002). No Cookies for You! Internet Explorer Service Pack Helps Thwart Cross-Site Script Attacks. Retrieved Sept. 5, 2007, from http://www.microsoft.com/presspass/features/2002/oct02/10-23xss-ie.mspx

Microsoft Corp. (2007). Mitigating Cross-site Scripting With HTTP-only Cookies. Retrieved Sept. 5, 2007, from http://msdn2.microsoft.com/en-us/library/ms533046.aspx

Miniwatts Marketing Group. (2008). World Internet Users and Population Stats. Retrieved Jan, 2009, from http://www.internetworldstats.com/stats.htm

Mozilla Corporation. (2006). Firefox. Retrieved March 4, 2006, from http://www.mozilla.com/firefox/

Mozilla Developer Center. (2009, Nov 20, 2009). Gecko DOM Reference. Retrieved Jan 5th, 2010, from https://developer.mozilla.org/en/Gecko_DOM_Reference

Myers, G. J. (1976). *The Art of Software Testing*. New York: John Wiley & Sons.

Net Applications. (2006). Browser Market Share. Retrieved November 18, 2006, from http://marketshare.hitslink.com/report.aspx?qprid=0

Net Applications. (2007). Browser Market Share for Calendar Q3, 2007 Retrieved Sept. 5, 2007, from http://marketshare.hitslink.com/report.aspx?qprid=0&qpmr=15&qpdt=1&qpct=3&qptimeframe=Q

Netscape Communications Corporation. (undated). Persistent Client State -- HTTP Cookies. Retrieved Sept. 04, 2007, from http://wp.netscape.com/newsref/std/cookie_spec.html

Nielsen//NetRatings. (2007). Nielsen//NetRatings Announces March U.S. Search Share Rankings. Retrieved May, 2007, from http://www.netratings.com/pr/pr_070320.pdf

Ntafos, S. C. (1998). *On random and partition testing*. Paper presented at the Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis.

Offutt, J., & Wu, Y. (2009). Modeling presentation layers of web applications for testing [Electronic Version]. *Software Systems and Modeling*, from http://www.springerlink.com/content/f1q564154110623x

Offutt, J., Wu, Y., Du, X., & Huang, H. (2004). *Bypass Testing of Web Applications*. Paper presented at the The Fifteenth IEEE International Symposium on Software Reliability Engineering, Saint-Malo, Bretagne, France.

Omniture Inc. (2007). Omniture SiteCatalyst. from http://www.omniture.com/products/web_analytics/sitecatalyst

OneStat.com. (2007, July 2, 2007). Mozilla's Firefox global usage share is still growing according to OneStat.com. Retrieved Sept. 5, 2007, from http://www.onestat.com/html/aboutus_pressbox53-firefox-mozilla-browser-market-share.html

Park, J. S., & Sandhu, R. (2000). Secure cookies on the Web. *IEEE Internet Computing, 4*(4), 36-44.

PHP Group. (2008). The PHP Manual: Magic Quotes. Retrieved March 14, 2008, from http://ca.php.net/magic_quotes

Port80 Software. (2007). Industry Surveys: Top Application Servers. Retrieved Jan, 2009, from http://www.port80software.com/surveys/top1000appservers/

Rathaus, N. (2004). PlaySMS SQL Injetion via Cookie. Retrieved March 14, 2008, from http://www.securiteam.com/unixfocus/5UP0F2ADPS.html

Reay, I., Beatty, P., Dick, S., & Miller, J. (2006). Privacy Policies Versus National Culture and Legislation on the Internet. *ACM Transactions on Computer-Human Interaction*.

Reay, I., Beatty, P., Dick, S., & Miller, J. (2007). A Survey and Analysis of the P3P Protocol's Agents, Adoption, Maintenance, and Future. *Dependable and Secure Computing, IEEE Transactions on, 5*(2), 151-164.

Ricca, F., & Tonella, P. (2001). *Analysis and testing of Web applications*. Paper presented at the Proceedings of the 23rd International Conference on Software Engineering.

Romano, J., Kromrey, J., Coraggio, J., & Skowronek, J. (2006). *Appropriate statistics for ordinal level data*. Paper presented at the Florida Association for Institutional Research.

Sachindra, J., Neeraj, A., Raghu, K., & Sumit, N. (2003). *A bag of paths model for measuring structural similarity in Web documents*. Paper presented at the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.

Samar, V. (1999). *Single sign-on using cookies for Web applications*. Paper presented at the IEEE 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Schneck, P. B. (1979). Comment on "When to Use Random Testing". *IEEE Transactions on Computers, C-28*(8), 580-581.

Secunia. (2005a). PaFileDB Administrative User Authentication SQL Injection Retrieved March 14, 2008, from http://secunia.com/advisories/16566/

Secunia. (2005b). phpCOIN SQL Injection and File Inclusion Vulnerabilities. Retrieved March 14, 2008

Secunia. (2006). e107 Cookie Parameter SQL Injection Vulnerability. Retrieved March 14, 2008, from http://secunia.com/advisories/20089/

SecuriTeam. (2004). Internet Software Sciences's Web+Center SQL Injection. Retrieved March 14, 2008, from http://www.securiteam.com/windowsntfocus/5RP0N0ADGK.html

SecuriTeam. (2008). MyBB SQL Injetion (Exploit). Retrieved March 10, 2008, from http://www.securiteam.com/exploits/5GP0E1PI0Y.html

Security Space. (2006a). Internet Cookie Report. Retrieved October 2, 2006, from http://www.securityspace.com/s_survey/data/man.200609/cookieReport.html

Security Space. (2006b). Technology Penetration Report. Retrieved November 16, 2006, from http://www.securityspace.com/s_survey/data/man.200610/techpen.html

Selkow, S. M. (1977). The tree-to-tree editing problem. *Inf. Process. Lett., 6*(6), 184–186.

Smith, R. M. (1999). The Web Bug FAQ. Retrieved October 26, 2006, from http://www.eff.org/Privacy/Marketing/web_bug.html

Sobol, I. M. (1967). Uniformly distributed sequences with additional uniformity properties. *Journal of Computational Mathematics and Mathematical Physics, 16*, 6.

SourceForge.net. (2010a). e107. Retrieved Jan., 2010, from http://sourceforge.net/projects/e107/

SourceForge.net. (2010b). phpBB. Retrieved Jan., 2010, from http://sourceforge.net/projects/phpbb/

SourceForge.net. (2010c). phpMyAdmin. Retrieved Jan., 2010, from http://sourceforge.net/projects/phpmyadmin/

Tappenden, A. F., Beatty, P., Miller, J., Geras, A., & Smith, M. R. (2005). *Agile Security Testing of Web-Based Systems via HTTPUnit*. Paper presented at the Agile 2005, Denver, Colorado.

Tappenden, A. F., Huynh, T., Miller, J., Geras, A., & Smith, M. R. (2006). Agile Development of Secure Web-Based Applications. *International Journal of Information Technology and Web Engineering, 1*(2), 1-24.

Tezinde, T., Murphy, J., Nguyen, H. C., & Jenkinson, C. (2001). *Cookies: Walking the Fine Line between Love and Hate*. Paper presented at the Western Australian Workshop on Information Systems Research.

The Counter.com. (2006). Browser Stats. Retrieved November 18, 2006, from http://www.thecounter.com/stats/2006/October/browser.php

The Counter.com. (2007). Browser Stats. Retrieved Sept. 5, 2007, from http://www.thecounter.com/stats/2007/August/browser.php

The PHP Group. (2008). The PHP Manual: Magic Quotes. Retrieved March 14,, 2008, from http://ca.php.net/magic_quotes

Tonella, P., & Ricca, F. (2004). *A 2-layer model for the white-box testing of Web applications*. Paper presented at the Proceedings. Sixth IEEE International Workshop on Web Site Evolution.

Trager, C. (2010). BugTracker.NET. Retrieved Jan, 2010, from http://ifdefined.com/bugtrackernet.html

United Nations Statistics Division. (2007). Standard Country and Area Codes Classifications. Retrieved June, 2007, from http://unstats.un.org/unsd/methods/m49/m49regin.htm

Verton, R. (2007). WebSpell Authentication Bypass and arbitrary code execution. Retrieved March 24, 2008, from http://archives.neohapsis.com/archives/bugtraq/2007-02/0426.html

Vijayaraghavan, G., & Kaner, C. (2003). *Bug Taxonomies: Use Them to Generate Better Tests*. Paper presented at the STAR EAST 2003.

Vind, J. (2007). Critical Sql Injection in NukeSentinel 2.5.12. Retrieved March 11, 2008, from http://www.waraxe.us/advisory-58.html

von Mayrhause, A., Chen, T., Hajjar, A., Bai, A., & Anderson, C. (1998). *Fast antirandom (FAR) test generation*. Paper presented at the High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International.

W3 Schools. (2006). Browser Statistics. Retrieved October 19, 2006, from http://www.w3schools.com/browsers/browsers_stats.asp

W3C. (2005, Jan 19, 2005). W3C Document Object Model. Retrieved Jan, 2010, from http://www.w3.org/DOM/

W3C. (2006). Platform for Privacy Preferences (P3P) Project. Retrieved November 18, 2006, from http://www.w3.org/P3P/

Whittaker, J. A. (2000). What is software testing? And why is it so hard? *IEEE Software, 17*(1), 70-79.

Xiao, M., El-Attar, M., Reformat, M., & Miller, J. (2007). Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques. *Empirical Software Engineering, 12*(2), 183-239.

Xu, L., Xu, B., & Jiang, J. (2005). Testing web applications focusing on their specialties. *SIGSOFT Softw. Eng. Notes, 30*(1), 10.

Yahoo! Inc. (2006). Yahoo! Search Marketing. Retrieved May 7, 2007, from http://www.content.overture.com/d/

Yang, W. (1991). Identifying Syntactic Differences Between Two Programs. *Software—Practice and Experience, 21*(7), 739–755.

Yin, H., Lebne-Dengel, Z., & Malaiya, Y. K. (1997). *Automatic test generation using checkpoint encoding and antirandom testing*. Paper presented at the PROCEEDINGS The Eighth International Symposium On Software Reliability Engineering.

