THE BEHAVIOURAL STUDY OF LOW INTERACTION HONEYPOTS: DSHIELD AND GLASTOPF IN VARIOUS WEB ATTACKS

Anujot Boparai, Ron Ruhl, Dale Lindskog
Master of Information Systems Security Management

Concordia University College of Alberta
7128 Ada Boulevard, Edmonton, AB T5B 4E4, Canada
aboparai@student.concordia.ab.ca
(587-708-5458)

ABSTRACT

Web application honey pots are one of the available solutions to track and understand the attack pattern and type of methods/techniques used by the attackers on the application. In this research paper, the study has been conducted to analyze the logging behavior of low interaction honeypots: DShield and Glastopf. To do this, different types of web attacks have been used to understand the behavior of the logging system and the response mechanism of honeypots against the different attacks by anonymous users called Hackers or intruders. In the experiment, we had deployed two honey pots (DShield and Glastopf) to analyze their capabilities as honey pots and how these systems have been used as a decoy system. The experiment had critically evaluated the strengths and capabilities of these honeypots and compared the logging information that these systems were capable of doing.

Keywords*- Web application honey pots, DShield, Glastopf, open source honey pots, low interaction honey pots.*

I. INTRODUCTION

Honey pot is a kind of weapon that can help the victim to understand his enemy and prepare for counter measures. There is a lot of research work that has already been done in honey pots which can help to detect the attacks in a web application and its services depending on the level of interaction with user. Web application security is one of the major concerns as this is the most targeted applications by the attackers and even the script kiddies attempt to hack the web sites by exploiting the various vulnerabilities of the web applications [1]. This is the fact that the attacks on the web applications are increasing eventually and they all have become one of the ways for the attackers or hackers to penetrate into the system for accessing the confidential information which might result into huge loss for the victims [2].

There are many counter measures that are taken to avoid the security issues in the website. However since new threats and security issues are emerging every day, it becomes critical to track these new issues, threats and methods so that appropriate counter measures can be applied and such issues can be prevented in future [3]. Web application honey pots are devised allowing the attackers to attack them and then extracting the information about the type of attacks and its attacker from their logging system. This is one of the reasons these web applications are made vulnerable to be attacked. This strategy of attracting the attackers helps the honey pots to capture the activities that the attackers which is later on analyzed by the system administrators to read and extract the information on these attacks. This strategy helps the administrators to study the different type of attacks that attackers use against these vulnerable applications, which are actually honey pots running in background and unknown to outside users [5].

In Real world attack scenario, an attacker tries to perform an attack on website firstly it gets interacted with IDS (Intrusion Detection System) which directs the traffic to either genuine website or honeypots depending on the legitimacy of the packets. The inspection of the packets depends on the type of IDS used. In this paper we are not going in depth to understand the IDS working. This paper was all about how honeypots have reacted to hose intrusions which have already been detected by IDS and sent to them (honeypots) for further evaluation.

