Concordia University College of Alberta

Master of Information Systems Security Management (MISSM) Program

7128 Ada Boulevard, Edmonton, AB

Canada T5B 4E4

Statistical Analysis of Software Design Error Vulnerability Data

by

NJI, Lionel

A research paper submitted in partial fulfillment of the requirements for the degree of

Master of Information Systems Security Management

Research advisors:

Pavol Zavarsky, Director of Research and Associate Professor, MISSM

Dale Lindskog, Assistant Professor, MISSM

Statistical Analysis of Software Design Error Vulnerability Data

by

NJI, Lionel

Research advisors:

Pavol Zavarsky, Director of Research and Associate Professor, MISSM

Ron Ruhl, Director and Associate Professor, MISSM

Reviews Committee:

Andy Igonor, Assistant Professor, MISSM
Dale Lindskog, Assistant Professor, MISSM
Ron Ruhl, Assistant Professor, MISSM
Pavol Zavarsky, Associate Professor, MISSM

The author acknowledges the significant contributions to the work by Academic Advisors and Review Committee Members and gives the right to Concordia Univeristy College to reproduce the work for the Concordia Library, Concordia Websites and Concordia MISSM classes.

A Thesis for the Degree of Master of Information Systems Security Management

Statistical Analysis of Software Design Error Vulnerability Data

Nji Lionel Nji
M.Sc. Physics, ADDIS, B.Sc. Physics and Computer Science

Information Security is my responsibility, and so it's yours.

The security depends not only on how strong the lock is but also on how well you protect the key that opens the lock.

Concordia University College of Alberta, 7128 Ada Boulevard, Edmonton AB T5B 4E4, Canada

Dedication

This thesis is dedicated to my lovely daughter, Andin Elvevoll-Nji and her wonderful mum, Lena Susanne Elvevoll.

Acknowledgement

I am grateful to Dr. Dale Lindskog and Dr. Pavol Zavarsky, my academic supervisors, and to Ron Ruhl, the Director of the Information Systems Security Management Program. Your advice, reviews and feedback were very helpful. I benefited a lot from your experience.

My gratitude goes to Andin Elvevoll-Nji, my lovely daughter. Your lovely and timely chats were very sweet and supportive. In fact, they were and remain priceless. Thank you, indeed, Lena Susanne Elvevoll, the wonderful mother of my princess. You tirelessly took care of her and this gave me the much needed peace of mind and concentration. Words alone can not express my appreciation for the quality of work you have done and are currently doing.

My appreciation goes to Mary Gallivan (Landlady). You gave me the much needed shelter, home and supports. Above all, you gave me a family in Edmonton. Thank you for packaging my lunch box. You did a great job.

My gratitude also goes to my brothers (Jomia Denis Nji, Bambot Wilfred Nji) and my sisters (Ndungbowo Eevlyn Nji, Bibiana Bisona Nji and Mirabel Mbewo) in Cameroon for their continuous support. My cousins, Dr Bigoga Jude Daiga and Nyema Lilian Bigoga are not left out.

I can not forget you, mummy Andin Paulin and Ba Nji Ndimafon. You both work very hard to lay a solid foundation for today's work. Apart from being the best parents in the world, both of you are the best teachers of all times. Thank you, indeed.

I would like to thank my former teacher of the former After-Degree program in Information Systems Security and now my work supervisor, Robert Martin. Your continuous support can not be forgotten. Thank you for the reviews and feedback. I would also like to thank the members of the Information Policy and Compliance Unit, Alberta Health and Wellness, for their moral support.

I would like to particularly offer my appreciation to Anuj Tewari, my former classmate of the former After-Degree program in Information Systems Security and now my colleague at Alberta Health and Wellness, for being there for me whenever I needed his support.

To finish, I would like to thank the authorities of Concordia University College of Alberta, Canada, for offering such an innovative program in Information Systems Security Management. My appreciation for the Athletic and Jimmie Condon Scholarships awarded to me. The money was well spent on my tuition.

Abstract

The research studied the trend followed by software design error vulnerabilities. Statistical analysis methods were used to analyze software design error vulnerability data that were collected after January 1988 and before January 2007. The source of the data was the U.S Government National Vulnerability Database (NVD). The purpose of this research was to obtain an understanding and hence attempt to explain the trend followed by designed error vulnerabilities during the above specified period.

It was found out that more that half the software vulnerability data collected and stored in the NVD between 1988 and 2006 were high severity vulnerability data. Approximately, one third of the data stored in the NVD within this period were software design error vulnerability data. The majority of the software design error vulnerabilities were of high severity. Software design error vulnerabilities generally exhibited a decreasing trend between 1998 and 2006.

Design error vulnerabilities that targeted information confidentiality, integrity and availability fluctuated in the cause of the years and generally exhibited an increasing trend from 1988 to 2006. The research also revealed that most of the software design error vulnerabilities increasingly and particularly targeted information confidentiality. A probably reason being that there has been a gradual shift in the motives behind vulnerability exploitation towards financial gain.

Finally, it was found out that most of the design error vulnerabilities were exploited remotely. This could partly be attributed to the rapid growth and the influence of the Internet. The mode of exploitation whereby a target accesses an attacker's resource was least utilized for exploiting these vulnerabilities. The variation of the remote of exploitation of the design error vulnerabilities during the course of the years was found to be inversely proportional to the local mode of exploitation.

Table of Content

1.0 Introduction

Information security is the protection of the confidentiality, integrity and availability of information assets. A piece of information is kept confidential if and only if it is disclosed only to those who have the need-to-know. The integrity of information refers to the completeness and the correctness of the information. By availability, the information is readily available to authorized users when needed. Design error vulnerabilities, a problem that arises from a fundamental mistake in a software design, if successfully exploited, can lead to breaches of confidentiality, integrity and/or availability.

A large number of software vulnerabilities are reported every year (Table 1.0) by software vendors and information security watchdogs such as NIST (National Institute of Science and Technology). These vulnerabilities take different forms and could include buffer overflow, race conditions, and configuration vulnerabilities.

Table 1.0: General Statistics

Vuln. / # of Vuln.	Input validation error	Access validation error	Exceptional condition error	Race condition	Configuration error	Design error	% of Design error
# of Vuln. in 2006	4765	272	324	38	67	879	14
# of Vuln. in 2005	3224	298	438	52	94	908	18
# of Vuln. in 2004	1389	198	323	24	78	471	19
# of Vuln. in 2003	697	124	182	24	53	340	24
# of Vuln. in 2002	985	162	199	27	118	586	28
# of Vuln. in 2001	815	141	160	50	89	461	27
# of Vuln. in 2000	372	176	121	21	87	171	18

Input validation error (e.g., boundary condition error and buffer overflow): This error occurs when a functional module fails to properly validate the input it accepts from another module or process. Failure to validate the input may cause the module accepting the input to fail or it may indirectly cause an interacting module to fail [14].

Access validation errors are errors, which result from incorrectly specified constructs. [14] Exceptional condition error can include unanticipated return codes and failure events.

A race condition or race hazard is a flaw in a system or process whereby the output of the process is unexpectedly and critically dependent on the sequence or timing of other events. The term originates with the idea of two signals racing each other to influence the output first. Race conditions can occur in poorly designed electronics systems, especially logic circuits, but they can and often do also arise in computer software [11].

Configuration errors consist of faults introduced after software has been developed and faults introduced during the maintenance face of the software development life cycle. A static audit

analysis of a system can reveal a majority of configuration errors [14]. This may also result when software is adapted to a new environment.

A design error vulnerability is a problem that arises from a fundamental mistake in the design of a software.

Software vulnerabilities have widespread impact and can potentially cause billions of dollars in downtime and disruptions to firms. Every year, the Computer Security Institute [10] conducts a survey on computer crime and security. According to the 2006 report, based on 331 respondents, a significant amount of dollars was lost by firms due to computer crimes or exploited vulnerabilities as depicted in Table 1.1.

A large amount of vulnerability statistics data is publicly available. This research studies the trend followed by design error vulnerability by statistically analyzing software design error vulnerability data that were collected after January 1988 and before January 2007. The purpose of this analysis was to try and obtain an understanding and hence attempt to explain the trend followed by designed error vulnerabilities during this specified period. Consequently, the results obtained were used in predicting the evolution of information security activities such as the development of secure software, between 1988 and 2006 and also make predictions for future trends.

This document contains five chapters. Chapter Two provides the fundamentals of software vulnerability. The distinction between software vulnerability and software bugs is addressed. Ways by which vulnerabilities could be exploited are discussed. The chapter closes by classifying vulnerabilities into three groups vis-à-vis design error, configuration and operational vulnerabilities.

Table: 1.1: Statistics of dollar amount lost due to exploited vulnerabilities.

Type of Attack	**Dollar Amount Loss (USD)**
Virus contamination	15,69,460
Unauthorized access to information	10,61700
Denial of service	2,922,2010
Phishing	647,510
System penetration by outsiders	758000
Website defacement	162,500
Password sniffing	161,210
Exploit of organisational DNS server	90100
Total Losses for 2006	52,494,290

Chapter Three discusses the research methodology used during the research. For a better understanding of the source of the data used in this research and how the data was collected, a brief description of the National Vulnerability Database is presented. The various steps followed during data acquisition are provided. The close of the chapter presents the collected data sets.

In Chapter Four, the statistical methods that were used for data analysis are described. The results obtained are presented and the analyses of those results are provided. The

results are split into five major sections. The first section presents the overall picture of software vulnerabilities that were reported and registered in the NVD after January 1988 and before January 2007. The second section presents results pertaining to design error vulnerabilities. Finally, the last three sections provide results for design error vulnerabilities that allowed unauthorized disclosure of information, unauthorized modification of information and disruption of service, respectively.

Chapter Five attempts to interpret the results presented in chapter four in the context of information system security. The chapter is composed of nine sections. Section 5.1 looks at the interpretation of the results in relation to the changes in the security industry. Section 5.2 provides an interpretation of the results in relation to the financial impact suffered by software vendors when a vulnerability is reported in their products. In Section 5.3 an interpretation to the results is provided by considering the influence of security experts when t security concerns are made public. Section 5.4 looks at the interpretation of the results from the view point of the change in the motives of attackers. Section 5.5 looks at the influence of the Internet. Section 5.6 and 5.7 relates the results to changes to computer usage and operating system, and the need to meet legislative or regulatory obligations, respectively. Section 5.8 looks at the part played by user of secure coding standard by software vendors. Section 5.9 provides some limitations to the use of the NVD. Chapter Six concludes the research work.

2.0 Software Vulnerability Basics (7)

A software [8] is simply a set of written coded commands that tells a computer what tasks to perform, e.g. Windows 2003 operating system, Microsoft Office, Photoshop, etc. Computer software is utilized by different organizations for performing various tasks. In fact, organizations are so dependent on computer software that it has became an integral part of many business processes. However, many questions are usually raised about the security of computer software and the implication of the presence of security holes in them.

This chapter provides the fundamentals of software vulnerabilities that will facilitate the understanding of the research materials presented in this thesis. It is divided into three sections. Section 2.1 attempts to make a distinction between a software vulnerability and a software bug. Section 2.2 identifies and explains three different ways or modes through which vulnerabilities could be exploited. In Section 2.3, an attempt is made to classify vulnerabilities into three major groups.

2.1 Software Bug and Software Vulnerability

A software bug [7] is a mistake or error in a program that results in unexpected and typically undesirable outcomes. A program that is relatively free of bugs is reliable [7]. It rarely fails on users and handles exceptional conditions gracefully. It is written "defensively" so that it can handle uncertain environment and malformed inputs.

On the other hand, a software vulnerability [7] is a specific flaw (error) in a piece software that allows an attacker to do something malicious such as expose or alter sensitive information, disrupt or destroy a system or take control of a computer system or program. The Organization of Internet Security [9] defines vulnerability as a flaw within a software system that can cause it to work contrary to its documented design and could be exploited to cause the system to violate its documented security policy. A security policy is simply a list of what are allowed and what are forbidden. For instance, a security policy might state that a particular software system must accept at least an eight-character password for authentication. If there is problem that allows the system to accept a six-character password, then that security policy has been violated. Generally, a vulnerability may either allow an attacker to [12]:

- Execute commands as another user.
- Access data that is contrary to the specified access restrictions for that data.
- Pose as another entity.
- Deny service to authorized users.

In general, software vulnerabilities can be thought of as a subset of software bugs. A bug must have some security relevant impact or properties for it to be considered a security vulnerability. Consequently, almost all security vulnerabilities are software bugs but only some software bugs turn out to be security vulnerabilities. In some cases, it is difficult to draw a clear-cut line between software vulnerability and software bug.

For instance, a program that allows a user to edit a critical system file that he or she should not have access to might be operating correctly according to its specification and design. This probably would not fit into the definition of a software bug but would definitely be termed a security vulnerability. A secure program is similar to a robust program. It can repel a focused attack by intruders who attempts to manipulate its environment and input so that they can leverage it to achieve some nefarious end.

2.2 Modes of Vulnerability Exploitation

The process of attacking a vulnerability in a program is known as exploiting [7]. Attackers might exploit a vulnerability by running the program in a clever way such as by altering or monitoring the program's environment while it runs, or if the program is inherently insecure, simply using the program for its intended purpose. When attackers use an external program or script to perform an attack, this attacking program is often called an exploit. An exploit [11] is therefore, a piece of software or sequence of commands that take advantage of a bug or vulnerability in order to cause unintended or unanticipated behaviour to occur on computer software (or hardware). Many exploits are designed to provide super user-level or administrator access to a computer system. However, it is also possible to use several exploits, first to gain low-level access, then to escalate privileges repeatedly until root or administrative privileges are gained. An attacker can exploit a vulnerability [11]:

- Remotely: A remote exploit works over a network and exploits the security vulnerability without any prior access to the vulnerable system e.g. virus attack.

- Locally: A local exploit requires prior access to the vulnerable system and usually increases the privileges of the person running the exploit past those granted by the system administrator.

- When a target accesses an attacker's resource: This is an exploit against client application that requires some interaction with the user. It may be used in combination with social engineering method.

Social engineering is a collection of techniques used to manipulate people into performing actions or divulging confidential information, e.g. Phishing. Phishers attempt to fraudulently acquire sensitive information, such as usernames, passwords and credit card details, by masquerading as a trustworthy entity in an electronic communication. eBay, PayPal and online banks are common targets. Phishing is typically carried out using email or an instant message, and often directs users to a website, although phone contact has been used as well

2.3 Vulnerability Classification

In this section, attempts are made to distinguish between vulnerabilities and classify them. A vulnerability class [7] is collection or set of vulnerabilities that share some unifying features—a pattern or concept that isolates a specific feature shared by several

different software flaws. In other words, vulnerability classes are just mental devices for conceptualizing software flaws. There is no single and clean taxonomy for grouping vulnerabilities into accurate, non-overlapping classes. It is quite possible for a single vulnerability to fall into multiple classes, depending on the code auditor's terminology, classification system, and perspective [7]. Generally, there are three classes of vulnerabilities:

- Design error vulnerabilities
- Implementation vulnerabilities
- Operational vulnerabilities.

2.3.1 Design Error Vulnerability

A design error vulnerability is a problem that arises from a fundamental mistake in the design of a software. With respect to the Software or System Development Life Cycle (SDLC) phases, this fundamental mistake may occur during phase 1, 2 or 3. A SDLC is a framework for describing the phases involved in developing information systems [22]. These types of flaws often occur because of assumptions made about the environment in which a program will run or the risk of exposure that program components will face in the actual production environment. Design flaws are also referred to as high-level vulnerabilities, architectural flaws, or problems with program requirements or constraints.