Yue, C., Xie, M., & Wang, H. (2007). *Automatic Cookie Usage Setting with CookiePicker*. Paper presented at the Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on.

Zalewski, M. (2006). Cross Site Cooking. Retrieved October 6, 2006, from http://www.securiteam.com/securityreviews/5EP0L2KHFG.html

Zweig, M., & Campbell, G. (1993). Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clin Chem, 39*(4), 561–577.

Appendix A

National Cookie Usage

Table A-1. Cookie usage per site according to country

Country	Number Of Sites		Sites Setting Cookies		Site Setting 1st Party Cookies		Sites Setting 3rd Party Cookies		Sites Setting Sessional Cookies		Sites Setting Persistent Cookies		Sites Setting 3rd Party Persistent Cookies	
United States	44,673	46%	29,589	66%	24,478	55%	16,639	37%	23,459	53%	22,822	51%	14,491	32%
China	17,196	18%	12,193	71%	10,806	63%	5,802	34%	10,741	62%	7,672	45%	3,278	19%
Japan	6,658	7%	3,431	52%	2,574	39%	1,730	26%	2,283	34%	2,678	40%	1,386	21%
United Kingdom	3,493	4%	2,628	75%	2,297	66%	1,471	42%	2,143	61%	1,980	57%	1,299	37%
Canada	2,527	3%	1,726	68%	1,402	55%	907	36%	1,357	54%	1,294	51%	790	31%
Germany	2,411	2%	1,631	68%	1,271	53%	1,053	44%	1,381	57%	1,215	50%	858	36%
France	1,837	2%	1,405	76%	1,006	55%	1,056	57%	1,000	54%	1,219	66%	992	54%
South Korea	1,716	2%	1,311	76%	1,159	68%	544	32%	1,201	70%	618	36%	392	23%
Netherlands	1,509	2%	1,016	67%	805	53%	666	44%	774	51%	809	54%	592	39%
Hong Kong	1,401	1%	705	50%	583	42%	273	19%	529	38%	427	30%	197	14%
Taiwan	1,298	1%	743	57%	641	49%	271	21%	651	50%	361	28%	185	14%
Spain	1,222	1%	860	70%	708	58%	505	41%	733	60%	641	52%	439	36%
Russian	768	1%	665	87%	499	65%	604	79%	466	61%	637	83%	595	77%
Australia	766	1%	541	71%	469	61%	284	37%	436	57%	377	49%	248	32%
Turkey	742	1%	551	74%	480	65%	239	32%	506	68%	280	38%	175	24%
Sweden	699	1%	563	81%	496	71%	307	44%	488	70%	410	59%	284	41%
Israel	670	1%	523	78%	469	70%	255	38%	476	71%	333	50%	206	31%
Italy	634	1%	457	72%	367	58%	260	41%	348	55%	352	56%	240	38%
Denmark	430	0%	353	82%	301	70%	209	49%	311	72%	276	64%	197	46%
Brazil	399	0%	266	67%	245	61%	79	20%	233	58%	158	40%	65	16%
Greece	321	0%	215	67%	188	59%	102	32%	184	57%	130	40%	75	23%
Thailand	314	0%	225	72%	201	64%	126	40%	213	68%	183	58%	84	27%
Switzerland	283	0%	199	70%	169	60%	104	37%	166	59%	139	49%	93	33%
Czech Republic	269	0%	210	78%	146	54%	170	63%	143	53%	181	67%	159	59%
Norway	261	0%	212	81%	173	66%	153	59%	195	75%	176	67%	140	54%
Belgium	234	0%	192	82%	172	74%	121	52%	149	64%	157	67%	115	49%
Poland	229	0%	167	73%	126	55%	121	53%	124	54%	137	60%	114	50%
Austria	228	0%	167	73%	127	56%	104	46%	137	60%	120	53%	95	42%
India	223	0%	135	61%	112	50%	57	26%	121	54%	57	26%	36	16%
Singapore	212	0%	131	62%	103	49%	75	35%	98	46%	93	44%	60	28%
Hungary	208	0%	169	81%	130	63%	132	63%	116	56%	145	70%	126	61%
Mexico	179	0%	103	58%	91	51%	44	25%	81	45%	66	37%	39	22%
Argentina	171	0%	126	74%	110	64%	73	43%	108	63%	93	54%	62	36%
Finland	169	0%	132	78%	104	62%	96	57%	111	66%	104	62%	91	54%
Malaysia	164	0%	99	60%	86	52%	40	24%	85	52%	59	36%	29	18%
Lithuania	155	0%	144	93%	132	85%	47	30%	124	80%	136	88%	42	27%
Saudi Arabia	132	0%	79	60%	63	48%	33	25%	72	55%	36	27%	25	19%
Viet Nam	123	0%	98	80%	90	73%	30	24%	90	73%	47	38%	17	14%
New Zealand	119	0%	91	76%	73	61%	66	55%	74	62%	75	63%	61	51%
Egypt	114	0%	83	73%	77	68%	28	25%	79	69%	31	27%	17	15%

Country	Number Of Sites		Sites Setting Cookies		Site Setting 1st Party Cookies		Sites Setting 3rd Party Cookies		Sites Setting Sessional Cookies		Sites Setting Persistent Cookies		Sites Setting 3rd Party Persistent Cookies	
Ireland	107	0%	85	79%	75	70%	47	44%	76	71%	60	56%	44	41%
Romania	107	0%	91	85%	75	70%	78	73%	70	65%	84	79%	76	71%
Portugal	99	0%	76	77%	67	68%	37	37%	62	63%	50	51%	31	31%
Bulgaria	87	0%	71	82%	61	70%	45	52%	59	68%	55	63%	42	48%
Ukraine	83	0%	44	53%	36	43%	34	41%	36	43%	38	46%	30	36%
Chile	79	0%	55	70%	49	62%	35	44%	49	62%	42	53%	34	43%
Uruguay	75	0%	36	48%	29	39%	16	21%	33	44%	23	31%	14	19%
Virgin Islands, British	70	0%	23	33%	15	21%	13	19%	16	23%	15	21%	10	14%
Venezuela	68	0%	47	69%	40	59%	17	25%	41	60%	26	38%	14	21%
South Africa	67	0%	53	79%	48	72%	37	55%	46	69%	46	69%	34	51%
United Arab Emirates	65	0%	52	80%	44	68%	22	34%	46	71%	24	37%	15	23%
Indonesia	65	0%	46	71%	34	52%	18	28%	35	54%	22	34%	11	17%
Slovenia	62	0%	52	84%	45	73%	32	52%	48	77%	40	65%	31	50%
Slovakia	61	0%	48	79%	31	51%	36	59%	27	44%	43	70%	36	59%
Serbia and Montenegro	55	0%	28	51%	23	42%	11	20%	23	42%	17	31%	9	16%
Estonia	53	0%	41	77%	27	51%	28	53%	32	60%	33	62%	27	51%
Costa Rica	43	0%	30	70%	27	63%	16	37%	22	51%	22	51%	15	35%
Croatia	43	0%	23	53%	23	53%	14	33%	21	49%	16	37%	12	28%
Philippines	32	0%	18	56%	18	56%	2	6%	17	53%	13	41%	2	6%
Colombia	30	0%	15	50%	12	40%	10	33%	11	37%	12	40%	9	30%
Panama	30	0%	20	67%	16	53%	8	27%	14	47%	16	53%	8	27%
Peru	29	0%	14	48%	11	38%	7	24%	13	45%	5	17%	5	17%
Latvia	26	0%	23	88%	20	77%	17	65%	21	81%	19	73%	16	62%
Iceland	25	0%	21	84%	20	80%	15	60%	19	76%	18	72%	13	52%
Jordan	24	0%	14	58%	10	42%	7	29%	10	42%	6	25%	4	17%
Gibraltar	23	0%	20	87%	19	83%	15	65%	18	78%	18	78%	15	65%
Kuwait	22	0%	18	82%	15	68%	4	18%	18	82%	2	9%	2	9%
Malta	22	0%	13	59%	12	55%	3	14%	12	55%	7	32%	2	9%
Dominica	21	0%	9	43%	9	43%	8	38%	8	38%	9	43%	8	38%
Belize	19	0%	16	84%	16	84%	3	16%	14	74%	5	26%	3	16%
Macao	19	0%	10	53%	7	37%	4	21%	9	47%	5	26%	2	11%
Cyprus	19	0%	14	74%	13	68%	7	37%	13	68%	10	53%	6	32%
Syrian Arab Republic	17	0%	10	59%	8	47%	3	18%	9	53%	3	18%	1	6%
Luxembourg	17	0%	14	82%	13	76%	4	24%	12	71%	8	47%	4	24%
Bermuda	16	0%	14	88%	14	88%	4	25%	14	88%	13	81%	4	25%
Iran	15	0%	6	40%	6	40%	3	20%	6	40%	1	7%	1	7%
Grenada	14	0%	9	64%	2	14%	8	57%	1	7%	9	64%	8	57%
Pakistan	14	0%	8	57%	8	57%	3	21%	7	50%	4	29%	1	7%
Dominican Republic	10	0%	8	80%	8	80%	4	40%	8	80%	6	60%	3	30%
Ecuador	10	0%	6	60%	6	60%	1	10%	5	50%	2	20%	1	10%
Puerto Rico	10	0%	7	70%	7	70%	4	40%	7	70%	4	40%	4	40%
Lebanon	10	0%	8	80%	6	60%	3	30%	7	70%	5	50%	3	30%
Morocco	9	0%	2	22%	2	22%	1	11%	2	22%	1	11%	1	11%
Qatar	9	0%	5	56%	4	44%	1	11%	5	56%	1	11%	1	11%
Netherlands Antilles	8	0%	6	75%	5	63%	3	38%	5	63%	4	50%	2	25%
Antigua and Barbuda	7	0%	5	71%	5	71%	1	14%	5	71%	5	71%	1	14%
Bosnia and Herzegovina	6	0%	6	100%	5	83%	4	67%	6	100%	4	67%	4	67%
Tunisia	6	0%	4	67%	4	67%	1	17%	4	67%	1	17%	1	17%
Bolivia	5	0%	1	20%	1	20%	1	20%	1	20%	1	20%	1	20%
Bahamas	5	0%	2	40%	2	40%	2	40%	1	20%	2	40%	2	40%
Guatemala	4	0%	3	75%	3	75%	0	0%	3	75%	2	50%	0	0%
Sri Lanka	4	0%	3	75%	2	50%	1	25%	3	75%	1	25%	1	25%
Mauritius	4	0%	4	100%	2	50%	3	75%	3	75%	2	50%	2	50%