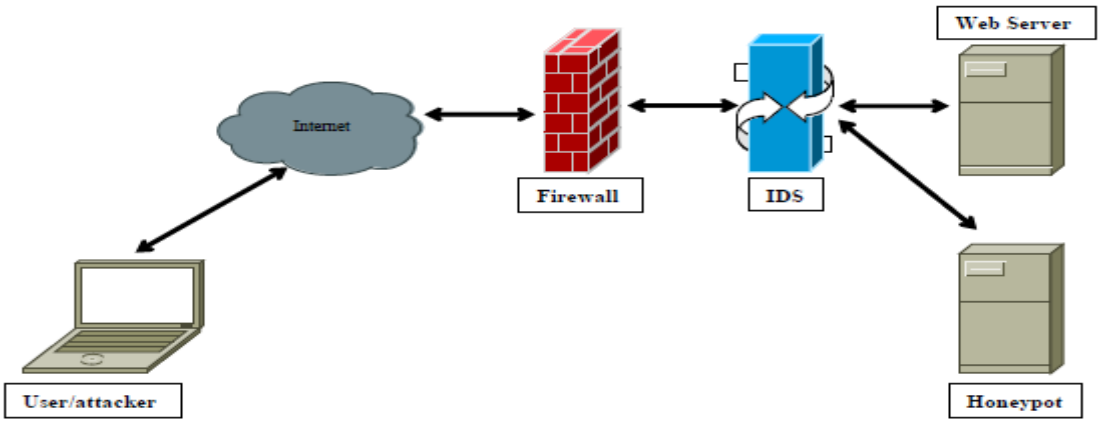

Figure-1: Real World Attack Scenario

In the figure above, we can see that Web application server has an IDS system deployed in front of them in order to direct the traffic to the web application and detect attack. If the user is genuine it is directed to the main web application but if the end user seems malicious then its request is directed to the web application honey pots which are devised as a duplicate copy of original website (i.e. which looks like the original one) made vulnerable for the attackers to attack. Although there are chances that genuine users are directed to the honeypots by IDS (False positive) and vice versa (False negative). This system helps track the patterns that attackers use to attack the honey pot. As told earlier this paper here does not focus on how malicious users are identified and traffic is redirected to the web application honey pots.

The web application honey pots are classified into different levels as per their level of interaction with the attackers. Table 1 below shows the effort of installation/configuration, deployment, information gathering and risk assessments that a user would experience when they use different types of web application honey pots based on their level of interaction.

Interaction	Installation/configuration	Deployment/Maintenance	Information gathering	Risk
Low	Easy	Easy	Limited	Low
Medium	Involved	Involved	Variable	Medium
High	Difficult	Difficult	Extensive	High

Table 1: Tradeoffs of honey pot level of interaction [7]

These web application honey pots log the information of an attack and the attackers into the system depending on their interaction level with the attackers i.e. low interaction level honey pots have the least and limited interaction with the attacker whereas the high interaction honey pots have a high responsive nature and can collect more appropriate trace of the attacks [9].

DShield Web Application Honeypot

DShield web application honey pot is one of the projects and was developed using PHP. This is a low level interaction project that provided a very limited response to the attackers and can be molded easily to respond well to the attackers [10]. This honeypot was last updated in 2010 but is still not outdated.

Glastopf Web Application Honeypot

One of the other low level interaction web application tools is ***Glastopf*** which is dynamic in nature and is strong enough to capture various types of attacks. This honey pot deployment works on a basic principle to respond to the attacker in a best possible way [11].

Google hack honey pot was also known to be a web application honey pot and is no longer active due to lack of its community support. It used to work on modified templates for logging the attacks information to the honey pot.

II. LITERATURE REVIEW

Attacks on web applications are done to gain confidential information from the system. This is the reason why a system is required to study these attacks and prepare for counter measures that can be used to prevent such attacks. Honey pots are useful in this regard as they act as decoy systems and inspect the traffic and malicious activities that are happening on a system by the attackers. These honey pot systems act as the original system, but are usually a replica of the original system that attracts the attackers and then studies about the different type of attacks and exploits that are used to compromise the system [12].

In 2010, it was observed that the web applications attacks reached a level of 60% out of total attacks done on varied type of applications [13]. This brings into attention that there is a need of a method or tool to study these attacks and make the web application protected from them. Honey pots are one of the best ways to detect such attacks like XSS (Cross-site scripting), SQL injections, Local file inclusions etc. on web applications.

A honey pot application earlier was deployed to detect the distributed denial of service against the network. This honey pot was also applicable for the web applications to track the DDOS attacks on them. In this proposed system, honey pots were deployed to the production servers to learn the attack patterns from the DDoS attack acting as a decoy system. The Logic applied behind this was that a whole traffic received within the honey pot is considered as an attack packet most likely. Using this design, it gets difficult for the attackers to known that the systems they are attacking are honey pots and since these honey pots keep on changing their IP addresses so during the DDOS it gets difficult for the attackers to know the exact location of the server. Moreover these honey pots camouflage their IP address from the pool belonging to server block. In this honey pot, the biggest challenge is that because the attackers IP address comes from multiple administrative domains it is difficult to coordinate among different ISPs for accepting messages from the honey pots [14].

Another solution was introduced with architecture for low level interaction honey pot called "Honeyd@Web" which was usable at production site and was light weight and secures to use. However it was able to do a very low level of interaction with the web applications. This deployment of honey pot was able to mimic as a web server to outside world and can log the attacks [15].

Most of the web-application honeypots and honeypot deployment schemes focus on protecting e-commerce based web-application systems [16]. Jacob Farneth and other researches proposed a way to analyze and thus create a low interaction web-application honeypot that mimics the behavior of a web-application running on an embedded system. They analyzed web server requests and responses and used it to develop emulation programs for low interaction honeypot that is running on an embedded system [17].