Based on the SDLC, the design of a software system is driven by a set of requirements. This is a list of objectives a software system must meet in order to accomplish the goals for which it was created. Requirements usually address what a software system has to accomplish, e.g. allow a user to retrieve a transaction file from a server. They can also specify capabilities the software must have, e.g., it must support 1000 simultaneous downloads. From the set of requirements, the design specifications of the software are constructed. This specifications focus on how the software will meet its intended goals. Specifications are the plans for how the program should be constructed to meet the requirements. Typically, they include

- A description of the different components of a software system,
- Information on how the components will be implemented and what they will do,
- Information on how the components will interact.

Specifications could involve architecture diagrams, logic diagrams, process flowcharts, interface and protocol specifications, class hierarchies, and other technical specifications.

As an example, Internet Explorer (IE) prior to IE7 has been the target of the majority of Phishing and malware attacks over the past few years. Many of these attacks have taken advantage of the close integration between the browser and the operating system. The browser and hence malwares inherited elevated privileges from the user and was able to make damaging changes to the host system. IE7 that was released with Microsoft VISTA in November 2006 was rewritten to have reduced integration with the core operating

system and to run with a reduced privilege (privilege escalation exploits based on code injection to other processes in the same session should no longer work) [23].

2.3.2 Implementation Vulnerability

In implementation vulnerability, the code is generally doing what it should, but there is a security problem in the way the operation is carried out. This vulnerability occurs during the implementation stage of the SDLC and it is often carried to the integration and testing phases. These problems can happen if the implementation deviates from the design to solve technical discrepancies. However, technical artifacts and nuances of the platform and language environment in which the software is constructed cause most exploitable situations. Implementation vulnerabilities are also referred to as low-level flaws or technical flaws. Buffer overflows is an example of implementation vulnerability.

A buffer overflow [11] is an anomalous condition where a process attempts to store data beyond the boundaries of a fixed-length buffer. The result is that the extra data overwrites adjacent memory locations. The overwritten data may include other buffers, variables and program flow data and may cause a process to crash or produce incorrect results. They can be triggered by inputs specifically designed to execute malicious code or to make the program operate in an unintended way.

Some previous implementations of Telnet daemons did not cleanse user environment variables correctly, allowing intruders to leverage the dynamic linking features of a UNIX machine to elevate their privileges on the machine. There were also flaws that allowed intruders to perform buffer overflows and format string attacks (caused by the use of unfiltered user input) against various versions of Telnet daemons, often without authenticating at all. These flaws resulted in attackers being able to remotely issue arbitrary commands on the machine as privileged users. Basically, attackers could run a small exploit program against a vulnerable Telnet daemon and immediately get a root prompt on the server [7].

2.3.2 Operational Vulnerability

Operational vulnerabilities are security problems that arise through the operational procedures and general use of a piece of software in a specific environment [7]. Specifically, these can include:

- Configuration issues of the software in its environment or supporting software and computers.
- Automated and manual process issues that surround the system.
- Certain types of attacks on users of the system, such as social engineering and theft.

In terms of the SDLC phases, these vulnerabilities occur during operation and maintenance phases, although they have some overlap into the integration and testing phase. Consider a business software application. One of the enforced password policies for this application is that password expires after 60 days. In order to change expired passwords, each user connects to the server via telnet and is prompted for the old

password. Only after entering the correct old password is the user allowed to enter a new password. Depending on the environment, this process could represent a major operational vulnerability because of the multiple risks associated with using Telnet, including sniffing and connection hijacking. In a nutshell, the operational procedure for changing expired passwords for the application is flawed because it exposes authentication credentials to attackers as telnet traffic is not encrypted.

To conclude this section, the SDLC provides a clean distinction between design and implementation vulnerabilities. However, this is deceptive as it is not usually easy to distinguish between the two. Many implementation vulnerabilities could also be interpreted as situations in which the design did not anticipate or address the problem adequately. On the flip side, one could argue that lower-level pieces of a software system are also designed, in a fashion. In general, when people refer to design vulnerabilities, they mean high-level issues with program architecture, requirements, base interfaces, and key algorithms.

In some situations, it is also not easy to distinguish between operational vulnerabilities and implementation or design vulnerabilities. For instance, if a program is installed in an insecure manner, one could easily argue that it is a failure of the design or implementation. The application is expected to be developed in a manner that it is not vulnerable to these environmental concerns. As a consequence of a lack of a strict distinction between operational vulnerabilities and implementation or design vulnerabilities, operational vulnerabilities can, alternatively, be defined as [7] security issues that results from the unsafe deployment and configuration of software, unsound management and administration practices surrounding software, supporting components such as application and web servers, and direct attacks on the software users.

3.0 Research Methodology

This chapter provides information about the data that was used in conducting this research and the method that was used to collect the data. It is divided into three main sections. The first section provides a brief description of the U.S National Vulnerability Database (NVD) and its main components. Data for this research was retrieved from this database. The second section describes the method that was adopted in order to retrieve data from the NVD. This section also identifies and defines the various variables that were studied during the research. The last section presents and categorizes the data sets that were collected.

3.1 The National Vulnerability Database

NVD is a comprehensive cyber security vulnerability database that integrates all publicly available U.S. Government vulnerability resources and provides references to industry resources. It was created with a unique mission and mandate of providing previously unavailable technical capabilities and to offer needed support for a variety of vulnerability standards. The database is used by the U.S Department of Homeland Security (DHS) to offer official vulnerability information on all known computer vulnerabilities. This information is provided via a fine-grained search engine while integrating all publicly available U.S. Government vulnerability resources. Statistics on the nature of these vulnerabilities are provided through the NVD statistics engine. This service allows users to assess changes in vulnerability discovery rates within specific products or within specific types of vulnerabilities. The NVD is a product of the NIST (National Institute of Standard and Technology) Computer Security Division and is sponsored by the U.S Department of Homeland Security's National Cyber Security Division.

The NVD is completely based upon the Common Vulnerabilities and Exposures (CVE) standard vulnerability dictionary and it is synchronized with CVE - a list of standardized names for vulnerabilities and other information security exposures. An exposure is a state in a computing system (or set of systems), which [12]:

- Allows an attacker to conduct information gathering activities.
- Allows an attacker to hide activities.
- Includes a capability that behaves as expected, but can be easily compromised.
- Is a primary point of entry that an attacker may attempt to use to gain access to the system or data.
- Is considered a problem according to some reasonable security policy.

3.1.1 Common Vulnerability Scoring System (CVSS)

The NVD depends completely upon CVE. The NVD provides Common Vulnerability Scoring System (CVSS) scores for all CVE vulnerabilities and integrates Open Vulnerability Assessment Language (OVAL) queries. OVAL [12] is an international,

information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. It is a collection of XML (Extensible Markup Language) schema for representing system information, expressing specific machine states, and reporting the results of an assessment.

CVSS is an open standard for scoring vulnerabilities. It is used for assigning vulnerability severity. The CVSS scores within NVD can be used to prioritize how an organization handles vulnerabilities. For example, vulnerabilities with scores of 7 and greater should be addressed with great rapidity (possibly through an expedited change management process) while vulnerabilities with scores of less than 3 can usually be addressed through the regular patching process. In addition, one can click on a CVSS score within the NVD to bring up a scoring calculator that will allow users to understand how the score was created and to customize the score for the organization.

Sometimes, not all the information needed to create CVSS scores may be available. This typically happens when a vendor announces a vulnerability but declines to provide certain details. In such situations, the NVD analysts assign CVSS scores using the information that is available. The NVD provides severity rankings of “Low”, “Medium”, and “High” in addition to the numeric CVSS scores but these qualitative rankings are simply mapped from the numeric CVSS scores:

- Vulnerabilities are labeled “Low” severity if they have a CVSS base score in the range 0.0-3.9.
- Vulnerabilities are labeled “Medium” severity if they have a base CVSS score in the range 4.0-6.9.
- Vulnerabilities are labeled “High” severity if they have a CVSS base score in the range 7.0-10.0.

The NVD is updated immediately with raw vulnerability information whenever a new vulnerability is added to the CVE standard dictionary of vulnerabilities. These raw vulnerabilities are then analyzed by NVD analysts and augmented with vulnerability attributes (e.g. vulnerable version numbers) within hours on normal U.S. government business days. The following impact types are assigned to the NVD vulnerabilities:

- Allows unauthorized disclosure of information.
- Allows unauthorized modification of information.
- Allows disruption of service.
- Security protection: provides unauthorized access. This refers to getting some sort of general privileges in the application or entire computer (e.g., getting "root access" or an application account). This could be user level access to the operating system or getting administrator privileges.

The NVD only records what impact types a vulnerability directly allows. Many vulnerabilities give an attacker general privileges on a computer or within an application (e.g., the ability to execute code). With that privilege, an attacker can always violate confidentiality, integrity, and availability. This is not denoted within NVD for two reasons:

- It is obvious that this is true.
- Some vulnerabilities (usually buffer overflows) allow direct violation of confidentiality, integrity, or availability (usually availability) and then also allow one to gain general "unauthorized access".

At the time of this research, all the NVD data were freely available via XML feeds. There were no fees, licensing restrictions, or even a requirement to register. As of May 2, 2007, there were 24135 CVE Vulnerabilities, 90 US-CERT (Computer Emergency Response Team) Alerts, 1918 US-CERT Vulnerability Notes and 2966 Oval Queries.

3.1.2 NVD Statistic Engine and Data Acquisition

The NVD statistics engine is a general-purpose vulnerability statistics generation engine that allows one to generate statistics on vulnerability trends over time. It could be used to graph and chart vulnerabilities discovered within a product or to graph and chart sets of vulnerabilities containing particular characteristics (e.g. remotely exploitable buffer overflows). With the statistical engine, one can track particular products or vendors as well as sets of vulnerabilities with particular attributes (such as remotely exploitable buffer overflows). The most important usage of the statistics engine is to look at the past history of a product as an indicator to see whether or not it is likely to be vulnerable in the future. The figure below shows the graphical user interface of the statistic engine.

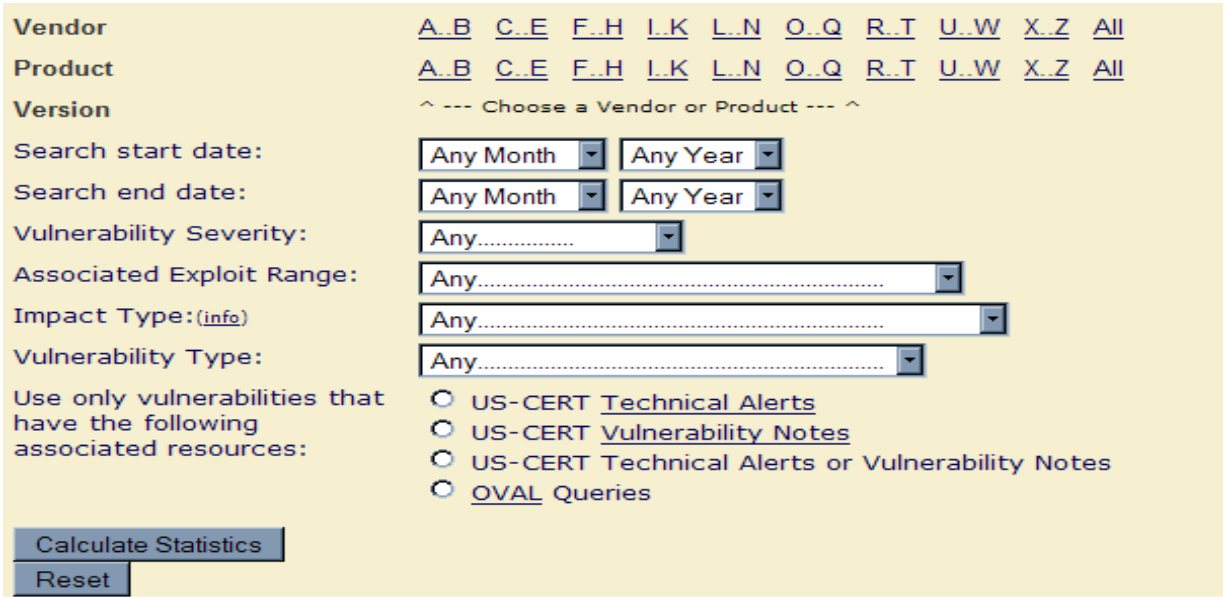


Fig 3.1.2: NVD Statistic Engine.

The statistic engine allows one to choose:

- A vendor on which a search is to be performed, e.g. Apple, Microsoft. As this research is not focused on a particular vendor, this field was therefore irrelevant during data acquisition. It was not taken into consideration during data search.

- A product of the selected vendor above, e.g. AFP server (Apple). As this research is not focused on a particular product, this field was irrelevant during data search and was not considered.

- The version number of the product chosen: NVD may not contain all vulnerable version numbers. Using this option may cause one to overlook vulnerabilities. This field did not play a role during data acquisition for this research.

- A start date for the search: The values for this field ranged from "after January 1988" to "before January 2007". For this research it was set to "after January1988".

- An end date for the search: The values for this field also ranged from "after January1988" to "before January 2007". For this research it was set to "before January 2007"

- The vulnerability severity to be considered during a search. Possible values for this field are: any, high, "medium and high", medium and low. The value "medium and high" was irrelevant for this research. All the others were, however, relevant.

- The associated exploit range: Possible values for this field are: any, remotely exploitable, locally exploitable and "target must access attacker's computer".

- The impact type: This field value could take one of the following values: any, "allows unauthorized disclosure of information", "allows unauthorized modification of information", "allows disruption of service", "provides administrative access", "provides user account access" and "provides other access". Only the first three values were considered for this research.

- Vulnerability types: In addition to the vulnerbility types described under Table 1.0 in the Introduction section of this document, they include:

    - Environmental error: These errors are dependent on the operational environment, which makes them difficult to detect. It is possible that this vulnerability manifest itself when the software is run on a particular machine, under a particular operation system or a particular configuration [14].

    - Race condition: A race condition or race hazard is a flaw in a system or process whereby the output of the process is unexpectedly and critically dependent on the sequence or timing of other events. The term originates with the idea of two signals racing each other to influence the output first. Race conditions can occur in poorly designed electronics systems, especially logic circuits, but they can and often do also arise in computer software [11].

    - Design error: Design error was the only consideration during this research and is described above.

    - Other error. This could be synchronization error (this is introduced because of the existence of a timing window between two operations or fault that results

due to improper or inadequate serialization of operations), origin validation error (this is introduced when an object accepts input from an unauthorized subject or when the system fails to properly or completely authenticate a subject), etc [14].

- Vulnerabilities that contain one or none of the following associated resources: US-CERT technical alerts, US-CERT vulnerability notes, US-CERT technical alerts or vulnerability notes or OVAL queries. None of these was considered during this research.

Based on the classification of vulnerabilities presented in Chapter Two, exceptional condition error and race conditions are design error vulnerabilities. Input validation error is an implementation vulnerability while configuration error and environmental errors are operational vulnerabilities.

3.2 The Data

This section presents the data sets acquired from the NVD. The rationale is to make the data available to researchers who, for whatever reason, may want to perform the same research again or analyze the same data using different methods. In the tables below, the absence of vulnerability statistics for a particular year simply means that the search engine did not return any data corresponding to that year. Also, by considering percentage values, errors associated with sheer volume of sold software are eliminated. Without loss of generality, the percentage values were calculated to the nearest whole numbers. Five major data sets were retrieved from the NVD database. They include:

- An overall vulnerability statistics for all vulnerabilities that occurred after January 1988 and before January 2007.