Country	Number Of Sites		Sites Setting Cookies		Site Setting 1st Party Cookies		Sites Setting 3rd Party Cookies		Sites Setting Sessional Cookies		Sites Setting Persistent Cookies		Sites Setting 3rd Party Persistent Cookies	
Yemen	4	0%	1	25%	1	25%	0	0%	0	0%	1	25%	0	0%
Libyan Arab Jamahiriya	3	0%	1	33%	1	33%	0	0%	0	0%	1	33%	0	0%
Algeria	3	0%	2	67%	2	67%	0	0%	1	33%	1	33%	0	0%
Paraguay	3	0%	2	67%	2	67%	1	33%	2	67%	1	33%	1	33%
Macedonia		0%	3	100%	3	100%	2	67%	3	100%	2	67%	2	67%
Barbados	3	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Belarus	3	0%	2	67%	2	67%	1	33%	1	33%	2	67%	1	33%
Moldova, Republic Of	3	0%	3	100%	3	100%	3	100%	3	100%	3	100%	3	100%
Bahrain	3	0%	1	33%	1	33%	0	0%	1	33%	0	0%	0	0%
Samoa	2	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Sudan	2	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Honduras	2	0%	1	50%	1	50%	0	0%	1	50%	0	0%	0	0%
Saint Kitts and Nevis	2	0%	2	100%	2	100%	0	0%	2	100%	0	0%	0	0%
Cote Divoire	2	0%	2	100%	1	50%	1	50%	1	50%	1	50%	1	50%
Nicaragua	2	0%	1	50%	1	50%	1	50%	1	50%	1	50%	1	50%
Myanmar	2	0%	1	50%	1	50%	1	50%	1	50%	0	0%	0	0%
Guinea-Bissau	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Botswana	1	0%	1	100%	0	0%	1	100%	1	100%	0	0%	0	0%
Trinidad and Tobago	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Cuba	1	0%	1	100%	1	100%	0	0%	1	100%	0	0%	0	0%
Albania	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Fiji	1	0%	1	100%	1	100%	0	0%	0	0%	1	100%	0	0%
Liechtenstein	1	0%	1	100%	1	100%	0	0%	1	100%	1	100%	0	0%
American Samoa	1	0%	1	100%	0	0%	1	100%	0	0%	1	100%	1	100%
Georgia	1	0%	1	100%	1	100%	1	100%	1	100%	1	100%	1	100%
Nigeria	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Holy See (Vatican City State)	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Iraq	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Kazakhstan	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Oman	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Afghanistan	1	0%	1	100%	1	100%	1	100%	1	100%	1	100%	1	100%
Cambodia	1	0%	1	100%	1	100%	0	0%	1	100%	0	0%	0	0%
El Salvador	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Brunei Darussalam	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Turks and Caicos Islands	1	0%	1	100%	0	0%	1	100%	1	100%	1	100%	1	100%
Bangladesh	1	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Faroe Islands	1	0%	1	100%	0	0%	1	100%	1	100%	1	100%	1	100%
Totals	97,050	100%	65,423	67%	54,616	56%	35,593	37%	53,064	55%	47,624	49%	29,029	30%

Table A-2. Cookie usage according to country

Country	Number of Cookies Set		Number of 1st Party Cookies		Number of 3rd Party Cookies		Number of Sessional Cookies		Number of Persistent Cookies		Number of 3rd Party Persistent Cookies	
United States	140,940	50%	78,615	56%	62,325	44%	52,074	37%	88,866	63%	14,491	10%
China	47,205	17%	30,231	64%	16,974	36%	21,760	46%	25,445	54%	3,278	7%
Japan	11,012	4%	6,226	57%	4,786	43%	4,231	38%	6,781	62%	1,386	13%
United Kingdom	11,376	4%	6,735	59%	4,641	41%	4,423	39%	6,953	61%	1,299	11%
Canada	7,081	2%	4,236	60%	2,845	40%	2,680	38%	4,401	62%	790	11%
Germany	7,653	3%	3,755	49%	3,898	51%	2,704	35%	4,949	65%	858	11%
France	6,495	2%	2,831	44%	3,664	56%	2,192	34%	4,303	66%	992	15%
South Korea	4,584	2%	2,643	58%	1,941	42%	2,731	60%	1,853	40%	392	9%
Netherlands	4,482	2%	2,378	53%	2,104	47%	1,627	36%	2,855	64%	592	13%
Hong Kong	2,070	1%	1,339	65%	731	35%	1,032	50%	1,038	50%	197	10%
Taiwan	2,374	1%	1,436	60%	938	40%	1,196	50%	1,178	50%	185	8%
Spain	3,724	1%	2,131	57%	1,593	43%	1,365	37%	2,359	63%	439	12%
Russian	4,373	2%	1,250	29%	3,123	71%	1,015	23%	3,358	77%	595	14%
Australia	2,162	1%	1,346	62%	816	38%	931	43%	1,231	57%	248	11%
Turkey	1,665	1%	1,068	64%	597	36%	888	53%	777	47%	175	11%
Sweden	2,499	1%	1,344	54%	1,155	46%	940	38%	1,559	62%	284	11%
Israel	2,092	1%	1,276	61%	816	39%	980	47%	1,112	53%	206	10%
Italy	1,759	1%	1,006	57%	753	43%	600	34%	1,159	66%	240	14%
Denmark	1,600	1%	790	49%	810	51%	660	41%	940	59%	197	12%
Brazil	902	0%	708	78%	194	22%	450	50%	452	50%	65	7%
Greece	651	0%	371	57%	280	43%	292	45%	359	55%	75	12%
Thailand	1,140	0%	815	71%	325	29%	562	49%	578	51%	84	7%
Switzerland	808	0%	475	59%	333	41%	326	40%	482	60%	93	12%
Czech Republic	1,040	0%	327	31%	713	69%	279	27%	761	73%	159	15%
Norway	1,240	0%	472	38%	768	62%	501	40%	739	60%	140	11%
Belgium	879	0%	541	62%	338	38%	311	35%	568	65%	115	13%
Poland	798	0%	294	37%	504	63%	213	27%	585	73%	114	14%
Austria	757	0%	348	46%	409	54%	295	39%	462	61%	95	13%
India	405	0%	226	56%	179	44%	187	46%	218	54%	36	9%
Singapore	456	0%	253	55%	203	45%	172	38%	284	62%	60	13%
Hungary	764	0%	273	36%	491	64%	173	23%	591	77%	126	16%
Mexico	336	0%	193	57%	143	43%	147	44%	189	56%	39	12%
Argentina	582	0%	338	58%	244	42%	231	40%	351	60%	62	11%
Finland	626	0%	247	39%	379	61%	204	33%	422	67%	91	15%
Malaysia	321	0%	218	68%	103	32%	155	48%	166	52%	29	9%
Lithuania	754	0%	556	74%	198	26%	267	35%	487	65%	42	6%
Saudi Arabia	239	0%	136	57%	103	43%	105	44%	134	56%	25	10%
Viet Nam	263	0%	205	78%	58	22%	150	57%	113	43%	17	6%
New Zealand	432	0%	165	38%	267	62%	171	40%	261	60%	61	14%
Egypt	224	0%	173	77%	51	23%	129	58%	95	42%	17	8%
Ireland	381	0%	222	58%	159	42%	145	38%	236	62%	44	12%
Romania	522	0%	205	39%	317	61%	104	20%	418	80%	76	15%
Portugal	346	0%	202	58%	144	42%	131	38%	215	62%	31	9%
Bulgaria	245	0%	159	65%	86	35%	117	48%	128	52%	42	17%
Ukraine	246	0%	94	38%	152	62%	57	23%	189	77%	30	12%
Chile	266	0%	115	43%	151	57%	140	53%	126	47%	34	13%
Uruguay	126	0%	78	62%	48	38%	66	52%	60	48%	14	11%
Virgin Islands, British	67	0%	37	55%	30	45%	36	54%	31	46%	10	15%
Venezuela	129	0%	79	61%	50	39%	70	54%	59	46%	14	11%
South Africa	404	0%	199	49%	205	51%	120	30%	284	70%	34	8%