III. DSHIELD AND GLASTOPF

DShield and Glastopf are low interaction web application honey pots. The basic nature of the web application honey pots is to redirect only the malicious users to the web application honey pots instead of genuine users. This makes it important to initially trace which users are genuine and which ones are malicious users. Once the malicious users get redirected to the web application honey pots then they can be captured by the system as per the configurations done in the honey pot systems. In order to perform the study of the two low level interaction honey pots, a test environment set up was done as follows,

1. DShield Environment

DShield is a low interaction level web application honey pot that was deployed in CentOS 6.4 Linux Server. As per given figure-2, Apache web server was hooked with the PHP script engine along with logging. This PHP implementation was accessible via a URL and was a dummy site to perform the experiment. However in production site users are redirected to this site once they are detected to be malicious. Now, this web application can be attacked and logs generated within the Dshield deployment can be analyzed [19].

Figure-2 below shows the virtual environment that was setup to simulate the experiments. To replicate the malicious user, a virtual instance with IP: 192.168.140.134 was used. This user is considered to be a malicious user who will perform different security attacks on the honey pot. On the other hand instance with IP: 192.168.140.132 represents the DShield honeypot. The web application that is running on this instance was installed as part of DShield installation itself which is also hooked to the logs in Apache server.

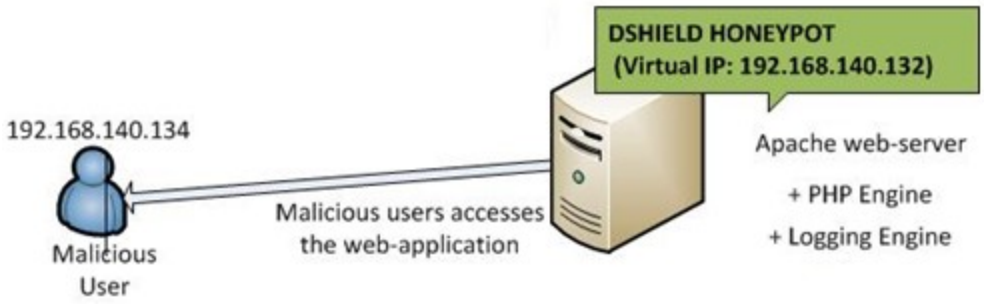

Figure-2: Design of DShield honey pot implementation

Initially Apache server and PHP libraries were installed on the Centos server. Secondly, DShield honey pot package was downloaded and installed. After installing the source virtual hosting was setup on Apache server by updating the configuration files. Following is the important configuration lines used in configuration settings of Apache server:

```
<Directory "/webdir/webhoneypot/html">
    RewriteEngine on
    RewriteRule ^(?!index.php)(.*) index.php/$1
</Directory>
```

These settings started redirecting the logs to Apache, which then were analyzed to trace the attacks that were simulated during the experiments. Once the setup was done, honey pots application was accessible through a URL (www.webtohoneypot.com).

To setup log analyzers in web application honey pot, a third party python based tool called "Apache-scalp" was used during the analysis of the logs that were generated in Apache server. This python utility has the capability to analyze and extract the logged information to a hypertext based file which is useful for the administrator to do the analysis.

2. Glastopf Environment

Glastopf is also one of the low interaction web application honey pot which can effectively trace varied web-application attacks like SQL injection. It supports multistage attacks, a vulnerability emulator and list of vulnerable requests, rather than the modified web app templates used by search engines to attract more attacks over time. Glastopf is deployed on Ubuntu 12.04 Linux server. This honey pot implementation uses python scripting along PHP Sandbox installed in server. It has a good capability of logging based on the interaction that

attacker would have with the application. Because of its low level interaction nature, it is capable of tracing limited attacks [20].

Figure-3 below shows the virtual environment that was setup to simulate the experiments. To replicate the malicious user, a virtual instance with IP: 192.168.140.134 was used. This user is considered to be a malicious user who will perform different security attacks on the honey pot. On the other hand instance with IP: 192.168.140.133 represents the Glastopf honeypot. The web application that is running on this instance was installed as part of Glastopf installation itself.