- Design error vulnerability statistics for all design error vulnerabilities that occurred after January 1988 and before January 2007.

- Design error vulnerability statistics for all design error vulnerabilities that occurred after January 1988 and before January 2007 and allowed unauthorized disclosure of information (breach of confidentiality).

- Design error vulnerability statistics for all design error vulnerabilities that occurred after January 1988 and before January 2007 and allowed unauthorized modification of information (breach of integrity).

- Design error vulnerability statistics for all design error vulnerabilities that occurred after January 1988 and before January 2007 and allowed disruption of service (breach of availability).

Each of the above five major data sets was sub-divided into three sets base on the severity (high, medium and low) of the design error vulnerabilities. Each severity-based data set was again divided into three subsets based on the three modes of exploitation described above.

The percentage values for each "daughter" data set were calculated relative to the "mother" data set. If a data set is split into subsets, the original data set is the mother data set and the subsets are the daughter data sets. For instance, the design error vulnerability data set is a daughter data set of the overall vulnerability data set which is the mother data set.

3.2.1 Overall Vulnerability Statistics

To obtain this data set, the "search start date" field of the statistic engine was set to "after January 1988" while the "search end date" field was set to "before January 2007". The entries of all other fields were set to "any". No particular vendor (and hence product and version) was considered in this research. Since this is an overall vulnerability statistics, the percentage values are set to 100.

Table 3.2.1.0: Overall vulnerability statistics

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1992	1991	1990	1989	1988
# of Vuln	6384	4870	2358	1276	1959	1672	1018	918	246	252	75	25	25	13	13	15	11	3	2
Total %	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Keeping the above setup constant, entry of the vulnerability severity field was varied from high through medium to low to obtain severity-based statistics presented in tables 3.2.1.1, 3.2.1.2 and 3.2.1.3. For each table, the percentage of the data that forms the overall statistics is indicated. Within the limit of statistical error, the partial percentage values should add up to approximately 100 (with approximately ±2 difference). The rest of the data sets were obtained by assigning appropriate values to the fields of the search engine.

High Severity

Table 3.2.1.1: Overall statistics for high severity vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1992	1991	1990	1989	1988
# of Vuln	2626	1927	948	625	935	777	434	317	128	145	46	18	17	8	13	12	8	2	2
% of Total	41	40	40	49	48	46	43	35	52	58	61	72	68	62	100	80	73	67	100

Medium Severity

Table 3.2.1.2: Overall statistics for medium severity vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1991	1990	1989
# of Vuln	1186	496	200	157	178	194	120	255	29	35	12	4	5	2	2	2	1
% of Total	19	10	8	12	9	12	12	28	12	14	16	16	20	15	13	18	33

Low severity

Table 3.2.1.3: Overall statistics for low severity vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1991	1990
# of Vuln	2555	2446	1210	494	846	701	464	346	89	72	17	3	3	3	1	1
% of Total	40	50	51	39	43	42	46	38	36	29	23	12	12	23	7	9

3.2.2 Design Error Vulnerability Statistics

Table 3.2.2.0: Design vulnerability statistics

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1992	1991	1990	1989
# of Vuln	839	903	470	337	582	461	171	194	45	47	12	5	11	10	6	9	6	1
% of Total	13	19	20	26	30	28	17	21	18	19	16	20	44	77	46	60	55	33

High Severity

Table 3.2.2.1: Statistics for high severity design error vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1992	1991	1990
# Of Vul	203	240	151	130	236	197	74	70	23	23	7	3	7	6	6	8	4
% of design vul	24	26	32	38	40	43	43	36	51	49	58	60	64	60	100	89	67

Medium Severity

Table 3.2.2.2: Statistics for medium severity design error vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1991	1990	1989
# Of Vul	137	147	54	57	68	77	25	41	7	9	4	1	4	2	1	1	1
% of design vul	16	16	11	17	12	17	15	21	15	19	33	20	36	20	11	17	100

Low severity

Table 3.2.2.3: Statistics for low severity design error vulnerabilities

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1993	1990
# Of Vul	499	516	265	150	278	187	72	83	15	15	1	1	2	1
% of design vul	59	57	56	44	48	40	42	43	33	32	8	20	20	17

3.2.3 Statistics on vulnerabilities affecting Confidentiality

This section provides statistics for design error vulnerabilities that allowed unauthorized disclosure of information. Overlaps between daughter data sets that were obtained based on the mode of vulnerability exploitation are expected. This is because it is possible for a vulnerability to have more than one mode of exploitation. In this case, adding up the partial percentages may results to a value higher than a 100. This line of reasoning holds for the next two sections involving statistics that affected integrity and availability. Also, since the NVD has four severity levels and only three are considered for this research, the partial percentages for severity-based data sets would not add up to 100. This should also hold through for partial percentages for data sets that allowed unauthorized disclosure of information, unauthorized modification of information and disruption of services since design error vulnerabilities that allowed user and administrative accesses were not considered in this research.

Table 3.2.3.0: Overall statistics for vulnerabilities that affected confidentiality

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1995	1994	1993	1991	1990
# Of Vul.	385	295	134	95	202	148	66	63	8	8	1	1	1	2	1	1
% Of design vul.	46	33	28	28	35	32	38	32	18	17	8	20	9	20	11	17

High Severity

Table 3.2.3.0.1: Statistics for high severity vulnerabilities that affected confidentiality

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1994	1991
# Of Vul.	50	26	10	8	35	27	7	7	3	1	1
%	13	9	7	8	17	18	11	11	37	100	100

Table 3.2.3.0.1.0: Statistics for high severity vulnerabilities that affected confidentiality and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1991
# Of Vul.	41	24	7	8	32	24	7	5	2	1
%	82	92	70	100	91	89	100	71	67	100

Table 3.2.3.0.1.1: Statistics for high severity vulnerabilities that affected confidentiality and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1994
# Of Vul.	7	5	4	1	6	6	2	3	1	1
%	14	19	40	12	17	22	28	43	33	100

Table 3.2.3.0.1.1: Statistics for high severity vulnerabilities that affected confidentiality and could be exploited via a target accessing attacker's resource

Year	2006	2005
# Of Vul.	2	1
%	4	3

Medium Severity

Table 3.2.3.0.2: Statistics for medium severity vulnerabilities that affected confidentiality

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996
# Of Vul.	50	41	14	22	22	29	7	12	3	2	1
%	13	14	10	11	15	19	11	19	37	25	100

Table 3.2.3.0.2.0: Statistics for medium severity vulnerabilities that affected confidentiality and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997
# Of Vul.	40	18	6	7	12	12	5	3	1
%	80	43	43	32	54	41	71	25	50

Table 3.2.3.0.2.1: Statistics for medium severity vulnerabilities that affected confidentiality and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996
# Of Vul.	9	22	7	15	10	17	2	10	3	1	1
%	18	54	50	68	45	59	28	83	100	50	100

Table 3.2.3.0.2.2: Statistics for medium severity vulnerabilities that affected confidentiality and could be exploited via a target accessing attacker's resource

Year	2006	2005	2004	2003
# of Vul.	2	1	1	1
%	4	2	7	4

Low severity

Table 3.2.3.0.3: Statistics for low severity vulnerabilities that affected confidentiality

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1995	1993	1990
# of Vul	283	228	110	65	145	92	52	44	2	6	1	2	1
%	73	77	82	68	71	62	79	70	25	75	100	100	100

Table 3.2.3.0.3.0: Statistics for low severity vulnerabilities that affected confidentiality and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997	1995
# Of Vul.	180	135	75	57	105	63	39	26	5	1
%	64	59	68	88	72	68	75	59	83	100

Table 3.2.3.0.3.1: Statistics for low severity vulnerabilities that affected confidentiality and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1993	1990
# Of Vul.	93	95	33	12	41	31	17	18	2	2	2	1
%	33	42	30	18	28	34	33	41	100	33	100	100

Table 3.2.3.0.3.2: Statistics for low severity vulnerabilities that affected confidentiality and could be exploited via a target accessing attacker's resource

Year	2006	2005	2004	2002	2001	2000	1999
# Of Vul.	10	3	4	2	4	6	4
%	3	1	4	1	4	11	9

3.2.4 Statistics on vulnerabilities affecting Integrity

This section provides statistics for design error vulnerabilities that allowed unauthorized modification of information.

Table 3.2.4.0: Overall statistics for vulnerabilities that affected integrity

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1995	1994	1993	1991	1990
# Of Vul.	224	226	94	102	125	71	25	23	4	5	1	1	1	1	1
% of design vul	27	25	20	30	21	15	15	12	9	11	20	9	10	11	17

High Severity

Table 3.2.4.0.1: Statistics for high severity vulnerabilities that affected integrity

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1994	1991
# Of Vul.	45	40	11	35	36	17	5	10	2	2	1	1
%	20	18	12	34	29	24	20	43	50	40	100	100

Table 3.2.4.0.1.0: Statistics for high severity vulnerabilities that affected integrity and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1991
# Of Vul.	35	32	8	33	32	14	2	5	1
%	78	80	73	94	89	82	40	50	100

Table 3.2.4.0.1.1: Statistics for high severity vulnerabilities that affected integrity and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1994
# Of Vul.	11	10	3	5	5	4	3	6	2	2	1
%	24	25	27	14	14	23	60	60	100	100	100

Table 3.2.4.0.1.2: Statistics for high severity vulnerabilities that affected integrity and could be exploited via a target accessing an attacker's resource

Year	2006	2005
# Of Vul.	2	1
%	4	2

Medium Severity

Table 3.2.4.0.2: Statistics for medium severity vulnerabilities that affected integrity

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997	1995	1993	1990
# Of Vul.	53	41	16	30	19	23	10	4	1	1	1	1
%	24	18	17	29	15	32	40	17	20	100	100	100

Table 3.2.4.0.2.0: Statistics for medium severity vulnerabilities that affected integrity and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997	1993
# Of Vul	42	14	8	9	8	13	7	1	1	1
%	79	34	50	30	42	56	70	25	100	100

Table 3.2.4.0.2.1: Statistics for medium severity vulnerabilities that affected integrity and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1995	1993	1990
# Of Vul	9	24	7	21	11	11	3	4	1	1	1
%	17	58	44	70	58	48	30	100	100	100	100

Table 3.2.4.0.2.2: Statistics for medium severity vulnerabilities that affected integrity and could be exploited via a target accessing an attacker's resource

Year	2006	2005	2004	2003	2000
# Of Vul	3	4	1	1	1
%	6	10	6	3	10

Low severity

Table 3.2.4.0.3.0: Statistics for low severity vulnerabilities that affected integrity

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997
# Of Vul	126	145	67	37	70	31	10	9	2	2
%	56	64	71	36	56	44	40	39	50	40

Table 3.2.4.0.3 Statistics for low severity vulnerabilities that affected integrity that could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997
# Of Vul	77	61	37	21	48	15	8	5	2
%	61	42	55	57	68	48	80	55	100

Table 3.2.4.0.3.0 Statistics for low severity vulnerabilities that affected integrity that could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998
# Of Vul	40	65	22	15	20	17	5	3	2
%	32	45	33	40	28	55	50	33	100

Table 3.2.4.0.3.1 Statistics for low severity vulnerabilities that affected integrity that could be exploited via a target accessing an attacker's resource

Year	2006	2005	2004	2003	2002	1999
# Of Vul	17	20	9	1	1	2
%	13	14	13	3	1	2

3.2.5 Statistics on vulnerabilities affecting availability

This section provides statistics of design error vulnerabilities that allowed disruption of service.

Table 3.2.5.0: Overall statistics for vulnerabilities that affected availability

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996	1993	1992
# Of Vul	201	161	96	66	87	81	17	34	14	5	1	1	1
design vul %	24	18	20	19	15	17	10	17	31	11	8	10	17

High Severity

Table 3.2.5.0.1: Statistics for high severity vulnerabilities that affected availability

Year	2006	2005	2004	2003	2002	2001	2000	1998	1992
# Of Vul	51	25	5	11	16	10	3	3	1
%	25	15	5	17	18	12	18	21	100

Table 3.2.5.0.1.0: Statistics for high severity vulnerabilities that affected availability and could be remotely exploited

Year	2006	2005	2004	2003	2002	2001	2000	1998
# Of Vul	44	21	3	8	13	7	2	1
%	86	84	60	72	81	70	67	33

Table 3.2.5.0.1.1: Statistics for high severity vulnerabilities that affected availability and could be locally exploited

Year	2006	2005	2004	2003	2002	2001	2000	1998	1992
# Of Vul.	9	7	2	6	3	4	1	2	1
%	18	28	40	54	19	40	33	67	100

Table 3.2.5.0.1.1: Statistics for high severity vulnerabilities that affected availability and could be exploited via a target accessing an attacker's resource

Year	2006	2005
# Of Vul.	2	1
%	4	4

Medium Severity

Table 3.2.5.0.2: Statistics for medium severity vulnerabilities that affected availability

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997	1993
# Of Vul.	17	31	4	7	10	6	4	3	1	1
%	8	19	4	11	11	7	23	9	20	100

Table 3.2.5.0.2.0: Statistics for medium severity vulnerabilities that affected availability and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1993
# Of Vul.	12	16	2	5	5	5	2	2	1
%	70	52	50	71	50	83	50	67	100

Table 3.2.5.0.2.1: Statistics for medium severity vulnerabilities that affected availability and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1997	1993
# Of Vul.	4	16	2	2	5	2	2	1	1	1
%	23	52	50	28	50	33	50	33	100	100

Table 3.2.5.0.2.1: Statistics for medium severity vulnerabilities that affected availability and could be exploited via a target accessing an attacker's resource

Year	2006	2005
# Of Vul	2	1
%	12	3

Low Severity

Table 3.2.5.0.3: Statistics for low severity vulnerabilities that affected availability

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996
# Of Vul	129	105	87	48	61	65	10	31	11	4	1
%	64	65	91	72	70	80	59	91	78	80	100

Table 3.2.5.0.3.0: Statistics for low severity vulnerabilities that affected availability and could be exploited remotely

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997
# Of Vul	81	59	69	40	52	41	8	22	7	3
%	63	56	79	83	85	63	80	71	64	75

Table 3.2.5.0.3.1: Statistics for low severity vulnerabilities that affected availability and could be exploited locally

Year	2006	2005	2004	2003	2002	2001	2000	1999	1998	1997	1996
# Of Vul	46	44	18	10	10	28	3	11	4	1	1
%	36	42	21	21	16	43	30	35	36	25	100

Table 3.2.5.0.3.2: Statistics for low severity vulnerabilities that affected availability and could be via a target accessing an attacker's resource

Year	2006	2005	2004
# Of Vul	6	3	4
%	5	3	4

4.0 Analysis of Results

In this chapter, the analyses of the data presented in section 3.2 are provided. The chapter is divided into two main sections. The first section provides information about the methods that were used to analyze the data. The second section presents the results and the analyses of the results.

4.1 Data Analysis Method

This section provides the data analysis methods that were used in this research. It is divided into three sub sections: the Cartesian coordinate system, mean and curve fitting.

4.1.1 The Cartesian Co-ordinate System

The data sets presented in section 3.2 were plotted on the two–dimensional Cartesian coordinate system. The percentage values were plotted on the y-axis while the "year of occurrence" of the vulnerabilities was plotted on the x-axis. To produce each of the graphs presented below, MATLAB program codes were written and run.