Country	Number of Cookies Set		Number of 1st Party Cookies		Number of 3rd Party Cookies		Number of Sessional Cookies		Number of Persistent Cookies		Number of 3rd Party Persistent Cookies	
United Arab Emirates	146	0%	79	54%	67	46%	79	54%	67	46%	15	10%
Indonesia	126	0%	60	48%	66	52%	80	63%	46	37%	11	9%
Slovenia	289	0%	135	47%	154	53%	108	37%	181	63%	31	11%
Slovakia	190	0%	76	40%	114	60%	55	29%	135	71%	36	19%
Serbia and Montenegro	70	0%	42	60%	28	40%	30	43%	40	57%	9	13%
Estonia	177	0%	74	42%	103	58%	74	42%	103	58%	27	15%
Costa Rica	105	0%	66	63%	39	37%	41	39%	64	61%	15	14%
Croatia	85	0%	61	72%	24	28%	43	51%	42	49%	12	14%
Philippines	62	0%	51	82%	11	18%	27	44%	35	56%	2	3%
Colombia	76	0%	31	41%	45	59%	24	32%	52	68%	9	12%
Panama	80	0%	50	63%	30	38%	30	38%	50	63%	8	10%
Peru	35	0%	15	43%	20	57%	28	80%	7	20%	5	14%
Latvia	99	0%	44	44%	55	56%	34	34%	65	66%	16	16%
Iceland	73	0%	51	70%	22	30%	39	53%	34	47%	13	18%
Jordan	30	0%	12	40%	18	60%	23	77%	7	23%	4	13%
Gibraltar	149	0%	105	70%	44	30%	52	35%	97	65%	15	10%
Kuwait	36	0%	20	56%	16	44%	26	72%	10	28%	2	6%
Malta	34	0%	24	71%	10	29%	22	65%	12	35%	2	6%
Dominica	39	0%	13	33%	26	67%	24	62%	15	38%	8	21%
Belize	31	0%	22	71%	9	29%	17	55%	14	45%	3	10%
Macao	22	0%	18	82%	4	18%	14	64%	8	36%	2	9%
Cyprus	47	0%	33	70%	14	30%	25	53%	22	47%	6	13%
Syrian Arab Republic	26	0%	12	46%	14	54%	21	81%	5	19%	1	4%
Luxembourg	43	0%	27	63%	16	37%	17	40%	26	60%	4	9%
Bermuda	82	0%	69	84%	13	16%	24	29%	58	71%	4	5%
Iran	16	0%	13	81%	3	19%	9	56%	7	44%	1	6%
Grenada	17	0%	2	12%	15	88%	1	6%	16	94%	8	47%
Pakistan	22	0%	11	50%	11	50%	15	68%	7	32%	1	5%
Dominican Republic	48	0%	25	52%	23	48%	18	38%	30	63%	3	6%
Ecuador	8	0%	7	88%	1	13%	6	75%	2	25%	1	13%
Puerto Rico	37	0%	13	35%	24	65%	22	59%	15	41%	4	11%
Lebanon	20	0%	15	75%	5	25%	11	55%	9	45%	3	15%
Morocco	4	0%	2	50%	2	50%	2	50%	2	50%	1	25%
Qatar	16	0%	4	25%	12	75%	13	81%	3	19%	1	6%
Netherlands Antilles	16	0%	9	56%	7	44%	9	56%	7	44%	2	13%
Antigua and Barbuda	25	0%	23	92%	2	8%	10	40%	15	60%	1	4%
Bosnia and Herzegovina	41	0%	16	39%	25	61%	19	46%	22	54%	4	10%
Tunisia	14	0%	7	50%	7	50%	10	71%	4	29%	1	7%
Bolivia	3	0%	1	33%	2	67%	2	67%	1	33%	1	33%
Bahamas	7	0%	2	29%	5	71%	1	14%	6	86%	2	29%
Guatemala	9	0%	9	100%	0	0%	5	56%	4	44%	0	0%
Sri Lanka	6	0%	2	33%	4	67%	3	50%	3	50%	1	17%
Mauritius	13	0%	3	23%	10	77%	4	31%	9	69%	2	15%
Yemen	1	0%	1	100%	0	0%	0	0%	1	100%	0	0%
Libyan Arab Jamahiriya	1	0%	1	100%	0	0%	0	0%	1	100%	0	0%
Algeria	3	0%	3	100%	0	0%	1	33%	2	67%	0	0%
Paraguay	4	0%	2	50%	2	50%	3	75%	1	25%	1	25%
Macedonia	12	0%	9	75%	3	25%	8	67%	4	33%	2	17%
Barbados	0	0%	0		0		0		0		0	
Belarus	13	0%	7	54%	6	46%	1	8%	12	92%	1	8%
Moldova, Republic Of	20	0%	9	45%	11	55%	5	25%	15	75%	3	15%
Bahrain	1	0%	1	100%	0	0%	1	100%	0	0%	0	0%
Samoa	0	0%	0		0		0		0		0	

Country	Number of Cookies Set		Number of 1st Party Cookies		Number of 3rd Party Cookies		Number of Sessional Cookies		Number of Persistent Cookies		Number of 3rd Party Persistent Cookies	
Sudan	0	0%	0		0		0		0		0	
Honduras	1	0%	1	100%	0	0%	1	100%	0	0%	0	0%
Saint Kitts and Nevis	2	0%	2	100%	0	0%	2	100%	0	0%	0	0%
Cote Divoire	4	0%	3	75%	1	25%	3	75%	1	25%	1	25%
Nicaragua	8	0%	5	63%	3	38%	3	38%	5	63%	1	13%
Myanmar	2	0%	1	50%	1	50%	2	100%	0	0%	0	0%
Guinea-Bissau	0	0%	0		0		0		0		0	
Botswana	3	0%	0	0%	3	100%	3	100%	0	0%	0	0%
Trinidad and Tobago	0	0%	0		0		0		0		0	
Cuba	1	0%	1	100%	0	0%	1	100%	0	0%	0	0%
Albania	0	0%	0		0		0		0		0	
Fiji	2	0%	2	100%	0	0%	0	0%	2	100%	0	0%
Liechtenstein	3	0%	3	100%	0	0%	1	33%	2	67%	0	0%
American Samoa	3	0%	0	0%	3	100%	0	0%	3	100%	1	33%
Georgia	6	0%	3	50%	3	50%	2	33%	4	67%	1	17%
Nigeria	0	0%	0		0		0		0		0	
Holy See (Vatican City State)	0	0%	0		0		0		0		0	
Iraq	0	0%	0		0		0		0		0	
Kazakhstan	0	0%	0		0		0		0		0	
Oman	0	0%	0		0		0		0		0	
Afghanistan	7	0%	1	14%	6	86%	3	43%	4	57%	1	14%
Cambodia	1	0%	1	100%	0	0%	1	100%	0	0%	0	0%
El Salvador	0	0%	0		0		0		0		0	
Brunei Darussalam	0	0%	0		0		0		0		0	
Turks and Caicos Islands	4	0%	0	0%	4	100%	1	25%	3	75%	1	25%
Bangladesh	0	0%	0		0		0		0		0	
Faroe Islands	2	0%	0	0%	2	100%	1	50%	1	50%	1	50%
Totals	284,073	100%	160,770	57%	123,303	43%	111,495	39%	172,578	61%	29,029	10%

Appendix B

CookieCruncher: Cookie Collection Test Harness

Cookie collection testing, as outlined in Chapter 5, involves the testing of a web application from the perspective of modifying the collections of cookies stored within a user-agent. This testing strategy, based on the results of the extensive survey conducted in Chapter 3, employs the technique of anti random testing in the generation of testing vectors that can be used to verify a web application. Due to the ability of this testing strategy to define large pools of testing data for any application, this process requires a computer-assisted testing harness that is capable of providing both a framework for test-data generation and an automated test execution and evaluation environment. This appendix will provide an overview of the testing tool CookieCruncher developed to automate test input generation, execution, and evaluation.

The remainder of this appendix will be organized as follows. Section B.1 will outline the manual process the test harness seeks to automate; Section B.2 will provide an overview of the testing harness; Section B.3 to B.5 will provide a detailed description of the instrumented browsing, test input generation, and test execution and evaluation processes highlighting the critical design decisions; Section B.6 will discuss the application-specific testing adaptations required to test real-world web applications; Section B.7 will outline the extendibility of the framework, suggesting avenues for future collaborations, and Section B.8 will summarize the chapter highlighting the key components of the testing harness.

B.1 Cookie Collection Testing: The Manual Process

Before any process can be automated, an accurate description of the underlying manual process that is to be performed is required. Cookie collection testing is a four-step process: test case definition, test input generation, test execution, and evaluation. To date, discussions of cookie collection testing have focused upon the second step within the process—test input generation. The remainder of this section will review the process of test input generation and fill in the blanks in regards to test-case definition, test execution, and test evaluation.

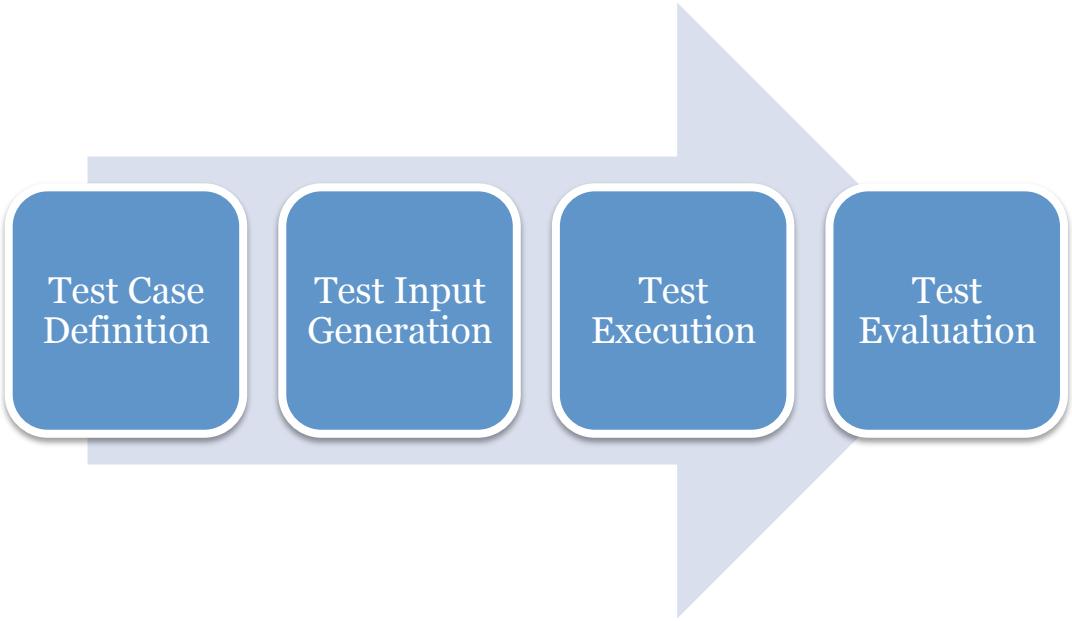


Figure B-1. The Four Step Cookie Collection Process

B.1.1 Test Case Definition

The test case definition is specified for cookie collection testing in Chapter 5.4.1. Table B-1 provides a brief summary of the definition of a test case within cookie collection testing.

Table B-1. Test Case Definition

Test Case ID	
Pre-Conditions:	*A list of HTTP Requests and associated inputs required bring an application to the known testing state.*
Test Request:	*The HTTP Request and associated inputs to be evaluated.*
Cookie Collection:	*A cookie collection representing the cookies present for to be sent with the Test Request.*

B.1.2 Test Input Generation

The generation of cookie collection tests requires two artifacts: a test request, and the collection of cookies used within the web application. Using the test request (specifically the cookies sent along with the test request) and the global cookie collection for a given application, a tester can construct an anti random set of testing cookie collections against which the request can be verified. The process of generating anti random cookie collections is outlined in detail in Chapter 5.3, and will only be briefly summarized here.