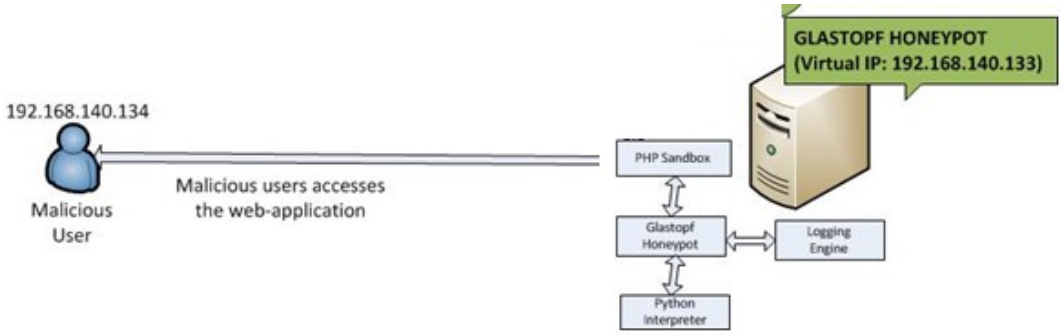

Figure-3: Design of Glastopf honey pot implementation

Figure-3 above shows the design of the testing environment that is deployed, tested and then analyzed. Initially all the required packages like python openssl library, php5 development libraries, C and C++ compilers were installed to the Ubuntu server. After these installations, a PHP sandbox was installed that would run the PHP applications within the sandbox. This set-up the pre-requisite environments after which Glastopf packages were downloaded and installed to the server. After the installation of the honey pot, Glastopf configuration and environment files were updated to redirect the logs to a log file and Database file in SQLite. Once the setup was done, the honey pot application was then accessible through a URL (www.webtohoneypot.com).

In order to Setup log analyzers in web application honey pot, logs within the Glastopf honeypot were redirected to a SQLite database file which needs SQLite package installed within the system to read the logs.

Environment setup details:

DShield	Glastopf
Apache server: 2.2.15 ***Dshield package:*** v3 with upgrades release with r123-1. ***Operating system:*** CentOS 6.4 (last updated: 07/2011)	***Glastopf package:*** 3.0.9 (python 2.7) ***Operating system:*** Ubuntu 12.04.1 (last updated: 12/2012)

Table 2: Environmental setup

The above table shows the description of the web application honey pots that were used during the environment implementation. These honey pots were released with last updates at different time. Glastopf has the recent updates due to which it now includes the capability to emulate the SQL injections.
These environments were built up using the virtual machines in VMware environment.

IV. EXPERIMENTS ON DSHIELD AND GLASTOPF

1. Scope of Experiments

Due to limited functionality of the web application honey pots, they support following attacks (i.e. any user who tries to attack these honey pots then the system administrator will be able to view the traces and its patterns from the generated logs):

- SQL Injection
- Cross-Site Scripting
- Remote File Inclusion
- Local File Inclusion

The focus of the experiment has been around these attacks. These web application honey pots capture the attacks and methods that were used to simulate the attacks which further helped the administrators to study and analyze these logs and build up the security for the web applications. During the experiment different attacks were been simulated on the web application honey pots and then their logs were analyzed.

2. Assumption(s):

The experiment assumed that the end user due to its malicious activities has been redirected to the web application honey pot to trace all the activities (as given in Figure-2 and 3) performed by attacker. These experiments used external tools to attack the web application honey pots. Note that these experiments can be simulated using any tool/techniques, but the idea here is to demonstrate the critical differences that these web application would exhibit as per their capabilities. In these experiments, backtrack image was used by hooking it into virtual environment where honey pot were running as a web application. This simulates a similar process where end user is accessing the web application without knowing that he is actually accessing the honey pot based web application. Experiments then were performed in virtualized environment using various penetration testing and exploitations tools like Arachni, W3AF, Nikto and Wapiti. The attacks only focused on the following vulnerabilities SQL Injection, Cross-Site Scripting (XSS), Command Injection, Remote File Inclusion and Local File Inclusion as they are the top listed attacks in OSWAP (Open Web Application Security Project).