4.1.2 Mean

The arithmetic mean is the standard average, often simply called the mean. It is the sum of the individual data points of a data set dived by the total number of data points. Mathematically,

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

In this case, since each year from 1988 to 2006 had a data entry, the value of n was set to 19. 19 is the number of years from 1988 to 2006 and hence the number of data points for each data set. In section 3.2, years with zero data points were not included in the data sets. The equation above was implemented using MATLAB built-in function, **mean.**

4.1.3 Curve Fitting

Curve fitting is finding a curve which matches a series of data points and possibly other constraints. In curve fitting or regression analysis an approximate fit rather than exact fit is looked for. Regression analysis estimates relationships between one or more response variables. In this research non linear regression (polynomial fit) was used to estimate the relationship between quantities. This entailed fitting the data points with an nth degree polynomial function. The main task was associated with finding the coefficients of the polynomial function. The polynomial function was then evaluated using the data points and then plotted against the independent variable to estimate the

relationship between the quantities. MATLAB built-in function **polyfit** and **polyval** were used to calculate the polynomial coefficients and to evaluate the polynomials at the data points, respectively.

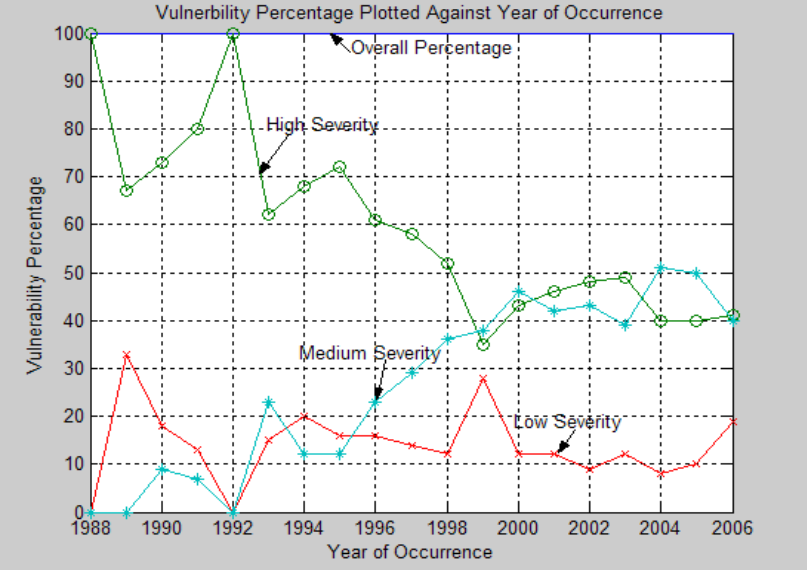
4.2 Analysis of Results

In this section, the results obtained from analyzing the data sets are presented. First the overall vulnerability statistics are presented. This is followed by results for design error vulnerability statistics. Other results include those of design error vulnerbility statistics that allowed unauthorized disclosure of information, unauthorized modification of information and disruption of service.

4.2.1 Overall Vulnerability Statistics

This subsection presents the results for the overall vulnerability statistics. The graph below shows a comparative study for high, medium and low severity vulnerabilities.

Figure 4.2.1.0: Graph of overall vulnerability percentage against year of occurrence.



The percentage of high severity vulnerabilities, although, it fluctuated in the course of the years, generally followed a decreasing trend attaining a minimum percentage value of 35% in 1999. The curve shows that in 2004, 2005 and 2006, about 40% of the vulnerabilities were of high severity. All in all, high severity vulnerabilities varied between 35% and 100% (1988, 1992) in the course of the years.

Unlike high severity vulnerabilities, medium severity vulnerabilities, generally took a rising trend between 1988 and 2006. It attained a maximum percentage value of 51% in 2004. A minimum value of 0% for a particular year either means that none of the

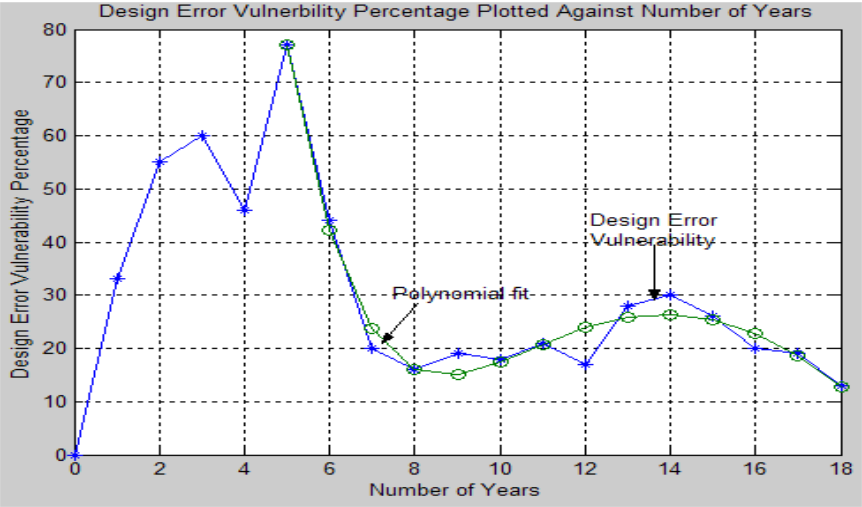
vulnerability data registered in the NVD for that year was of medium severity or that no information about medium severity vulnerability was found in the NVD for that year.

Apart from a few extreme values, low severity vulnerabilities have largely been constant in the course of the years. In 1989, a maximum percentage value of about 33% of vulnerabilities was of low severity.

An average of 59.7% of the vulnerabilities was of high severity followed by low severity vulnerabilities with an average value of 26.3%. Medium severity vulnerabilities averaged 14.1%.

4.2.2 Design Error Vulnerability

Figure 4.2.2.0: Graph of design error vulnerability percentage against "Number of years". The section of the graph from 1993 to 2006 has been fitted with a 5th degree polynomial function.



From figure 4.2.2.0, it can be seen that there was a sharp rise in design error vulnerabilities between 1988 and 1991. In 1991, 60% of the vulnerabilities registered in the NVD were design error vulnerabilities. This value dropped to 46% in 1992 before attaining a maximum of 77% in 1993. In 2006, only 13% of the registered vulnerabilities were design error vulnerabilities. From 1993 to 2006, design error vulnerabilities, generally, followed a decreasing trend. The variation could, approximately, be described by the 5th degree polynomial:

$$f(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + g \qquad (1)$$

Where the constant coefficients a = -0.0014, b = 0.1056, c = -3.0764, d = 43.8942, e = -302.0499 and g = 813.0975. X is the number of years and X ≥ 5. In fact, X = y – y1, where y1 = 5 and y is any value on the X-axis. X = 0 (the origin) corresponds to the year 1988 while X = 18 corresponds to the year 2006. Also, since the values on the y-axis are percentage values,

$$0 \leq |f(x)| \leq 100$$

If there is a value of X = W such that $|f(W)| > 100$, where $5 \leq X < W$, then the polynomial representation of the variation of the design error vulnerabilities breaks down.

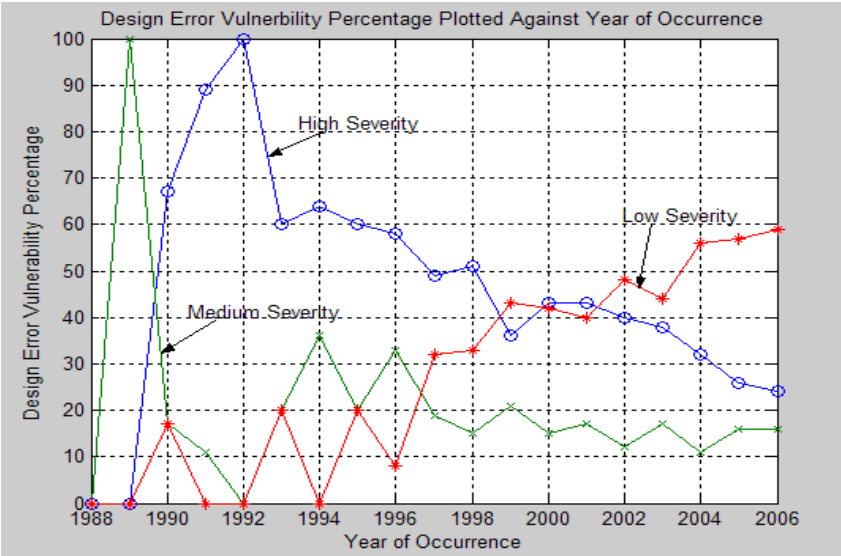
By evaluating equation (1) and within the limit of statistical error, it can be predicted that in 2007 (y = 19, X = y - y1 = 19 - 5 = 14), 49.8% of all vulnerabilities will be design error vulnerabilities. Similarly, in 2008 (y = 20, X = y - y1 = 20 - 5 = 15), 58.6% of all vulnerabilities will be design error vulnerabilities.

On an average, 29.5% or approximately 3 of 10 of the overall vulnerabilities registered in the NVD between January 1988 and December 2006 were design error vulnerabilities. Considering that there are pieces of information about other types of vulnerability registered in this database, this value is significantly high. However, it should be kept in mind that the mean value is very sensitive to extreme values.

4.2.2.1 High, Medium and Low Severity Design Error Vulnerabilities

Figure 4.2.2.1.0 shows that there was a sharp rise is high severity design error vulnerabilities from 1989 to 1992. In 1992, all the design error vulnerabilities that were recorded in the NVD were high severity giving a percentage value of 100%. Between 1992 and 2006, there was a decreasing trend in high severity design error vulnerabilities. In 2006, a minimum percentage value of 24% was attained. An average of 46.3% of the design error vulnerabilities were of high severity.

Figure 4.2.2.1.0: Comparative study of high, medium and low severity design error vulnerabilities



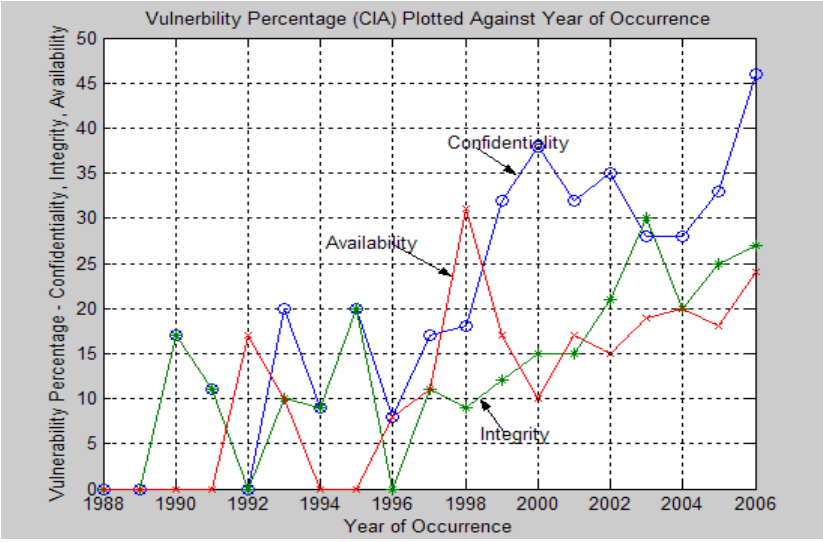
The mean percentage for medium severity component of the design error vulnerability was found to be 20.8%. In 1989, 100% of the registered design error vulnerabilities were of medium severity. There was just one design error vulnerability registered during that year. Between 1993 and 2006, medium severity design error vulnerabilities varied closely about the mean. A maximum of 36% (1994) of these vulnerabilities were of medium severity as oppose to a minimum of 11% in 2004.

Low severity design error vulnerabilities, while randomly fluctuating between some upper and lower values, increased from zero to 59% in 2006. On the average, 27.3% of the design vulnerabilities registered in the NVD between January 1988 and December 2006 were of low severity.
To conclude, most of the design error vulnerabilities were of high severity, followed by low severity design error vulnerabilities. Medium severity design error vulnerabilities were the least.

4.2.3 Confidentiality, Integrity and Availability

This section provides the results of a comparative study of design error vulnerability statistics that affected confidentiality, integrity and availability of information. Although the signals are quite noisy, it can clearly be seen that the overall result is an increase in all three quantities over the years. Design error vulnerabilities that allowed unauthorized disclosure of information (breach of confidentiality) varied between 0% (in 1992) and 20% (in 1993 and 1995). From 1996 to 2000, there was sharp rise in the percentage of these vulnerabilities. In 2000, 38% of design error vulnerabilities allowed a breach of information confidentiality. This percentage value dropped to 28% in 2004 before rising to a maximum of 46% in 2006.

Figure 4.2.3.0: Comparative study of design vulnerabilities that affected confidentiality, integrity and availability of information or information processing systems.



In 1988, 1989, and 1996, there was either no registered design error vulnerability in the NVD that allowed unauthorized modification of information or there was no information in the database that could be associated with design vulnerabilities that allowed unauthorized disclosure of information. In 2003, a maximum value of 30% of the registered design error vulnerabilities could allow unauthorized modification of information.

In 1998, a peak percentage value of 31% of design error vulnerabilities allowed disruption of service. Between 1999 and 2006, the percentage values varied between 17% and 24%, respectively. In 1988, 1989, 1991, 1994 and 1995, either there was no design error vulnerability allowed disruption of service or there was no information in the NVD that could be associated to design error vulnerabilities that allowed disruption of service.

On an average 20.6% of the design error vulnerabilities allowed unauthorized disclosure of information, 13.2% of them allowed unauthorized modification of information and 11.4% of them allowed disruption of service. Based on these averages and the curves above, it is clear that the design error vulnerabilities targeted information confidentiality the most and disruption of service the least. These average percentages do not add up 100 because some of the design error vulnerabilities that allowed administrative and user account access have not been studied in this research.

It should be noted that that decreasing trend of design error vulnerabilities shown in Figure 4.2.2.0 has not been reflected in Figure 4.2.3.0. A possible reason is because the quantities plotted in Figure 4.2.3.0 are not the only components that contributed to the design error vulnerability statistics. For instance, part of the design error vulnerabilities allowed administrative access and user account access. This contribution was left out during this research. Also, the NVD, during this research, reported only those impact types that directly affected a vulnerability. In this light, the impact types that did not directly affect software design error vulnerability were never taken into account and so their contributions are not represented in this study.

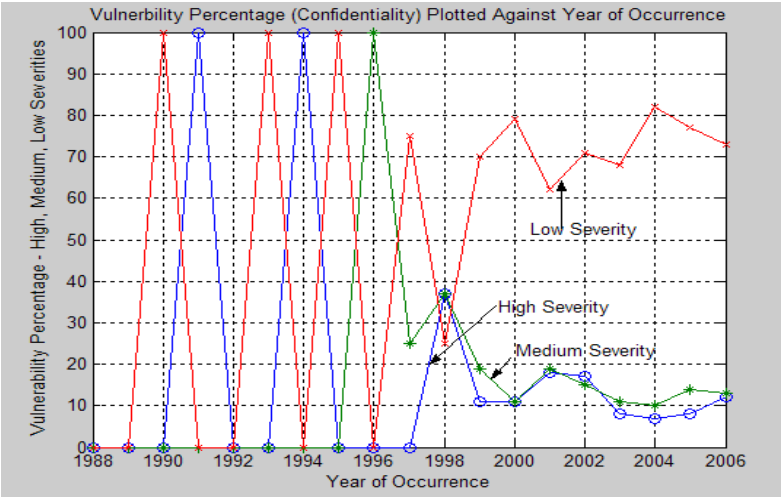
4.2.4 Confidentiality

This section presents the results obtained from the data sets for design error vulnerabilities that allowed unauthorized disclosure of information. First, the results of a comparative study of the severities of these vulnerabilities are presented. Second, for each level of severity, the results of a comparative study of the modes of exploitation of these vulnerabilities are presented.