Given the set *GC* of cookies present globally within an application, and the set *RC* of cookies presented within a test request, a set of seeding-vectors can be constructed based on the following rules:

Presence. Seeding vectors should be constructed based upon the presence and absence of cookies within the test request. To this end, test vectors should be devoted to the intersection of *GC* and *RC*, and the complement of $GC \cap RC$. Essentially, there should be a seed containing all of the cookies present in *RC,* and all of the cookies except those present in *RC*.

Global Cookies. Two seeding vectors should always exist: *GC* and the complement of *GC*. The set of testing vectors should contain a

vector representing the presence of all of the cookies in *GC*, and the absence of all cookies in *GC*.

Hostname. Seeding vectors should be present that reflect the distinct hostnames from which cookies are present. These vectors should include a first-party only vector, a third-party only vector, and a vector dedicated to cookies from each specific third-party host.

Expiration. Seeding vectors should be present that reflect the various expiration dates of the cookies within *GC*. Two seeding vectors should be derived from the simple partition of sessional versus persistent cookies, as well as a seeding vector devoted to each specific expiration date.

Secure. Two seeding vectors should be devoted to the presence and absence of secure connections only cookies.

HttpOnly. Two seeding vectors should be devoted to the presence and absence of HttpOnly cookies.

P3P Policy. Seeding vectors should be derived from the usage of P3P policy in conjunction with third-party cookies. Vectors should be based upon each specific P3P policy encountered within *GC*, as well as a vector devoted to the absence of all third-party cookies without P3P policies.

Path. Seeding vectors should be derived from cookies only valid for specific paths within the application. Vectors should be generated for the presence and absence of each path encountered within *GC*.

Once the seeding vectors have been generated for a particular test request, the process of anti random test data generation is applied to the collection of seeding vectors, resulting in a set of seeding and generated vectors that can be used to verify the web application for the specific test request.

B.1.3 Test Execution

The execution of a test case, as defined in Chapter 5.4.1, is a straightforward three-step process:

1. All pre-conditions defined within the test case must be realized. This necessitates an interaction with the web application to place the system within a known state, and involves interaction with the web application as a series of HTTP requests.
2. Once the system state is set, the cookie collection is modified to reflect the cookie collection defined within the test case. Cookie collection modification is performed through the creation and deletion of cookies present within the current cookie collection. A cookie is said to be present within a cookie collection based upon the following equivalence definition: two cookies are equivalent if and only if their name, path, and host values are identical (Kristol & Montulli, 1997, 2000).
3. A test request is sent to the web application. The application's response to the test request constitutes the output from a test.

B.1.4 Test Evaluation

An HTTP response is the final output of the testing execution process. This response requires manual inspection to evaluate whether the test case has triggered a fault, or if the application has executed without error. This decision requires the subjective discretion of the tester and should be made in concert with a comparison to the *expected* value of the output, which in cookie collection testing is often the response if cookie modification had not occurred, or if the test request was sent without any cookies present.

B.2 CookieCruncher: An Automated Cookie Collection Testing Tool

The survey conducted in Chapter 3 indicated that web applications within the 95th percentile were found to use anywhere between 0 – 10 cookies. Given a typical web application existing within this range, it is plausible that for each page, a possible 2^{10} test cases could be defined—clearly the manual testing process outlined in Section B.1 requires an automated

testing harness. This section will introduce the testing tool CookieCruncher which has been developed to automate the testing tasks outlined in Section B.1.

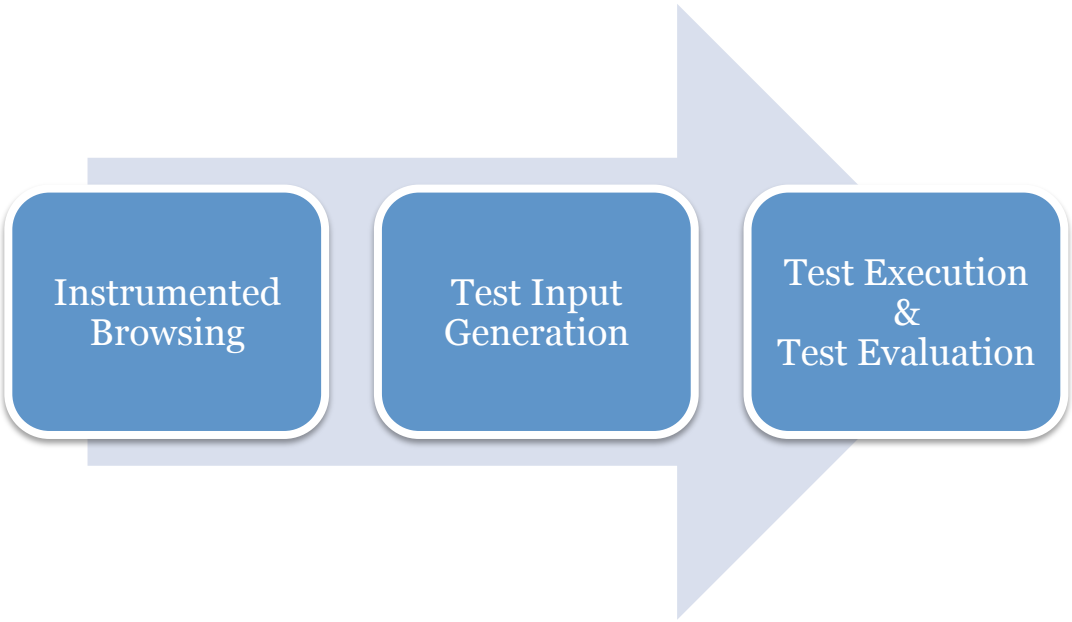


Figure B-2. CookieCruncher's Three Step Testing Process

CookieCruncher, like the manual process previously described, uses a three-stage testing process (Figure B-2) in which the test execution and test evaluation phases have been amalgamated. CookieCruncher is completely automated, and only requires minimal user interaction between each of the three phases. CookieCruncher has been developed as an extension to the Firefox web browser (Mozilla Corporation, 2006), allowing CookieCruncher to leverage the advanced technological platform offered by Firefox, the second most popular Internet browser (The Counter.com, 2006, 2007; W3 Schools, 2006). The development of CookieCruncher as a Firefox extension also allows the testing tool to be platform independent and able to perform tests from any machine capable of running Firefox.

B.2.1 CookieCruncher: An Overview of Basic Functionality

To test a web application with CookieCruncher, a tester simply records a set of test cases through the use of an instrumented browsing session.

This recording process is performed as the tester interacts with a web application through the standard Firefox browser window. During an instrumented browsing session, all of the requests and cookies are automatically recorded. Once the tester has completed the recording of a test session, a record of the browsing session is produced and saved for processing. A stored browsing session is used as input into the CookieCruncher test generator. This module reads the browsing session, compiles the list of cookies used within the browsing session, and outputs a test report containing test cases generated for each request recorded within the browsing session. Finally, the CookieCruncher Test Manager allows a tester to open a test report, and run a test suite upon a web application. The Test Manager automatically fetches each result, modifies the cookie collection, and evaluates the output of the test based upon a comparison of the output to the original browser request.

B.3 Instrumented Browsing

The Instrumented phase of the testing process involves the automated recording of a browsing session. Essentially CookieCruncher provides an interface in which a specific path[23] through a web application can be recorded. Figure B-3 describes the automated cycle that serves as the basis for the automatic recording of a browsing session.

Before the Step 1 in the process is executed, the system must first initialize the browser environment to allow for an accurate recording of the browsing session. First and foremost, all cookies present within the browsing environment must be removed, allowing for the detection of each and every cookie set by the web application. The caching ability of the browsing environment must also be disabled before instrumented browsing can begin. Disabling browser caching ensures that each and every request initiated by the tester reaches the web application. Once the

[23] A path through a web application is defined as an ordered series of HTTP requests with associated input values.

browsing environment has been initialized, the three-step cycle begins and is implemented until the tester terminates the browsing session.

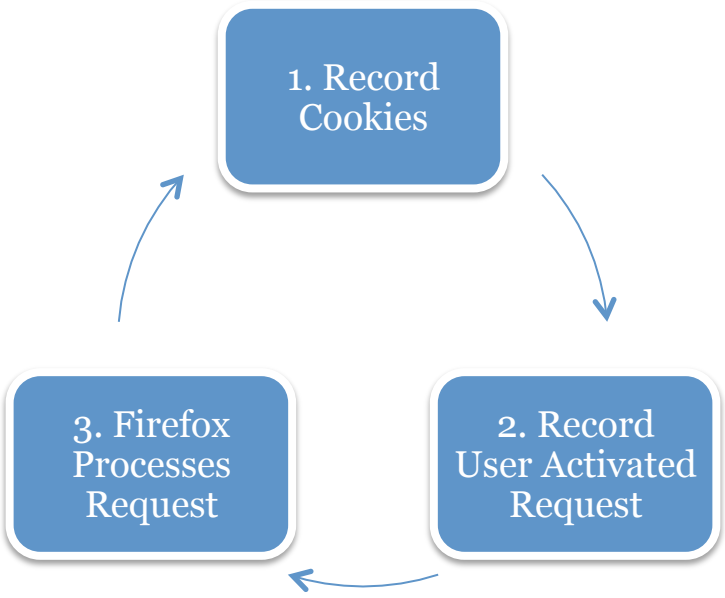


Figure B-3. The Three-Step Instrumented Browsing Process

The first step in the process is to record the cookies present within the browsing environment. This step, executed before any tester-initiated request is recorded, serves to record the cookies present that will be sent with the next request. Step 1 is performed before each request is sent to the server and provides an accurate description of the cookies present at each request.

The second step begins in response to a tester-activated request. Requests are generated anytime a tester clicks on a hyperlink, types a URL in the navigation bar, or submits a form. When this event is triggered, the system records the request, including all data sent along with the request, and allows Firefox to process the request unhindered, just as if the request had been made from within an unmodified Firefox environment.