4.2.4.0 High, Medium and Low Severities

Figure 4.2.4.0.1 shows that variation of high and low severity design error vulnerabilities were basically random between 1988 and 1996 with the variation alternating between zero and 100%. Before 1996, no medium severity design error vulnerabilities were in the NVD or none of the registered design error vulnerabilities were of medium severity.

Figure 4.2.4.0.1: Comparative study of high, medium and low severity design error vulnerabilities that allowed unauthorized disclosure of information.



From 1998 to 2006, there was a decreasing variation in high severity vulnerabilities the variation could be approximated by the 5th degree polynomial (Figure 4.2.4.0.2)

$$f(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + g \qquad (2)$$

Where the constant coefficients a = -0.0554, b = 1.2922, c = -10.8218, d = 38.9088, e = -56.3348, g = 37.1702. X = 0 corresponds to the year 1998 while X = 8 corresponds to the year 2006. For this polynomial to be meaningful in this context,

$$0 \leq |f(x)| \leq 100$$

Equation (2) predicts that in 2007 (x = 9), 0.5% of software design error vulnerabilities that allow unauthorized disclosure of information will be of high severity. This value will rise to 75.1% in 2008 (x = 10). This polynomial does appropriately represent the variation of the vulnerabilities for all values of X such that

$$0 \leq |f(x)| \leq 100 \text{ for } 0 \leq X < Y$$

Where Y is the value of X for which $|f(Y)| > 100$.

The variation in low severity vulnerabilities could be approximated by the 5th degree polynomial (Figure 4.2.4.0.2). Similarly, X = 0 corresponds to the year 1998 while X = 8 corresponds to the year 2006.

$$g(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + f \qquad (3)$$

Where the constants coefficients a = 0.0580, b = -1.4627, c = 13.3749, d = -53.4773, e = 89.1219, f = 24.5781. Low severity design error vulnerabilities increased from 25% in 1998 to maximum of 82 % in 2004. Similarly,

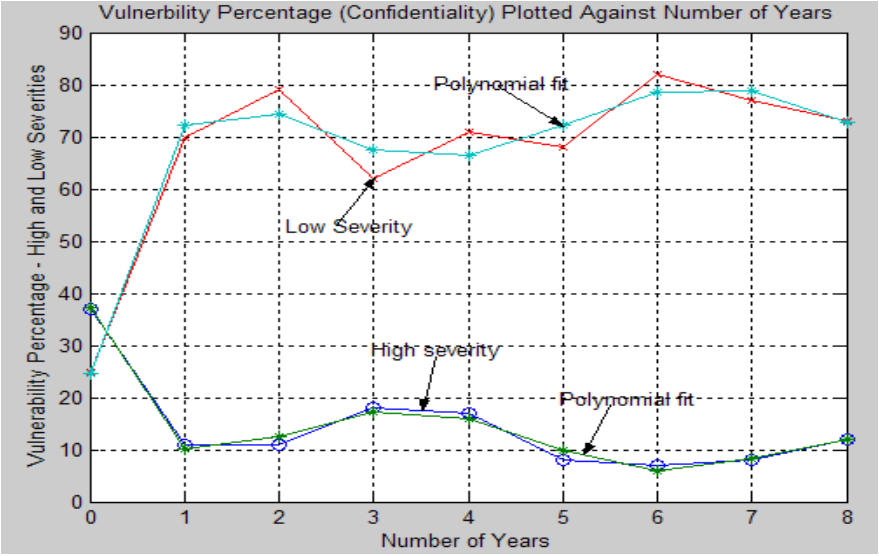
$$0 \leq |g(x)| \leq 100$$

Equation (3) predicts that in 2007 (x = 9), 73.4% of software design error vulnerabilities that allow unauthorized disclosure of information will be of low severity. However, the prediction for 2008 (x = 10) gives a value of 116. The polynomial model, therefore, breaks down after at x = 10 and can no longer be used to describe the variation of the vulnerabilities. An as a result,

$$0 \leq |g(x)| \leq 100 \text{ for } 0 \leq X < P$$

Where P is the value of X for which $|g(P)| > 100$.

Figure 4.2.4.0.2: Polynomial fit for high and low severity design error vulnerabilities that allowed unauthorized disclosure of information.



Like high severity vulnerabilities, medium severity vulnerabilities decreasingly varied from 1998 to 2006. This variation was, approximately, found to be a 7th degree polynomial equation of the form (Figure 4.2.4.0.3)

$$f(x) = px^7 + qx^6 + rx^5 + sx^4 + tx^3 + ux^2 vx + w \qquad (4)$$

Where the constants coefficients, p = 0.0103, q = -0.3045, r = 3.5269, s = -20.0921, t = 57.1345, u = -69.4251, v =10.9709, w = 37.0199. X = 0 corresponds to the year 1998 while X = 8 corresponds to the year 2006.

$$0 \leq |f(x)| \leq 100$$

In 2007 (X = 9), 39.8% of software design error vulnerabilities that allow unauthorized disclosure of information will be of medium severity. However, for X = 10, which corresponds to the year 2008, the value of $|f(10)| = 607.7$. This means that the polynomial model breaks down and hence can not describe the variation of medium

severity software design error vulnerabilities that allowed unauthorized disclosure after 2007. In this respect,

$$\mathbf{0 \leq |f(x)| \leq 100} \text{ for } \mathbf{0 \leq X < K}$$

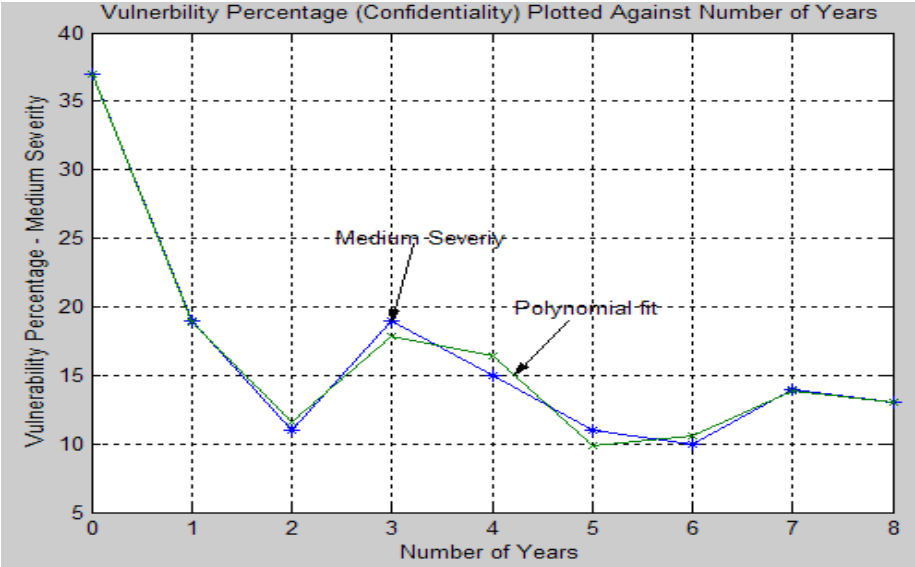
Where K is the value of X for which $|f(K)| > 100$

Medium severity design error vulnerabilities that allowed unauthorized disclosure of information varied between a maximum percentage value of 37% (1998) and a minimum of 10% (2004)

It can be concluded that while high and medium severity designed error vulnerabilities showed a decay pattern from 1998 to 2006 and 1996 to 2006, respectively, low severity design error vulnerabilities rose significantly from 1998 to 2006.

On an average, 17.3% of the design error vulnerabilities that allowed unauthorized disclosure of information were of high severity as oppose to 14.4 % and 51.7% medium and low severity design error vulnerabilities, respectively.

Figure 4.2.4.0.3: Polynomial fit for medium severity design error vulnerabilities that allowed unauthorized disclosure of information.



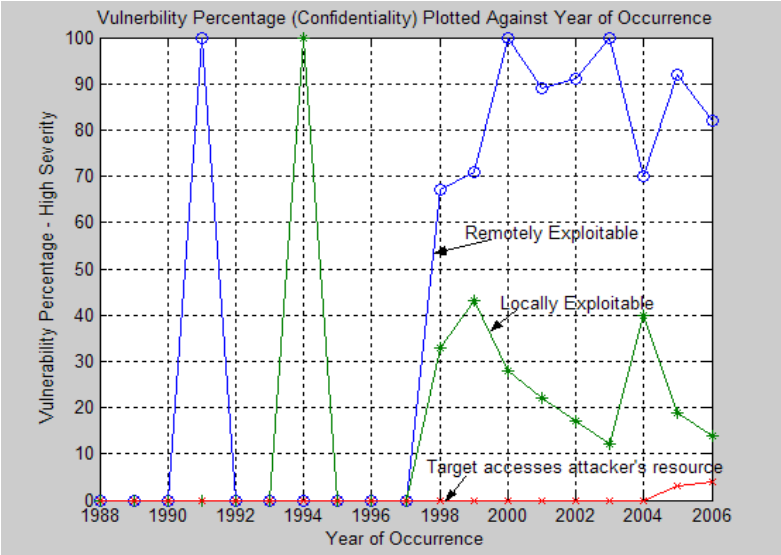
4.2.4.1 Modes of Exploitation for High Severity Vulnerabilities

Figure 4.2.4.1.0 presents the results of a comparative study of the three modes of exploitation in relation to high severity design error vulnerabilities that allowed unauthorized disclosure of information. In 1991 the lone high severity design error vulnerability that was registered in the NVD could only be exploited remotely. Similarly, the lone high severity design error vulnerability registered in 1994 could be exploited locally.

From 1998 to 2006, at least 67% (1998) of high severity design error vulnerability that allowed unauthorized disclosure of information could be exploited remotely. On the other hand, at least 11% (2003) and at most 41% (1999) of medium severity design error vulnerabilities could be exploited locally. Between 1988 and 2004, there was either no high severity design error vulnerability that could be exploited by a target accessing an attacker's resource or there was no information in the NVD about this mode of exploitation for high severity design error vulnerabilities. However, there was a linear increase from zero in 2004 to a maximum of 4% in 2006 of high severity design error vulnerabilities that could be exploited by this means.

All in all, most of the high severity design error vulnerabilities that allowed unauthorized access to information could be exploited remotely. On an average, 45.4% of the high severity design error vulnerabilities that allowed authorized disclosure of information were exploitable remotely, 17.3% of them were exploitable locally while only 0.4% were exploitable by a target accessing an attacker's resource.

Figure 4.2.4.1.0: Comparative study of the modes of exploitation for high severity design error vulnerabilities that allowed unauthorized disclosure of information.



4.2.4.2 Modes of Exploitation for Medium Severity Vulnerabilities

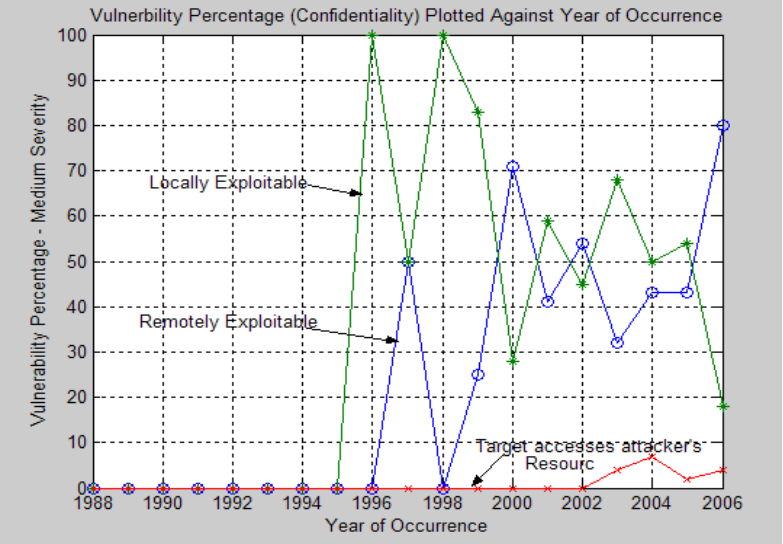
Figure 4.2.4.2.0 it can be seen that, for each year starting from 1996 to 2006, the percentage of medium severity design error vulnerabilities that allowed unauthorized disclosure of information and could be exploited remotely were inversely proportional to the percentage of medium severity design error vulnerabilities that allowed unauthorized disclosure of information and could be exploited locally. For each increase in the percentage of medium severity vulnerabilities that could be exploited remotely, there was, to a fair degree of approximation, a corresponding decrease in percentage of medium severity vulnerabilities that could be exploited locally. In fact, each point on the curve representing remotely exploitable medium severity design error vulnerabilities is a

mirror image of a point on the curve representing locally exploitable design error vulnerabilities.

In 1997, 50% of the medium severity design error vulnerabilities that allowed unauthorized disclosure of information could be exploited both remotely and locally. In 2004, a maximum of 7% of those vulnerabilities could be exploited by a target accessing an attacker's resource. Unlike between 1988 and 1998 where there were many zero percentage values of these vulnerabilities that could be exploited remotely and locally, between 1999 and 2006, vulnerabilities that could be exploited locally varied between 18% in 2006 and 83% in 1999 while those that could be exploited remotely varied between 25% in 1999 and 80% in 2006. Between 1988 and 2002, 0% of these vulnerabilities could be exploited by a target accessing an attacker's resource.

On an average, 23.1% of medium severity design error vulnerabilities that allowed unauthorized disclosure of information could be exploited remotely, 34.5% could be exploited locally and 0.9% could be exploited by a target accessing an attacker's resource.

Figure 4.2.4.2.0: Comparative study of the modes of exploitation for medium severity design error vulnerabilities that allowed unauthorized disclosure of information.



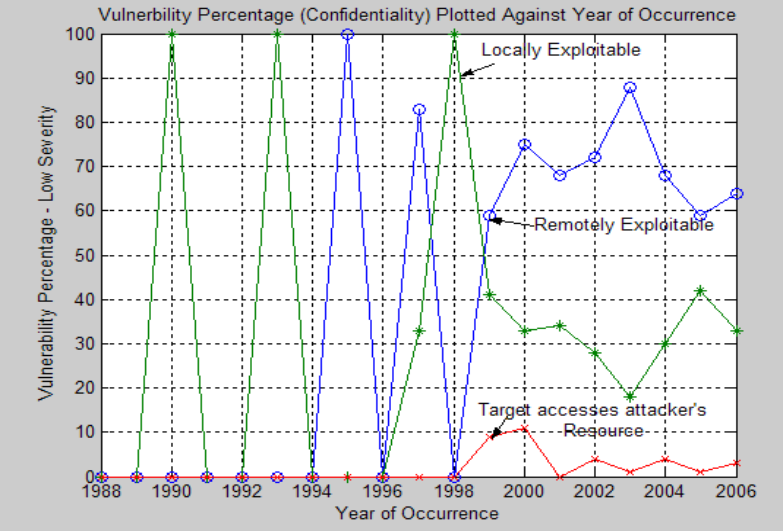
4.2.4.3 Modes of Exploitation for Low Severity Vulnerabilities

Figure 4.2.4.3.0 shows that between 1998 and 2006 the curve representing low severity design error vulnerabilities that allowed unauthorized disclosure of information and could be exploited remotely is a mirror image of the curve representing low severity design error vulnerabilities that allowed unauthorized disclosure of information and could be exploited locally - an inverse proportionality variation. The curve representing remotely exploitable low severity vulnerabilities shows a variation between 59% (1998) and 89% (2006) while that representing locally exploitable low severity vulnerabilities

shows a variation between 41% and 18% within the same period. Before 1999, 0% of the vulnerabilities could be exploited by a target accessing an attacker's resource. However, between 2001 and 2006, low severity design error vulnerabilities that affected information confidentiality and could be exploited by a target accessing an attacker's resource was periodic in nature with a period of one year and amplitude of 3%, approximately.