The processing of a request is the final step within the cycle. Although the process is unhindered, it is instrumented to record any additional requests for HTML or XML documents. This additional recording provides the ability for the system to record events triggered by

AJAX requests and frame objects. Once the processing has stopped and Firefox has determined that the page has fully loaded—including images, embedded Flash objects, and JavaScript components—the process returns to Step 1 and records the cookies for the next test request.

The strength of this recording process is the ability to accurately record requests generated from browsing a test application. This automated process allows for the creation of detailed tests by simply interacting with the system under test. Furthermore, because the application is accessed from within an actual browser environment, a number of web-specific testing conditions can be explored. For example, web navigation can be accurately documented because of the ability to use the built-in forward and backward navigation capabilities of the Firefox browser. Other notable web-specific capabilities that come built into the Firefox browser are:

- The ability to load and execute JavaScript components within an HTML document;
- The ability to intercept and record XMLHttpRequests—a fundamental component of AJAX applications; and
- The integration of Cascading Style Sheets (CSS) directives within the final page presentation.

This instrumented browsing process can be stopped after a request is fully loaded. Once stopped, the browsing session is exported to an XML document containing a detailed account of the browsing session. This document serves as the input to the next phase of the CookieCruncher test process—test input generation.

B.4 Test Input Generation

The test input generation phase of the testing process transforms a browsing report into a testing report. This process involves the processing of a browsing session, the creation of seeding vectors for each test request, and the use of Evolutionary Adaptive Random (eAR) testing for the

generation of each test suite. The three-step process was implemented using a Ruby script to integrate the three processes and relied on a C++ implementation of eAR testing for test-case generation.

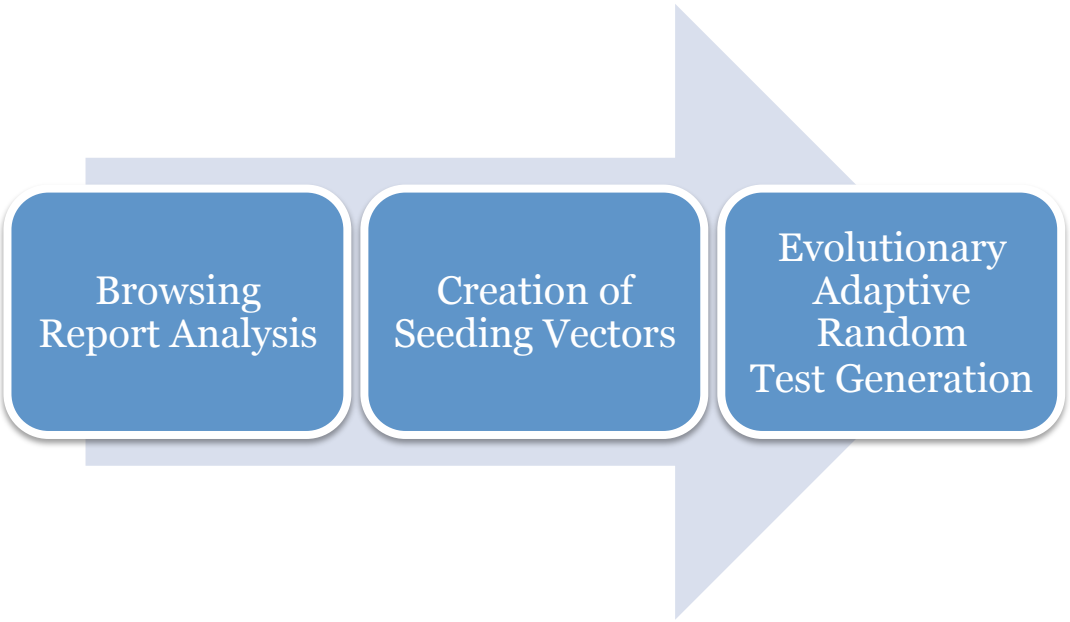


Figure B-4. CookieCruncher's Test Generation Process

B.4.1 Browsing Report Analysis

Browsing reports contain a statement of a series of HTTP requests and the cookie collections present at each request. The processing of this document involves the creation of a global set of cookies, *GC*. This global cookie collection is defined as

$$GC = \bigcup_{i=0}^{n} C_i, \tag{B-3}$$

where C_i is the cookie collection of the i^{th} test request in the report, and n is the number of test requests defined in a browsing report. The global list of cookies combined with the pre-existing cookies collections C_i is the basis for the creation of seeding vectors.

B.4.2 Generation of Seeding Vectors

The generation of seeding vectors is performed through the use of an automated script that is based upon the seeding vector definitions

provided in Chapter 5.3 and summarized in Section B.1.2. The output of this process is a collection of seeding vectors for each test request in the browsing report, and will be used as the basis for test input generation.

B.4.3 Evolutionary Adaptive Random Testing

The evolutionary adaptive random testing adaptations and definitions from Chapter 6.4.1 were adopted within CookieCruncher.

B.5 Test Execution & Evaluation

Within the CookieCruncher testing harness, the test execution and evaluation phases of the testing process are combined into a single process. This combination allows for the evaluation of each test result natively within the Firefox browsing environment, providing the novel ability for the automated evaluation of a fully rendered HTML document. The test execution and evaluation phase is followed the five-step process outlined in Figure B-5.

The first three steps within the process are identical to those outlined in Section B.1.3. Essentially, the preconditions (a series of requests required to ensure test system state) are resolved, the cookie collection is modified to reflect the value of the test vector, and the final page is requested. The output from these three steps is a fully rendered HTML document within the browsing environment, having requested all embedded contented including images, third-party content, and JavaScript components. The final document is then evaluated against the built-in testing oracles and the testing result is stored. Once a test case has been loaded, evaluated and recorded the testing framework restarts the process for the next test case.

Figure B-5. Test Evaluation & Execution Cycle

B.5.1 Testing Oracles

An automated evaluation of each test case requires the definition of a built-in oracle that can be used to detect the presence of a fault within the underlying system under test. To fulfill this requirement, the results of two specific test-vectors were selected as the basis against which all results would be subsequently measured. As indicated in Chapter 5.4.2, a vector representing the original cookie collection present when a test request was recorded, and the zeros vector, were selected as test oracles. These vectors were selected as representative of two distinct and plausible cookie collections—two cookie collections that every web application should handle correctly. The vector representing the recorded cookie collection provides a base-case in which no cookie collection modification has occurred. Essentially this vector represents the request recorded during the instrumented browsing session. The zeros vector, on the other hand,

represents a direct contrast to the recorded vector. The zeros vector presents the application with a browsing environment devoid of cookies; essentially this is equivalent to accessing the page from a browser with cookies disabled. In practice, these two vectors have been found quite effective at capturing the valid outputs related to cookie collection modification within real-world web applications.

B.5.2 The Tree, Context, and Composite Similarity Metrics

The selection of testing oracles is a fundamental component of the testing harness; of equal importance is the method by which testing results are compared against the oracles. Within CookieCruncher, two similarity coefficients were selected as a basis for test result evaluation. The tree similarity and context similarity coefficients, first selected for HTML page difference detection by Yue, Xie and Wang (2007), have been adopted for use within CookieCruncher. A detailed discussion of each of the metrics and the composite similarity metric is provided in Chapter 5.4.3.

B.5.3 Test Result Interpretations

For each test case executed and evaluated by CookieCruncher, a total of six results were returned: three (tree, context, and composite similarity) for each of the two testing oracles the system was evaluated against. All six of the results were recorded and reported by CookieCruncher, however, the result with the highest composite similarity was chosen to represent the test case as the final result. The testing oracle to which the test output was most similar was assumed to be the result for a given test case. This assumption, valid for any test case not containing a fault, results in the presentation of the test result closest to a composite similarity value of one. For test cases for which a fault was uncovered, the result and oracle against which the output had the highest composite similarity coefficient were reported, essentially providing an indication of which oracle the output was most similar to. In practice this indication was found to be sufficient for the detection of defects, and the test oracle to which the

output was most similar was found to be of particular interest when assessing the underlying error responsible for the defect.

B.6 Testing Hooks & Application-Specific Considerations

The testing of real-world web applications required the definition of application specific customizations to accurately reproduce tests. These application specific customizations, subsequently referred to as testing hooks, were necessary for all of the web applications that CookieCruncher was evaluated against. In the evaluation of the testing framework, two specific types of testing hooks were implemented: hooks that occurred once-per-test case; and hooks that occurred once-per-request within the test case.

The most common test hook encountered required a single HTTP request sent to a server-side script, resetting the backend database to a known testing state. This testing hook, an example of a once-per-test hook, was common to all applications for which CookieCruncher was evaluated against. A once-per-test testing hook involves an action, typically a specifically crafted HTTP request, being executed once before the pre-conditions for a specific test case are resolved. The use of this type of testing technique requires the creation of an associated server-side testing support script that is capable of performing the required task called for by the test hook.

Testing hooks were also required to be executed in-between requests to the system under test. These hooks were performed before each request within a test case, including those specified within the pre-conditions. These hooks where found in two varieties:

- The definition of a timeout value; and
- The manipulation of input values based upon the current server response.

A timeout value, specified in milliseconds, was required for applications that employed anti web-robot techniques. The presence of a timeout once-

per-request testing hook was found to successfully circumvent these security measures; however this technique did introduce a period of delay within each test case, increasing the overall amount of time required to run each test suite. These delays could be avoided through the disabling of the anti web-robot measures within the application, which is recommended for large-scale system testing.

The most common once-per-request testing hook involved the retrieval of specific variables from an HTTP response document, and the subsequent modification of a variable within the subsequent HTTP request. These hooks were implemented within CookieCruncher through the use of tester-specified regular expressions, providing an application- and variable-specific interface through which these testing hooks could be defined. In practice these once-per-request testing hooks were required to find specific variable=value pairs within a current cookie and/or the HTML form elements. These values, extracted from the response document, were then implanted into the subsequent request through the values present in the GET, POST, COOKIE, and REFERRER variables. It was found that without these testing hooks, the test cases recorded could not be reliably executed. The definition of these testing hooks required manual analysis and was not automated.

B.7 Extendibility and Future Plans

CookieCruncher currently exists as a stand-alone testing framework for web applications. However, CookieCruncher has been developed with interoperability as a fundamental goal of the system—specifically the XML interface between the Instrumented Browsing and Test-Input Generation components. This interface, fundamental in supplying the test input generation module with an instrumented browsing session, is thought to be adaptable to a wide-array of currently defined web testing methodologies. Through the adaptation of this interface, it is believed that the testing framework can process test cases derived from other web

application testing strategies, providing an efficient and cost-effective method for applying cookie collection testing upon a set of pre-existing test cases.

For example, consider the testing data collected through the analysis of the user-session data as prescribed by Elbaum, Rothermel, Karre, and Fisher (2005). This web application testing strategy employs data mining techniques on the data stored within HTTP server logs to extract test cases based upon the identification of user-sessions. The data extracted from the server logs is in the form of URL sequences (Elbaum et al., 2005) and is very similar to the test cases recorded by the Instrumented Browsing component. This form of testing has been shown to complement current web application white-box testing techniques, and it is believed that the testing data extracted from HTTP server logs can form a basis for the input of test cases into CookieCruncher, allowing for the further amalgamation of web testing techniques.

B.8 Summary

This appendix presented the testing harness CookieCruncher, specifically developed to implement an automated evolutionary adaptive random cookie collection testing strategy for web applications. All facets of the testing process and testing harness were discussed, detailing the specific design decisions associated with each component of the testing system, and providing a clear explanation of the techniques and algorithms implemented within each component of the test harness.

Finally, a discussion of the adaptability and interoperability of the testing harness was presented. The test harness has been developed to be adaptable to a wide array of web applications, and it is believed that the testing harness can itself be adapted to accept input from a wide array of current and future web application testing strategies.

Appendix C

Testing Six Real-World Web Applications

The following appendix will provide a summary of the testing activities preformed on each of the testing applications. A basic description of the application and the testing environment in which it was executed, followed by a summary of the use-cases analysis and test suite generated will be presented for each application. Finally, the application-specific testing modifications (testing hooks) required for the execution of each test case is presented.

C.1 BugTracker.net

BugTracker.net is a web application for the tracking of program defects and customer support issues. According to the developer, Bugtracker.net is used by development and customer support teams around the world (Trager, 2010). BugTracker.net allows users to manage large lists of issues and/or defects. Each issue is configurable and can be associated with multiple users and user groups from both within and outside a particular organization.

BugTracker.net is an open source application that is written using ASP.Net and C# and is implemented on top of a server running Microsoft IIS and Microsoft SQL Server. This application was executed within a test environment running Windows Server 2008 (version 6.0.6002), Microsoft ISS (version 7.0.6000.1638), Microsoft SQL Server 2008 (version 10.0.2531.0), and the Microsoft .NET framework (version 2.0.50727.400).

BugTracker.net was the only testing application that used AJAX to fetch data from within the browser. The use of asynchronous requests to fetch XML data was handled within the testing tool without modification. Essentially the tool recorded each AJAX request and allowed for the testing of both the application state that sent the AJAX request and each individual AJAX request.

C.1.1 Decision Tree & Use-Case Descriptions

A summary of the use-case analysis from which the testing suite was derived for BugTracker.net is provided in Table C-1. This analysis is by no means complete and is only intended to access a wide range of system components; it is not a complete definition of the system.

Table C-1. BugTracker.net Decision Tree & Use-Case Description

Decision Tree	Brief Use-Case Description
I. Administrator Log In	*A user logs into the system with administrator privileges, performs administrator tasks, and then logs out of the system.*
A. Reports	*An administrator can view the data stored within the application through a series of reports.*
1. Tickets by User	*An administrator can browse reports sorted by username.*
2. Tickets by Category	*An administrator can browse reports sorted by category.*
B. Add a user	*An administrator can add a user to the system.*
C. Delete a user	*An administrator can remove a user from the system.*
D. Add/Delete an Organization	*An administrator can add or delete organizations from the system.*
E. Add/Delete a Category	*An administrator can add or delete a issue category from the system.*
II. Regular User Log In	*A user logs into the system without administrative privileges, performs user tasks, and then logs out of the system.*

Decision Tree	Brief Use-Case Description
A. Browse Tickets	*A user can browse tickets by a variety of criterion.*
1. All	*A user can browser all of the tickets within the system.*
2. Tickets with Bugs	*A user can browse tickets that have an associated bug.*
3. Checked in Tickets	*A user can browse all of the tickets that have been checked into the system.*
4. Open Tickets	*A user can browse all of the tickets that have not been closed.*
5. Open & Assigned to Me	*A user can browser all of the tickets that are open and have been assigned to himself.*
B. Change a ticket	*A user can modify a ticket for which he has permission.*
1. Check in New Ticket	*A user can check in a new ticket.*
2. Close a New Ticket	*A user can close a new ticket.*
3. Close a Checked In	*A user can close a checked in ticket.*
4. Reopen a Closed Ticket	*A user can reopen a ticket that has been closed.*
C. Add a New Ticket	*A user can add a new ticket to the system*
D. Assign a Ticket	*A user can assign a ticket to another user or himself.*
E. Link Tickets	*A user can link a ticket to another pre-existing ticket*
F. Search Tickets	*A user can search the system for bugs based on a number of different criterion.*
1. Who Reported	*A user can search the system for tickets based upon the user who reported the ticket.*
2. Category	*A user can search the system for tickets based upon the category assigned to the ticket*
3. Status	*A user can search the system for tickets based upon the status of the ticket.*

Decision Tree	Brief Use-Case Description
4. Multiple Fields	*A user can search the system for tickets based upon a combination of multiple criterion such as reporter, category, and/or status.*
G. Change Settings	*A user can change their personal settings.*
III. Multiple Users	*A system administrator logs into the system and then logs out. Subsequently a non-administrator logs into the system and then logs out.*

The use-cases were implemented through a series of eleven instrumented browser sessions (output from the instrument browsing component of CookieCruncher), and after processing resulted in a test suite of 1,440 tests.

C.1.2 Testing Hooks

The four testing hooks required to execute the BugTracker.net test suite derived in Section C.1.1 are summarize in Table C-2.

Table C-2. Bug Tracker.net Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*
Replace EventValidation	Each Request	*Before each request within a test case the value of the EVENTVALIDATION variable within the POST, GET, REFERRER and COOKIE variables.*
Replace ViewState	Each Request	*Before each request within a test case the value of the VIEWSTATE variable within the POST, GET, REFERRER and COOKIE variables.*
Replace Timestamp	Each Request	*Before each request within a test case the value of the snapshot_timestamp variable within the POST, GET, REFERRER and COOKIE variables.*

C.2 e107

e107 is a web content-management system allowing organizations to produce a efficiently manage a content-based web site. The application has been downloaded over 1.3 million times since 2002 (SourceForge.net, 2010a). The application is written in PHP and uses MySQL as a backend database. Testing was preformed using an Apache HTTP server (version 2.2.14) running PHP (5.3.1) with a backend MySQL database (5.1.41).

C.2.1 Decision Tree & Use-Case Descriptions

A summary of the use-case analysis from which a testing suite was derived for e107 is provided in Table C-1. This analysis is by no means complete and is only intended to access a wide range of system components; it is not a complete definition of the system.

Table C-3. e107 Decision Tree & Use-Case Description

Decision Tree	Brief Use-Case Description
I. Guest User	*A user can browse the site without being registered or logging in.*
A. Browse Site	*A guest user can look at the various news articles.*
B. Submit an Article	*A guest user can submit a news item for publishing. If he does so, he will be asked to log into the site and will require a valid username & password.*
C. Create a User	*A guest user can signup for the site to become a registered user.*
II. Non-Privileged User	*A user can log into the site with a valid username and password and gain access to specific portions of the site. The user has the option to be remembered by the site.*
A. Browse Profile	*A registered user can browse and modify his user profile.*
B. Submit an Article	*A registered user can submit a news item for publishing.*
C. Change Settings	*A registered user can change his settings.*

Decision Tree	Brief Use-Case Description
D. Homepage	*A registered user has his own specific home page*
E. Vote in Poll	*A registered user can participate in the poll by voting and posting comments.*
F. Post a Comment	*A registered user can post a comment related to any news item within the site.*
G. Browse Other Members	*A registered user can access the profiles of other members.*
III. Administrative User	*An administrator can log into the site with his username and password and provide access to the non-privileged user features and the administrator features. The administrator has the option to be remembered by the site.*
A. Admin Area	*An administrator can access the admin area of the system through which he can access the administrative controls of the system.*
1. News	*An administrator can modify the news articles present within the site.*
a) View Submitted Articles	*An administrator can browse the news articles submitted to the site by registered users.*
b) Publish Articles	*An administrator can choose to publish an article submitted to the site.*
c) Reject Articles	*An administrator can reject an article submitted to the site.*
2. FrontPage	*An administrator can modify the content of the front page of the site.*

The use-cases were implemented through a series of six instrumented browse sessions (output from the instrument browsing component of CookieCruncher), and after processing resulted in a test suite of 1,024 tests.

C.2.2 Testing Hooks

One testing hook was required to execute the e107 test suite derived in Section C.2.1 is summarize in Table C-4.

Table C-4. e107 Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*

C.3 GeekLog

GeekLog is a content-management system "with support for comments, trackbacks, multiple syndication formats, spam protection, and all the other vital features of such a system" (GeekLog, 2010). The application is written in PHP and uses MySQL as a backend database. Testing was preformed using an Apache HTTP server (version 2.2.11) running PHP (4.4.9) with a backend MySQL database (5.1.45).

C.3.1 Decision Tree & Use-Case Descriptions

A summary of the use-case analysis from which a testing suite was derived for GeekLog is provided in Table C-5. This analysis is by no means complete and is only intended to access a wide range of system components; it is not a complete definition of the system.