A period is an interval of time that an event takes place within. It is measured between a start point and an end point and generally repeats or progresses, in a cycle with the end point of one period being the start point of the next. The amplitude is the magnitude of the maximum disturbance (percentage value) during one wave cycle.

Figure 4.2.4.3.0: Comparative study of the modes of exploitation for low severity design error vulnerabilities that allowed unauthorized disclosure of information.



Between 1988 and 2006, 38.7% of the low severity design error vulnerabilities that allowed unauthorized disclosure of information and could be exploited remotely as oppose to 31.2% of those that could be exploited locally and 1.7% of those that could be exploited by a target accessing an attacker's resource.

4.2.5 Integrity

This section presents the results obtained from the data sets for design error vulnerabilities that allowed unauthorized modification of information. First, the results of a comparative study of the severities of these vulnerabilities are presented. Second, for each level of severity, the results of a comparative study of the modes of exploitation of these vulnerabilities are presented.

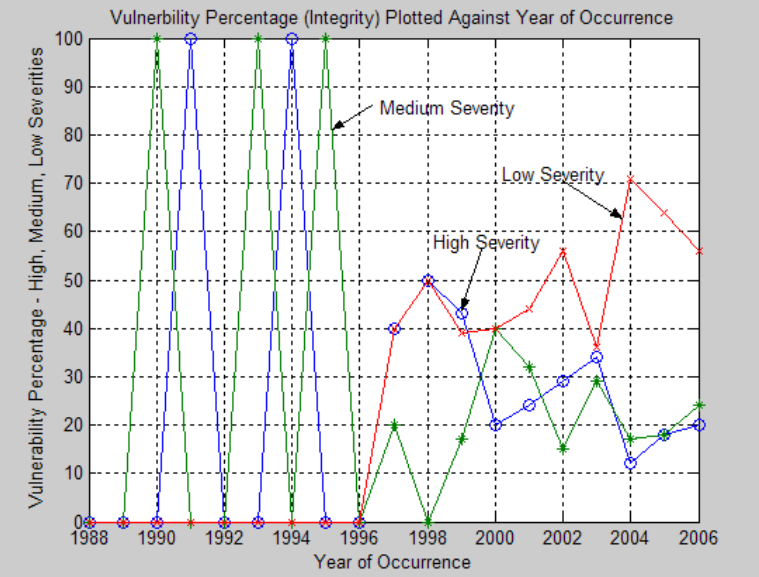
4.2.5.0 High, Medium and Low Severities

Between, 1988 and 1996, high and medium severity design error vulnerabilities that allowed unauthorized modification of information varied in a random manner between

0% and 100%. One occurrence of high severity design error vulnerability that allowed unauthorized modification of information was found in the NVD for 1991 and 1994. A similar result was registered in 1990, 1993 and 1995 for medium severity design error vulnerability.

0% of low severity vulnerability was registered between 1988 and 1996. Between 1997 and 2006, between 50% (1998) and 12% (2004) of the design error vulnerabilities that allowed unauthorized modification of data were of high severity. Within the same period, between 0% (1998) and 40% (2000) of these vulnerabilities were of medium severity and between 36% (2003) and 71% (2004) were of low severity.

Figure 4.2.5.0.1: Comparative study of high, medium and low severity design error vulnerabilities that allowed unauthorized modification of information.



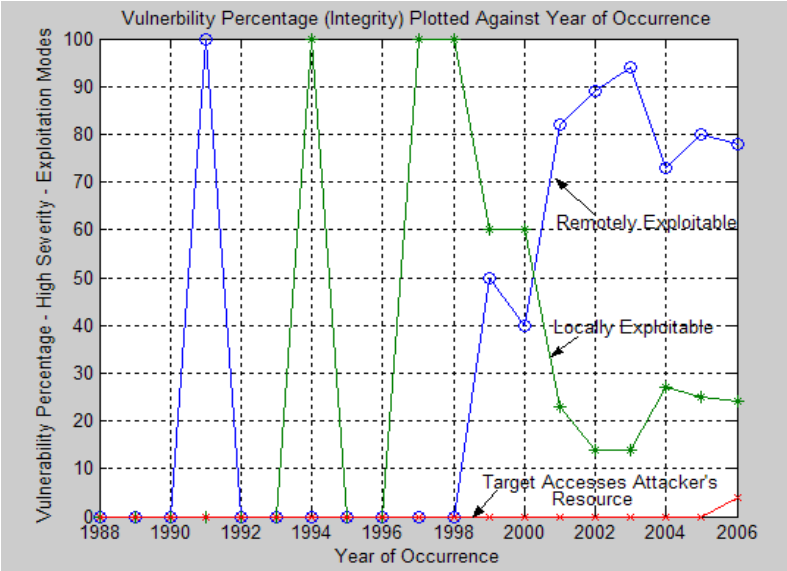
On an average, 25.8%, 26.9% and 26.1% of the design vulnerabilities that allowed unauthorized modification of information were of high, medium and low severities, respectively.

4.2.5.1 Modes of Exploitation for High Severity Vulnerabilities

Between 1998 and 2006 (figure 4.2.5.1.0) there was a rapid rise in the percentage of remotely exploitable high severity design error vulnerabilities that allowed unauthorized modification of information from a minimum of 0% in 1998 to a maximum of 94% in 2003. On the other hand, there was a rapid decrease in those vulnerabilities that were locally exploited from a maximum of 100% in 1998 to 14% in 2003. To a very rough approximation, the increase in remotely exploitable vulnerabilities could be said to be an exponential increase while the decrease in locally exploitable vulnerabilities could be said to an exponential decay in nature. The two curves show an inverse relationship with each other. Vulnerabilities that could be exploited by a target accessing an attacker's resource basically stayed constant at 0% until 2006 that it rose to a maximum of 4%.

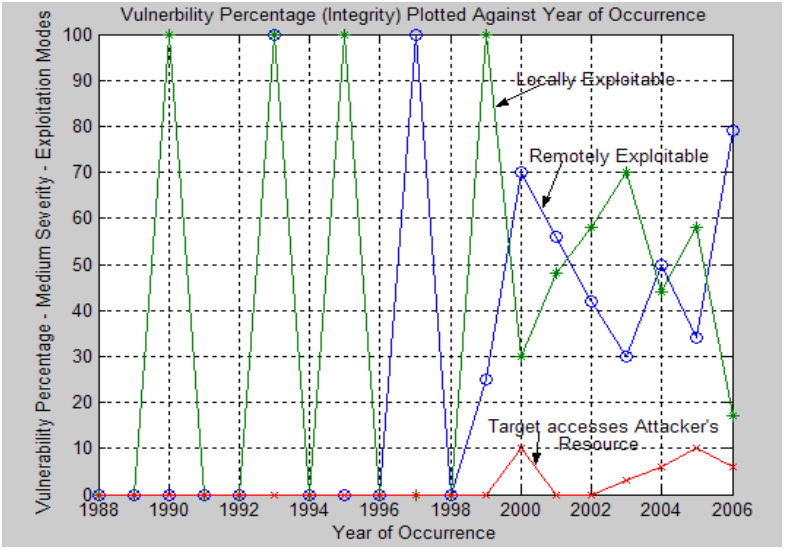
On an average, 36.1%, 28.8% and 0.2% of the high severity design error vulnerabilities that allowed unauthorized modification of information were, respectively, remotely exploitable, locally exploitable and exploitable by a target accessing an attacker's resource.

Figure 4.2.5.1.0: Comparative study of the modes of exploitation for high severity design error vulnerabilities that allowed unauthorized modification of information.



4.2.5.2 Modes of Exploitation for Medium Severity Vulnerabilities

Figure 4.2.5.2.0: Comparative study of the modes of exploitation for medium severity design error vulnerabilities that allowed unauthorized modification of information.



Between 1999 and 2006 (figure 4.2.5.2.0) the variation of medium severity design error vulnerabilities that allowed unauthorized modification of information and could be exploited remotely mirrored the variation of vulnerabilities that could be exploited locally. To a rough approximation, for each percentage increase in the remotely exploitable vulnerabilities, there was corresponding percentage decrease in the locally exploitable vulnerabilities.

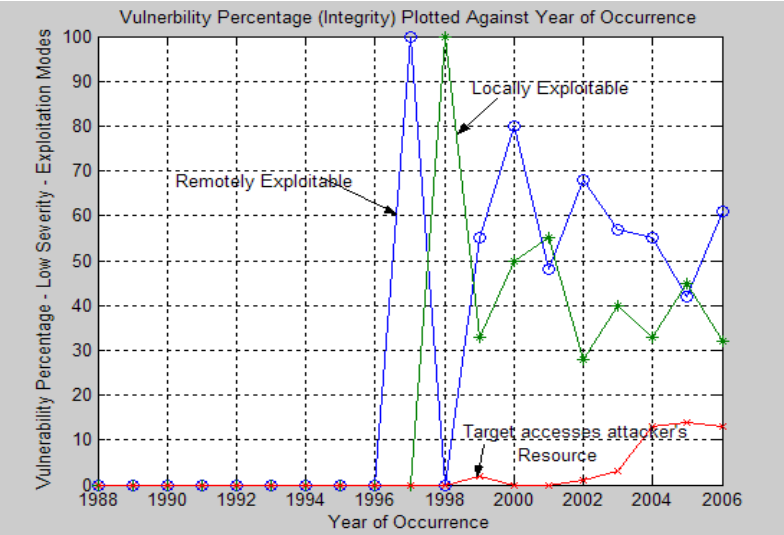
Remotely exploitable vulnerabilities varied between 25% in 1999 and 79% in 2006 while locally exploitable vulnerabilities varied between 100% in 1999 and 17% in 2006. Medium severity design error vulnerabilities that allowed unauthorized modification of information and could be exploited by a target accessing an attacker's resource varied from 0% to a maximum of 10% with a linear increase from 0% in 2002 to 10% in 2005.

On an average, 30.8% of these vulnerabilities could be exploited remotely, 38.2% locally and 1.8% by a target accessing an attacker's resource.

4.2.5.3 Modes of Exploitation for Low Severity Vulnerabilities

Between, 1998 and 2006 low severity design error vulnerabilities that allowed unauthorized modification of information and could be exploited remotely varied between 0% (1998) and 80% (2000) while locally exploitable design error vulnerabilities varied between 100% (1998) and 19% (2002). Generally, remotely exploitable vulnerabilities increased within this period.

Figure 4.2.5.3.0: Comparative study of the modes of exploitation for low severity design error vulnerabilities that allowed unauthorized modification of information



Locally exploitable vulnerabilities, on the other hand dropped. In 1997 all the low severity vulnerabilities were remotely exploitable. Similarly, in 1998, all the low severity vulnerabilities were locally exploitable. Prior to 1996, 0% of low severity design error vulnerabilities that allowed unauthorized modification of information could either be

exploited locally or remotely. For vulnerabilities that require a target to access an attacker's resource before they could be exploited, there was a rise from 0% to a maximum of 14% in 2005.

On an average, 29.8% of low severity design error vulnerabilities that allowed unauthorized modification of information were exploitable remotely, 21.9% of them were exploitable locally and 2.4% could only be exploited if a target did access an attacker's resource.

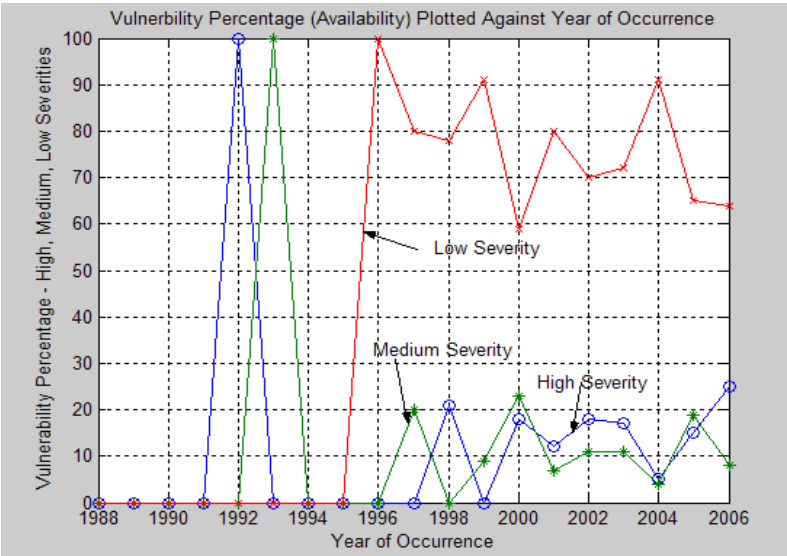
4.2.6 Availability

This section presents the results obtained from the data sets for design error vulnerabilities that allowed disruption of service. First, a comparative result of the severities of these vulnerabilities is presented. Second, for each level of severity, a comparative result for the mode of exploitation of these vulnerabilities is presented.

4.2.6.0 High, Medium and Low Severities

From 1996 to 2005, high and medium severity design error vulnerabilities that allowed disruption service, varied in a similar fashion. In fact, the curve representing high severity design vulnerabilities is a medium severity design error vulnerability curve shifted forward by one year but with some minor differences in their percentage values. Between 0% and 24% of the design error vulnerabilities that allowed disruption of service were either high or medium severity between 1996 and 2006. In 1992 and 1993, 100% of these vulnerabilities were of high and medium severities, respectively.

Figure 4.2.6.0.1: Comparative study of high, medium and low severity design error vulnerabilities that allowed disruption of service.

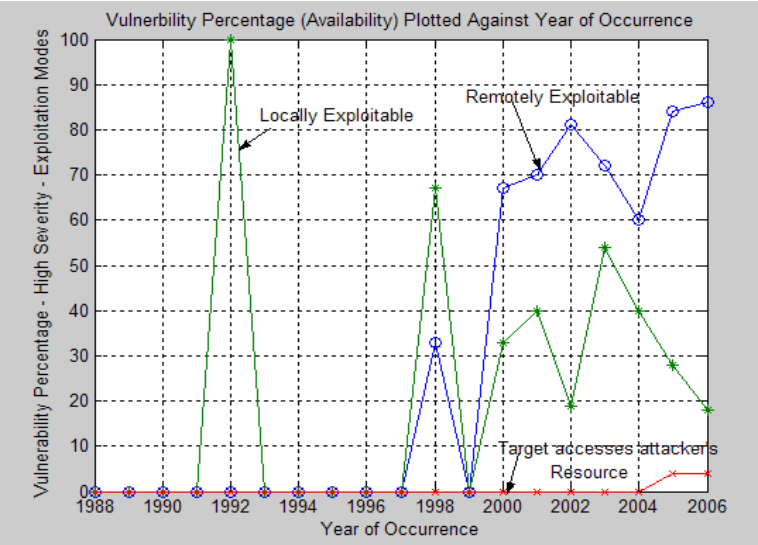


On the other hand, low severity design error vulnerabilities that allowed disruption of service varied between 100% (1996) and 64% (2006) and 0% otherwise.

On an average, 12.2% of the vulnerabilities were of high severity, 11.2% were of medium severity and 44.7% were of low severity.

4.2.6.1 Modes of Exploitation for High Severity Vulnerabilities

Figure 4.2.6.1.0: Comparative study of the modes of exploitation for high severity design error vulnerabilities that allowed disruption of service.



The variation of high severity design error vulnerabilities that allowed disruption of service and were remotely or locally exploitable was random. A majority of these vulnerabilities were remotely exploitable. Between 2000 and 2006, at least 60% (2004) and at most 86% (2006) of the vulnerabilities were remotely exploitable. Within the same period, at least 18% (2006) and at most 54% (2003) were locally exploitable. In 1992, 100% of the vulnerabilities were locally exploitable.

On an average, 29.1% of the high severity design error vulnerabilities that allowed disruption of service were exploitable remotely, 21% were exploited locally and 0.4% by a target accessing an attacker's resource.

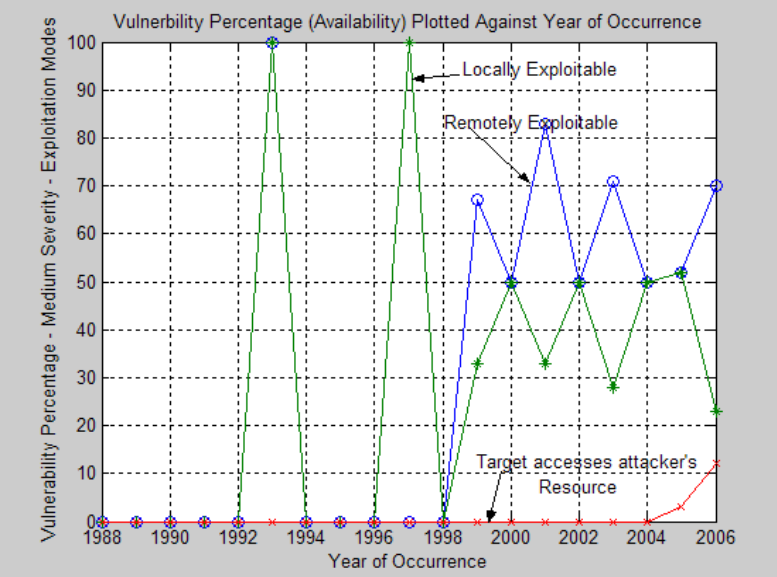
4.2.6.2 Modes of Exploitation for Medium Severity Vulnerabilities

As seen on figure 4.2.6.2.0, in 2000, 2002 and 2004, 50% of medium severity vulnerabilities that allowed disruption of service were remotely and locally exploitable as opposed to 52% in 2005.

In 1999, 2001, 2003 and 2006, a rise in remotely exploitable vulnerabilities was accompanied by a drop in locally exploitable vulnerabilities. Between 1999 and 2006,

while between 50% and 82% (2001) of the vulnerabilities were remotely exploitable, between 23% (2006) and 52% of the vulnerabilities were locally exploitable.
In 1993 and 1997, 100% of the vulnerabilities were locally exploitable.

0% of the vulnerabilities were remotely exploitable between 1988 and 1998. Also, 0% of them were exploited by a target accessing an attacker's resource between 1988 and 2004.
Figure 4.2.6.2.0: Comparative study of the modes of exploitation for medium severity design error vulnerabilities that allowed disruption of service.



On an average, 31.2% of the medium severity design error vulnerabilities that allowed disruption of service were exploitable remotely, 27.3% were exploited locally and 0.8% by a target accessing an attacker's resource.

4.2.6.3 Modes of Exploitation for Low Severity Vulnerabilities

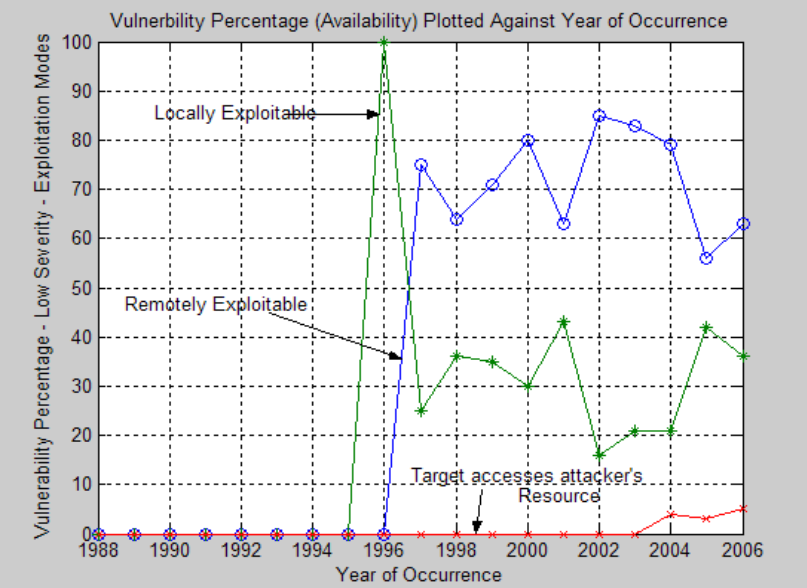
From 1996 to 2006 Low severity design error vulnerabilities that allowed disruption of service and were remotely exploitable mirrored those vulnerabilities that were exploited locally. An increase in those vulnerabilities that were remotely exploitable was accompanied by a decrease in locally exploitable vulnerabilities.

After 1996, between 56% (2005) and 85% of low severity design error vulnerabilities that allowed disruption of service were remotely exploitable while between 15% (2005) and 42% (2001) were locally exploitable. In 1996, 100% of the vulnerabilities were locally exploitable.

From 1988 to 1996, 0% of the vulnerabilities were remotely exploitable and from 1988 to 1995, 0% of the vulnerabilities were locally exploitable. From 1988 to 2003, 0% of the low severity vulnerabilities that allowed disruption of service were exploitable by a target accessing an attacker's resource. The percentage rose to 5% in 2006.

On an average, 37.8% of the low severity vulnerabilities that allowed disruption of service were remotely exploitable, 21.3% of them were locally exploitable and 0.6% was exploitable by a target accessing an attacker's resource.

Figure 4.2.6.3.0: Comparative study of the modes of exploitation for low severity design error vulnerabilities that allowed disruption of service.



5.0 Interpretation of Statistical Results

This chapter provides possible interpretations of the results presented in the previous chapter. The chapter is divided into three sections. In the Section 5.1, attempts are made to identify some of the conditions or factors that affected the discovery of software vulnerabilities especially during the late 1980s or the early 1990s. During these periods, very few vulnerability data were recorded in the NVD. There are two sub-sections in this section. Section 5.1.1 attempts to provide an explanation for the few design error vulnerability disclosure logs that existed during the late 1980s and the early 1990s. Section 5.1.2 attempts to explain why design error vulnerability that allowed unauthorized disclosure of information showed and increase. This explanation is provided from a vulnerability discovery point of view by considering the shift in the motives of attackers.

In Section 5.2, potential facts that provide possible explanations to the variation of design error vulnerabilities between 1988 and 2006 are presented. This section is divided into severn sub-sections.

Section 5.2.1 attempts to interpret the results in relation to the financial loss experienced by software vendors when a vulnerability is announced in their products. Attempts are made in Section 5.2.2 to interpret the results in relation to the dissatisfaction of customers due to an increasing number of reported vulnerabilities in software products.

Section 5.2.3 attempts to interpret the results in relation to the growth and the growing influence of the Internet between 1988 and 2006. In Section 5.2.4 an attempt is made to interpret the results in relation to changes in the number of users per computer and changes in the operating system during the period from 1988 to 2006.

While Section 5.2.5 looks at the interpretation from the legislative or regulatory obligation stance point, Section 5.2.6 looks at the interpretation from the view point of the adoption of secure coding standard by software vendors. In section 5.2.8, attempts are made to interpret the results in relation to changes in the security industry.

Finally, this chapter closes by identifying some of the limitations of using the NVD. These limitations are presented in Section 5.3.

5.1 Design Error Vulnerability Discovery

This section addresses some of the issues associated with vulnerability disclosure log data. It provides potentials reasons why, for instance, in the late 1980s and the early 1990s, very few vulnerability data, in general, and design error vulnerability data, in particular, were recorded in NVD.

5.1.1 Few Design Error Vulnerability Disclosure Log Data

In Figures 4.2.4.0.1, 4.2.4.1.0, 4.2.4.2.0, 4.2.4.3.0 etc, it could be seen that that between 1988 and 1996, the percentages of these vulnerabilities were generally very high, often attaining 100%. A very limited number of vulnerabilities were recorded during these years as they were very few information security firms or organizations that kept track of vulnerability disclosures. There was no universally accepted information security standard at the time. Although the International Organization for Standardization was in existence since 1946 [28], it only got involved in information security in 1999 and in 2000, ISO 17799 (information security code of best practice) was published [29].

In the late 1990 going forward, many security firms probed up that brought about security awareness. Firms like Deloitte, KPMG, Ernst & Young started providing information security services like auditing, risk assessment, etc to businesses.

Every year, since 2001, the SysAdmin, Audit, Networking, and Security (SANS) institute releases a list of SANS top 20 most critical Internet security vulnerabilities [31]. Also, to educate developers, designers, architects and organizations about the consequences of the most common web application security vulnerabilities the Open Web Application Security Project (OWASP) releases, every year, OWASP Top 10 list of the most serious web application vulnerabilities and provides basic methods to protect against these vulnerabilities [30].

5.1.2 Changes in the Motives of Attackers

Figure 4.2.3.0 shows that most of the design error vulnerabilities targeted information confidentiality, followed by information integrity and then information availability. This trend could potentially be attributed to the fact that attacker motives, [16] has shifted to monetary gain instead of to prove skills and satisfy curiosity. Confidential data that could potentially lead to financial gain is the most sort after by attackers. Thus, more efforts seem to have been concentrated or invested in searching and finding design error vulnerabilities that allowed unauthorized disclosure of information than those that allowed unauthorized modification of information or allowed disruption of service.

Unauthorized disclosure of information is usually in the form of identity theft using various methods such as Phishing (see section 2.2) and spyware. By Phishing, a bogus e-mail message directs a recipient to an imitation Web site that mimics the appearance of a bank or Internet merchant Web site. The consumer is asked to update or confirm his or her personal information, and in doing so unwittingly disclose sensitive and confidential information such as social security, credit card or bank account numbers. According to report from Symantec in 2005, Phishing attempts were up 100 percent in the past six months prior to September 2005 [16].

On the other hand, spyware, in its most extreme form, steals social security numbers, passwords and other information stored on the hard drives of computers, either by searching for it or by recording the keystrokes of computer users. Spyware typically reaches a computer by being inadvertently downloaded from a Web site or by being

secretly bundled with free downloadable Internet software used for listening to music or watching movies on a computer [16].

According to [16], the threat from information-stealing software exceeds the threat posed by computer viruses and worms that cause disruption of service and spread themselves via e-mail. Between January 1, 2005 and June 30, 2005, malicious software code that revealed confidential information represented 74 percent of the top 50 malicious code samples reported to software security firm Symantec, up from 54 percent in the previous six-month period [16].

Traditional attack activity is usually motivated by curiosity, a desire to show off technical virtuosity, power, self-expression and peer recognition. This kind of attack which, in many cases, is likely to lead to unauthorized modification of data or disruption of service is gradually being paid less attention by attackers as their motive has shifted towards profit making.

5.2 Interpretation of Results

This section attempts to provide possible interpretation of the results obtained in the previous Chapter from the existence point of view of design error vulnerabilities between 1988 and 2006.

5.2.1 Financial Impact to Software Vendors

According to [4], many believe that software vendors follow the policy of “sell today and fix it tomorrow” or “I would rather have it wrong than late” for launching software product in the market. This policy is dictated by the need to launch products quickly before competitors. It seemed to work in the past because software errors which escape detection during pre-launch testing appear very infrequently in normal operations.

With the present Internet age, this theory will definitely not hold since hackers, researchers, independent security bodies look for flaws in vendors’ software products drastically, increasing the chances of exposing vulnerabilities. From the declining trend presented in figures 4.2.2.0 (from 1993) and 4.2.2.1.0 (from 1992), it appears one could, to a fair approximation, say that software vendors are drifting away from this policy and adopting a better policy that incorporate security into their products. Based on these results, perhaps, one may rephrase the above policy as “fix today and sell tomorrow” or “I would rather have it right and late”

One of the potential reasons why software vendors would incorporate good security features into their software products could likely be the impact of software vulnerability disclosure on their products. When a vulnerability from a software product is disclosed, there is negative and significant change in market value for the software vendor [4]. An average 0.6% value in stock price is lost by a vendor when a vulnerability is reported. This translats to an equivalent loss in market capitalization values of $0.86 billion per vulnerability announcement. Also, if an announced vulnerability is of high severity, the loss in market value of a software vendor is greater. The loss in market value is even

higher if the vulnerability is discovered by rivals or third party security firm rather than the software vendor itself.

According to Wall Street Journal report in Feb 2004 software vendors are spending time and effort in discovering flaws in their rivals' software products in order to influence the rivals' stock prices [4]. In this respect, software vendors are pressured to head towards the development of secure software. In addition, it costs vendors time and money to produce patches to fix vulnerabilities. For instance, the Wall Street Journal (11/09/2004) reported that Microsoft's Internet Explorer (IE) loses market share in the web browser market to competitors like Mozilla's Firefox, due to numerous flaws discovered in IE.

5.2.2 The Influence of Unsatisfied Security Experts

One could also argue that software consumers, in general, and information security experts, in particular, demand for more secure products now than before. This demand then puts pressure on software vendors to incorporate security into their software products.

For instance, in April 19, 2004 (Computerworld), Michael Kamens, global security director at Thermo Electron Corp said: "*We are extremely concerned by the high amount of vulnerabilities and patches from Microsoft. This goes against the credibility of what they have been saying*," [15].

"*Despite the fact that Microsoft has made this huge commitment to security, they are not saying why these vulnerabilities are showing up and what exactly they are doing to research and patch them*," Kesner said [15].

"*There are more questions asked than answered when such a large update is released*," said Robert Bagamery, a systems support specialist at a large Canadian utility company. "*I wonder what they've broken with the fix*," Bagamery said. "*I wonder how many more there are, and how many they know about but aren't talking [about]*" [15].

Security concerns like these, although Microsoft may not response to them, make them think "security" when it comes to developing software. For instance, Microsoft VISTA released in November 2006 was developed with better and enhanced security features than previous versions of Windows.

Before the advent of VISTA, user privileges in Windows XP, for instance, were controlled through group membership, including 'users' and 'administrators' with additional granularity achieved through local or Group Policy settings. The difficulty in controlling certain functions such as software installation led to many corporate IT departments to grant local Administrator to users. [23]

In VISTA, the User Account Control feature allows almost all users to have administrator privileges removed. Users logged in with administrator accounts operate as standard users until higher privileges are required. Standard users are prompted for the password of an administrator whenever they attempt a task requiring that privilege.

Administrators on the other hand are also prompted when they or an application wishes to perform a privileged operation. VISTA also has Windows Defender malware protection and enhanced Windows Firewall among other security features that were not found in previous versions of Microsoft operating systems [23].