Table C-5. GeekLog Decision Tree & Use-Case Description

Decision Tree	Brief Use-Case Description
I. Administrator	*A user logs into the system with administrator privileges, performs administrator tasks, and then logs out of the system.*
A. Submissions	*An administrator can browse stories submitted by users and publish or reject the stories.*
B. Block Management	*An administrator can add, re-order, or remove blocks from a page within the system.*
C. Poll Management	*An administrator can create, modify and delete polls from the system.*
D. User Management	*An administrator can create, modify and delete users from the system.*

Decision Tree	Brief Use-Case Description
E. Topic Management	*An administrator can create, modify, and delete topics from the system.*
F. Story Management	*An administrator can search, create, edit, and modify a story within the system.*
II. Regular user	*A user logs into the system, interacts with the system and logs out.*
A. Post a comment	*A user can post a comment to any story, news item, or poll within the system*
B. Contribute	*A user can contribute a story to the system.*
C. Search Stories	*A user can search the stories stored within the system based on a variety of criterion including keywords, date, topic, type, and authors.*
D. Browse topics	*A user can select which topic he wants to view, and read the stories contained within the specific topic.*
E. Calendar	*A user can access a private calendar within the system.*
1. Add Event	*A user can add events to his calendar.*
F. My Account	*A user can access the My Account section of the system.*
1. Change Personal Details	*A user can modify his personal details including name, email address, and password.*
G. Poll	*A user can vote and see the results of a poll.*
III. Anonymous User	*A guest user can access parts of the system without logging in.*
A. Poll	*A guest user can vote and view poll results.*
B. Contribute	*A guest user can contribute a story to the system.*
C. Search Stories	*A guest user can search the stories stored within the system based on a variety of criterion including keywords, date, topic, type, and authors.*
D. Browse topics	*A guest user can select which topic he wants to view, and read the stories contained within the specific topic.*

The use-cases were implemented through a series of fourteen instrumented browse sessions (output from the instrument browsing component of CookieCruncher), and after processing resulted in a test suite of 2,232 tests.

C.3.2 Testing Hooks

One testing hook was required to execute the GeekLog test suite derived in Section C.3.1 is summarized in Table C-5.

Table C-6. GeekLog Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*

C.4 phpBB2 & phpBB3

phpBB is a web forum application that allows users communicate over a series of thematically sorted bulletin boards. According to its website, phpBB is "the most widely used open source forum solution" (The PHP Group, 2008). phpBB has been downloaded over 18 million times since 2000 (SourceForge.net, 2010b). The application is written in PHP and uses MySQL as a backend database.

The testing phpBB2 was preformed using an Apache HTTP server (version 2.2.11) running PHP (4.4.9) with a backend MySQL database (5.1.45). phpBB3 was tested on a system using Apache HTTP server (version 2.2.14) running PHP (5.3.1) with a backend MySQL database (5.1.41). phpBB2 required an older version of the PHP framework and so the two system were tested on two different test machines. Despite the five-year gap between the two phpBB releases, phpBB2 and phpBB3 present a very similar feature set, allowing for the same use-case analysis to be used for both versions.

C.4.1 Decision Tree & Use-Case Descriptions

A summary of the use-case analysis from which a testing suite was derived for phpBB2 and phpBB3 is provided in Table C-7. This analysis is by no means complete and is only intended to access a wide range of system components; it is not a complete definition of the system.

Table C-7. phpBB Decision Tree & Use-Case Description

Decision Tree	Brief Use-Case Description
I. Log in as Administrator	*A user logs into the system with administrator privileges, performs administrator tasks, and then logs out of the system. A user can choose to have the system remember that his is logged in. The user interacts with the system, then logs out.*
A. Administration Control Panel	*An administrator can access the administration control panel.*
B. Manage Forums	*An administrator can create, modify and delete forums from the system.*
1. Manage Categories	*An administrator can create, modify, and delete categories from forums within the system.*
C. Manage Groups	*An administrator can create, modify and delete groups of users within the system. Users can be added or removed from groups.*
D. Manage Users	*An administrator can search, create, modify, and delete users from the system. Users can be temporarily banned from the system or deactivated.*
II. Guest User	*A guest user can access parts of the system.*
A. Browse a forums	*A guest user can browse the forums.*
1. Post Reply	*A guest user can post a reply to a comment in a forum. If he does so, he must log into the system with a valid username and password. If he cannot provide the necessary credentials, he is not allowed to post.*
B. Search forum	*A guest user can search the forums based on a number of criterion including keyword, forum, category, and user.*

Decision Tree	Brief Use-Case Description
III. Log in as Regular User	*A user logs into the system, interacts with the system and logs out. A user has the choice to have the system remember that his is logged in.*
A. Browse a forums	*A user can browse the forums.*
1. Post Reply	*A user can post a reply to a comment in a forum.*
1. Create new topic	*A user can create a new topic within a category within a Forum.*
B. Search forum	*A user can search the forums based on a number of criterion including keyword, forum, category, and user.*
C. User Control Panel	*A user can access the user control panel from which he can modify his settings*
1. Manage profile	*A user can access and modify his profile.*
2. Add Friend / Foe	*A user can add another user as his friend or foe within the system.*
2. Board Preferences	*A user can manage his preferences for how he will interact with the system.*
a) Online Status	*A user can set a preference so that other users can not see when he is online.*
D. Delete Cookies	*A user has the option to delete all board cookies from his web browser.*

The use-cases were implemented through a series of seven instrumented browse sessions (output from the instrument browsing component of CookieCruncher), and after processing resulted in a test suite of 580 tests for phpBB2 and 2,080 for phpBB3. This disparity was due to the large number of cookies used in phpBB3 and the extra testing requests required to navigate through phpBB3.

C.4.2 Testing Hooks

One testing hook was required to execute the phpBB2 test suite derived in Section C.4.1 is summarized in Table C-7.

Table C-8. phpBB2 Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*

Table C-9. phpBB3 Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*
Replace sid	Each request	*Before each request within a test case the value of the sid variable within the POST, GET, REFERRER and COOKIE variables.*
Replace creation_time	Each request	*Before each request within a test case the value of the creation_time within the POST, GET, REFERRER and COOKIE variables*
Replace form_token	Each request	*Before each request within a test case the value of the form_toekn variable within the POST, GET, REFERRER and COOKIE variables*
Replace Multipart creation_time	Each request	*Before each request within a test case the value of the sid variable within the POST, GET, REFERRER and COOKIE variables*
Replace Multipart form_token	Each request	*Before each request within a test case the value of the form variable within the POST, GET, REFERRER and COOKIE variables*
Replace sess	Each request	*Before each request within a test case the value of the sess variable within the POST, GET, REFERRER and COOKIE variables*
Replace confirm_key	Each request	*Before each request within a test case the value of the confirm_key variable within the POST, GET, REFERRER and COOKIE variables*

Delay	Timeout	*Before each request within a test case a timeout of 1 second was applied.*

While the use-case analysis and subsequent test cases remained constant between the two versions of phpBB, phpBB3 required nine testing hooks to reliably execute the requests within test cases, summarized in Table C-9. Seven of the nine testing hooks involved the search and replacement of application-specific variables present within a cookie, the request (GET or POST) or contained within the HTML document itself. These seven variables were required to replay each test case, and provided a redundancy layer allowing the application to function without the presence of one or all of the application's cookies.

C.5 phpMyAdmin

phpMyAdmin is a web application that enables users to administer multiple MySQL databases over the Internet. The application has been downloaded over 20 millions times since 2001 (SourceForge.net, 2010c). phpMyAdmin allows users to create and manage MySQL databases providing the ability to create, modify and delete tables, fields and rows.

phpMyAdmin is an open source application that is written in PHP and requires a MySQL database to manage. The testing of phpMyAdmin was preformed using an Apache HTTP server (version 2.2.14) running PHP (5.3.1) with a backend MySQL database (5.1.41).

phpMyAdmin was the only application tested that used HTML frames. The use of frames was handled within the testing tool without modification. Essentially the tool recorded the HTTP request for each frame allowing for the testing of each frame.

C.5.1 Decision Tree & Use-Case Descriptions

A summary of the use-case analysis from which a testing suite was derived for phpMyAdmin is provided in Table C-10. This analysis is by no means

complete and is only intended to access a wide range of system components; it is not a complete definition of the system.

Table C-10. phpMyAdmin Decision Tree & Use-Case Description

Decision Tree	Brief Use-Case Description
IV. Log in	*A user of the managed database can log into the system using his username and password for the database.*
A. Browse Databases	*A user can browse the various databases for which he has permissions within the MySQL server.*
1. Browse Tables	*A use can browse the Tables within a database, viewing the structure of the database and basic statistics such as number of rows and size on disk. A user can browse the data contained within the table.*
a) Manage Fields	*A user can create, rename, modify, and delete a field within a table, provided he has sufficient database permissions. Fields can be of various types including: Text, VarChar, int, DATE, etc.*
2. Manage Tables	*A user can create, rename, modify and delete a table within a database, provided he has sufficient database permissions.*
3. Export Tables	*A user can export the structure and/or data of a table to various formats including: a set of SQL statements, comma-separated values, and an excel worksheet.*
4. Search Tables	*A user can search the tables of a database to produce output that is the conglomeration of one or more tables.*
B. Manage Databases	*A user can create, rename, modify and delete a database, provided he has sufficient database permissions.*
C. Export Database	*A user can export the structure and/or data of a database to various formats including: a set of SQL statements, comma-separated values, and an excel worksheet.*
D. Customize Look	*A user can customize the look and feel of the web interface.*

Decision Tree	Brief Use-Case Description
1. Change Language	*A user can change the language in which the web interface is presented. This change should be persistent across browsing sessions.*
2. Change Font Size	*A user can adjust the size of the text within the web application, specified as a percentage. This change should be persistent across browsing sessions.*
3. Change Theme	*A user can change the theme used to render the look of the system. This change is persistent across browsing sessions..*

The use-cases were implemented through a series of seven instrumented browse sessions (output from the instrument browsing component of CookieCruncher), and after processing resulted in a test suite of 21,701 tests.

C.5.2 Testing Hooks

One testing hook was required to execute the phpMyAdmin test suite derived in Section C.5.1 is summarized in Table C-11.

Table C-11. phpMyAdmin Testing Hooks

Name	Type	Description
Reset DB	Once per test	*Once, before each test case is implemented a request was made to a server-side script that reset the database to a known state.*