In contrast, the large amount of updates release by Microsoft could be seen as a good security practice. In this respect, Microsoft recognizes the security problems and is committed in fixing them. When an antivirus vendor has many updates this is seen as a good security practice. In terms of ISO 15408 this would be seen as good security product support over time. Microsoft updates could, similarly, be considered good security product support over time.

5.2.3 The Influence of the Internet

From figures 4.2.4.1.0, 4.2.4.3.0, 4.2.5.1.0, 4.2.5.3.0, 4.2.6.1.0 and 4.2.6.3.0, it can be seen that high and low severity design error vulnerabilities that allowed unauthorized disclosure of information, unauthorized modification of information and disruption of service were predominantly exploited remotely. Generally, from 1998 – 2006, the figures above show that there was a significant rise in the percentage of design error vulnerabilities that were exploited remotely.

This rise could be explained by the evolution of web access which started approximately in the mid 1990s. For instance, in 1998, Microsoft released Windows 98 with integrated web technology (Internet Explorer 4.0) that facilitated Internet access. Also the number of Internet users in the world rose significantly from 1990. It rose from 2.6 million in 1990 to 385 million in 2006 [19]. In 1998, 145 million people had access to the Internet world wide.

In addition, the year 1998 which marked the rise of remote mode of exploitation for design error vulnerabilities fall within the dot-com bubble era [20]. The dot-com bubble was a speculative bubble covering roughly 1995–2001 during which stock markets in Western nations saw their value increase rapidly from growth in the new Internet sector and related fields.

Furthermore, in the mid 1990s there was a significant growth in computing power. Due to competition among manufacturers, the prices of computing equipment began to drop thus, making them readily affordable to a large number of users. Combined with the fall in the cost of the Internet, attackers were exposed to various types of readily available exploits, many of which were free. In some cases, exploits for some design error vulnerabilities were available at the Internet before vendors released patches to fix them.

5.2.4 Changes to Computer Usage and Operating System

The percentage of locally exploitable design error vulnerabilities, on the other hand, generally, dropped from1998 to 2006. Before the 1990s, the number of users per computer was relatively high. From the mid 1990s onwards the number of users per computer dropped reducing the number of possible events of local exploitation of

vulnerabilities. Many universities, for instance, deployed large number of computers for research purposes. The availability of open source operation system such as UNIX facilitated this mass computer deployment in research institutions.

From an operating system perspective, there has been a significant improvement in access control methods. For instance, in Microsoft Windows 95 and 98 released in 1995 and 1998, respectively, there were serious security issues associated with access control methods that CERT (Computer Emergency Response Team) issued a warning in April 17, 2000. CERT Coordination Center warned that MS Windows 95/98 operating systems were not designed to be used with computers storing data that was considered critical to a project or that had to be very securely protected" [21].

In Windows 2000, access control methods were improved to include administrator, power user and user. Windows 2003 was made to be installed in a lock down mode. Windows 2000, 2003 and XP all have better authentication mechanism and enforceable password policies.

From an organizational view point, for an external attacker to locally exploit a vulnerability, he or she must first pass the physical security barriers before gaining access to the organization's network environment. Administrative controls such as oath of confidentiality help to deter internal users from locally exploiting design error vulnerabilities.

The mode of exploitation for which a target must access an attacker's resource showed an insignificant variation generally, staying at 0% before 1998. Social engineering such as Phishing plays a significant role in this mode of exploitation. Phishing is prominent in web applications. Just after 1998, as shown on Figures 4.2.5.2.0 and 4.2.5.3.0, to use of this mode of exploitation on design error vulnerabilities rose slightly. This was immediately after the release of Windows 98 with integrated web browser that facilitated the use of web applications.

5.2.5 Legislative or Regulatory Obligations

Another reason that might have contributed to the downward trend of design error vulnerabilities, in general, and high severity design error vulnerabilities, in particular, between the mid 1990 and 2006 is the fact that business organizations became obligated to comply with government passed legislations. Some of these legislations were put in place to ensure accountability while others were to ensure the protection of personal or health information.

In Canada, the Alberta Health Sector is governed by the Health Information Act (HIA). This act was passed in 2000 and one of its key components is that a custodian is responsible for protecting all health information under his or her control. A custodian as stated in the Act is a health services provider who is paid under the Alberta Health Care Insurance Plan to provide health services.

In 2002, the Sarbane-Oxley Act was passed in the United States in response to a number of major corporate and accounting scandals including those that affected

Enron. According to the Act, management is responsible for protecting the data with appropriate IT resources, procedures and practices and ensuring that outsourced data is protected.

A Violation of legislation could lead to heavy financial penalty in addition to reputable damage and loss of sensitive information. In order to avoid these impacts, organizations are obliged to select and implement secure software solutions, among other solutions, to protect corporate data. This then calls for the developing of more secure software by software vendors. As an example, the HIA states that access to health information must be on a need-to-know basis. This, thus, calls for stronger authentication features to be incorporated into clinical systems. Also, software systems that will transmit or share information across untrusted network would need to have strong encryption capabilities incorporated into it to ensure compliance with privacy legislation.

5.2.6 Adoption of Secure Coding Standard

Considering the negative financial impact to software vendors when vulnerabilities are reported in their products, the shift in motives of attackers towards financial benefits, software consumer complaints due to ever increasing number of design error vulnerabilities in software products, legislative or regulatory obligations, the presence of widely available and sophisticated attack tools over the Internet, etc, organizations have increasingly been making more efforts in developing, maintaining and implementing secure coding standards for developing software products. The gradual adoption of this standard by software vendors may have contributed to the development of more secure software and hence may have contributed the drop experienced by high severity design error vulnerabilities between the mid 1990s and 2006 (Figure 4.2.2.0).

In addition, there have been significant advancements in software engineering and software design (new software development tools, code security auditing and testing, new ways to identify and prevent software vulnerabilities, advances in threat modeling, etc) that may have contributed to the drop in design error vulnerabilities.

Five years ago (2002), Microsoft committed to the Trustworthy Computing initiative in response to the large number of malware attacks that targeted security weaknesses of Windows XP. Service Pack 2 for Windows XP that came with a host-based firewall was developed under this banner.

Trustworthy Computing is Microsoft's aim to deliver secure, private, and reliable computing experiences for everyone based on sound business practices [26]. The Committee on Information Systems Trustworthiness' publication, Trust in Cyberspace, [25] defines such a system as one which does what users expect it to do and not something else despite environmental disruption, human user and operator errors, and attacks by hostile parties. Design and implementation errors must be avoided, eliminated or somehow tolerated. It is not sufficient to address only some of these dimensions, nor is it sufficient simply to assemble components that are themselves trustworthy. Trustworthiness is holistic and multidimensional.

From Microsoft Trustworthy Computing initiative, Microsoft Security Development Lifecycle (SDL) was born. The SDL is the methodology that was used to ensure that

VISTA was developed with secure code. Windows VISTA is the first desktop operating system that was written completely under the SDL.

The Trustworthy Computing Security Development Lifecycle (SDL) is a process that Microsoft has adopted for the development of software that needs to withstand malicious attacks. The process encompasses the addition of a series of security-focused activities and deliverables to each of the phases of Microsoft's software development process [27].

Windows XP is the most targeted operating system by malware attacks since it is most widely used. Many malware attacks relied on design error vulnerabilities in the core Windows systems such as Internet Explorer (IE) running as part of the operating system with elevated privileges. In VISTA, many of the hooks used by malware to install into the system are now being blocked or monitored. For instance, while changes to startup program require user confirmation, IE runs at a reduced privilege.

5.2.7 Changes in the Security Industry

Looking at the overall vulnerability picture, figure 4.2.1.0, the percentage of high severity vulnerabilities, generally, declined especially between 1995 and 2006. Figure 4.2.2.0 shows that the percentage of design error vulnerabilities also dropped from 1993 towards 2006. There was a decline in higher severity design error vulnerabilities from 1992 to 2006 as shown on figure 4.2.2.1.0.

The decrease is these vulnerabilities could be partly attributed to the fact that the scope of security has been gradually shifting from technical to organizational. Information security is now being incorporated into business drivers. With respect to software development, security is incorporated into software development processes. In this case, funding to security is now looked upon as an investment rather than an expense. For instance, Microsoft Trustworthy Computing initiative described above.

5.3 Limitations of NVD

It is worth pointing out that the use of the NVD as a source of data presented a number of limitations or shortcomings. These include:

- The NVD only records impact types that a vulnerability directly allows. Consequently, impact types that a vulnerability indirectly allows is out of the picture.

- Information needed to create CVSS scores may not be unavailable. For instance, a vendor announces a vulnerability but declines to provide certain details. In such situations, the NVD analysts assign CVSS scores using the information that is available. This lack of adequate information may affect the score allocated to a vulnerability and hence its impact.

- At the time that data this research was conducted, vulnerabilities in the NVD were scored using CVSS version 1. Version 2 of CVSS was released in June 2007 [32].

CVSS version 1 metrics did not contain the granularity of CVSS v2. While these scores are approximations, there are expected to be reasonably accurate CVSS v2 scores.

- If a NVD value is zero for a particular year for a particular vulnerability type, there is no way to determine whether the zero value means that there was no record about the vulnerability in the database for that year or that there was no occurrence of the vulnerability for that year.

- There were very few records of vulnerabilities data during the late 1980s and early 1990s. This may partly explain the noise (very random variations) associated with the data sets as shown by the various plots. This, somehow, affected the interpretation of the results.

- Since there were limited number of vulnerabilities registered during the late 1980s and early 1990s and since arithmetic mean is sensitive to extreme values, the calculated mean values may not reflect the correct picture.

- The period from 1988 to 2006 for which the research was based may be too short for the research results to be generalized.

6.0 Conclusion

This chapter closes the curtain for this research. Despite the limitations outlined in Section 5.9, the research results led to a number of conclusions.

On the overall, most of the vulnerabilities registered in the NVD between 1988 and 2006 were high severity vulnerabilities. High severity vulnerabilities statistics in this database averaged 59.7% - more than half of the NVD data. This indicates that, despite the decreasing trend followed by the overall high severity vulnerabilities, the high average value of these vulnerabilities is a serious cause of security concerns. This suggests that applying fixes for these vulnerabilities should accordingly be prioritized. Overall low severity vulnerabilities did variably increase during the period from 1988 to 2006.

With an average of 29.5%, design error vulnerabilities data occupied, approximately, one third of the content of the NVD. Taking into consideration that the database holds information about a good number of other vulnerabilities, the high average percentage of design error vulnerabilities is potential indication that these vulnerabilities are the primary means through which information assets could be exposed to threat agents. Within the limit of statistical errors, the results obtained by evaluating equation (1) at X = 14 and X = 15 predict that 49.8% and 58.6% of the vulnerabilities in 2007 and 2008, respectively, will be design error vulnerabilities. Although an average, 46.3% of the design error vulnerabilities were of high severity, high severity design error vulnerability followed a decreasing trend.

Design error vulnerabilities that targeted information confidentiality, integrity and availability, although fluctuated in the cause of the years, generally exhibited an increasing trend (Figure 4.2.3.0). Low severity design error vulnerabilities that affected information confidentiality, integrity and availability, generally, increased from 1997 to 2006. On the other hand, high and medium severity design error vulnerabilities that allowed unauthorized disclosure of information, unauthorized modification of information and disruption of service, generally, declined since the late 1990.

Most of the design error vulnerabilities targeted information confidentiality. A probable reason being that vulnerability exploitation has shifted more towards financial profit rather than proving skills and satisfying curiosity. By evaluating equation (2) at X = 9 (2007)and X = 10 (2008), it could be predicted that in 2007 and 2008, 0.5% and 75.1%, respectively, of design error vulnerabilities that will allow unauthorized disclosure of information will be of high severity. In 2007 (X = 9), equation (3) predicts that 73.4% of the design error vulnerabilities that will allow unauthorized disclosure of information will be of low severity. The equation could not be used to predict the percentage value of these vulnerabilities for 2008.

Information availability was targeted the least by design error vulnerabilities between 1988 and 2006.

From the late 1990 onward, remote exploitation was the principal method of exploiting design error vulnerabilities. With the growing power of the Internet, it could be predicted that, remote mode of vulnerability exploitation will still be the principal method of exploiting vulnerabilities. The mode of exploitation whereby a target accesses an

attacker resource was found to be a least utilize method of exploiting design error vulnerabilities.
It was found out that, generally after 1998, the variation of the remote mode of exploitation of the design error vulnerabilities was inversely proportional the local mode of exploitation of the vulnerabilities. The variation of these two modes of exploitation mirrored each other is some cases. In this case, to a fair approximation, knowing the variation of one mode of exploitation could possibly predict the variation of the other. Figures 4.2.4.0.1, 4.2.4.3.0, 4.2.5.2.0 illustrate this fact.

All in all, the results seem to indicate that more security features were incorporated into software by software vendors during the period from 1988 to 2006. Nevertheless, more sophisticated ways of exploiting design error vulnerabilities, in particular, and vulnerabilities, in general, are continuously being developed by attackers and hackers.

7.0 Bibliography

[1] National Vulnerability Database: http://nvd.nist.gov/statistics.cfm

[2] NVD Common Vulnerability Scoring System Support: http://nvd.nist.gov/cvss.cfm

[3] Phil K. "How to Write an Abstract", 1997.

[4] Rahul T., Sunil W. "Impact of Software Vulnerability Announcements on the Market Value of Software vendors-an Empirical Investigation", 2005.

[5] http://www.first.org/cvss/cvss-guide.html#scoring

[6] http://cve.mitre.org/about/terminology.html

[7] Mark D. John M., Justin S. "The Art of Software Security Assessment Identifying and Preventing Software Vulnerabilities", 2006.

[8] http://amol.org.au/capture/course/glossary.html

[9] The Organization of Internet Safety (OIS): http://www.oisafety.org

[10] Computer Crime and Security Survey 2006 by Computer Security Institute: http://www.gocis.com

[11] http://en.wikipedia.org

[12] http://oval.mitre.org/

[13] http://securitytracker.com/alerts/2005/Dec/1015335.html

[14] Taimur A., Ivan K., and Eugene H. S. "Use of Taxonomy of Security Faults", 1996.

[15]http://www.computerworld.com/softwaretopics/os/windows/story/0,10801,92349,00.html

[16] http://www.detnews.com/2005/technology/0509/20/A04-320608.htm

[17] http://www.aic.gov.au/publications/htcb/htcb006.html

[18] http://www.softwaretechnews.com/stn8-2/noopur.html

[19] http://www.unesco.org/courier/2001_02/photoshr/46uk.htm

[20] http://en.wikipedia.org/wiki/Dot-com_bubble

[21] http://www.cert.org/tech_tips/win-95-info.html

[22] Kathy S. "Information Technology Project Management", Third Edition.

[23] Information Security Forum Special Interest Group: "Best Practice in Endpoint Security".

[24] http://www.microsoft.com/windows/products/windowsvista

[25] http://en.wikipedia.org/wiki/Trustworthy_Computing

[26] http://www.microsoft.com/mscorp/twc/default.mspx

[27] Steve L. “The Trustworthy Computing Security Development Lifecycle” March 2005.

[28] http://www.sis.pitt.edu/~mbsclass/standards/martincic/isohistr.htm

[29] http://17799-news.the-hamster.com/issue10-news7.htm

[30] http://www.owasp.org/index.php/Top_10_2007

[31] http://www.sans.org/top20/

[32] http://www.first.org/newsroom/releases/20070620-1.html