3. Selection of tools for attack simulations and their significance*:*

- **W3af version 2.0** short for Web Application - Attack and Audit Framework is an open source tool for vulnerability assessment in the web application. This tool is written in Python and is widely known to be used for scanning web application(s) against SQL injection (sqli), local file inclusion (lfi), cross site scripting (xss) etc. W3AF can detect and then exploit the XSS vulnerabilities in the web application(s) while others tools are capable of detecting XSS only.
- **Nikto version 2.1.5** was used for browser injection. Browser injection(s) is a most common way that the attackers would use to evaluate the website and retrieve the information. These browser attacks would use code injection like local file injections and remote file injections.
  wget is a common http client used within Linux platforms. There is possibility to retrieve sensitive information of the servers by attacking the web application running on Linux servers and the attackers can query the web directories to get information and then exploit them.
- **Sqlmap version 1.0** is a tool used for performing SQL injections to a website and is one of the well - known tool used for security testing. Most of the attackers also would use such tools to invade the website.

Using these tools, we stimulated the attack by sending a malicious request; the honeypots processed the request, logged the request and replied to the attack. DShield and Glastopf have almost similar working techniques.

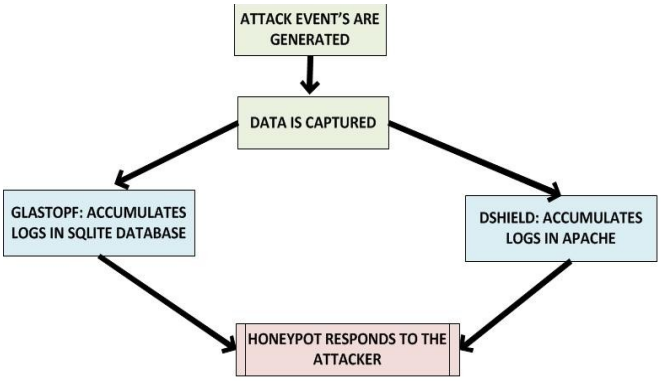

Figure-4: Attack events captured by Dshield and Glastopf

4. Experiments and evaluation*:*

A. SQL injection(s)

SQL injections are one of the crucial attacks that the attacker would like to simulate in real environment to get access to the database or any files within the database. Automated as well as manual scan were executed against the Dshield and Glastopf honey pots. All the automated scans were executed using the SQL map version 1.0 tool using different standards of level. This level executes the SQL injections at different depth levels.

Glastopf emulates the SQL injections whereas in case of Dshield it is known that SQL injections are not supported. Automated scan that was executed against the Dshield did not capture any logs during the experiment. SQL injections are emulated by the Glastopf since it has an add-on attached within its package. Glastopf responds to the SQL attacks due to its supportive configuration and an addition of the SQL attack handler plugin. It does this by responding to the attackers with the paths that they expect after the SQL injection to the website. These path based vulnerability signature are called 'dorks' which are used as a bait for attackers [13]. Number of logged events represents that output that was exposed to the logs which corresponds to the different permutation/combinations that were emulated by Sqlmap. Glastopf responded with "200 status ok" which in real world means Glastopf accepted the attack and confirmed the user.

SQL injection attack patterns	No. of logged events	Pattern
./sqlmap.py --url 'http://192.168.140.133/index.html' --data='username=test& password=test&server=1&lang=en-iso-8859-1&convcharset=iso-8859-1' -p 'username' --level 2	**770**	**Unknown**
./sqlmap.py --url 'http://192.168.140.133/index.html' --data='username=test& password=test&server=1&lang=en-iso-8859-1&convcharset=iso-8859-1' -p 'username' --level 3	**226**	**Unknown**
./sqlmap.py --url 'http://192.168.140.133/index.html' --data='username=test& password=test&server=1&lang=en-iso-8859-1&convcharset=iso-8859-1' -p 'username' --level 4	**322**	**Unknown**
./sqlmap.py --url 'http://192.168.140.133/index.html' --data='username=test& password=test&server=1&lang=en-iso-8859-1&convcharset=iso-8859-1' -p 'username' --level 5	**1620**	**Unknown**
Manual attacks - Pass on the string to username/password as: • ' or '1'='1 • ' or '1'='1' -- ' • ' or '1'='1' ({ ' • ' or '1'='1' /* '	**3**	**Unknown**

Table-3: SQL injection attack pattern(s) in Glastopf

Table-3 shows the different set of attacks that were executed against the Glastopf web application honey pot. Manual attacks were also performed on the Glastopf instance. The output is generated in the SQLite database within events table. All the entries are created within the database with different patterns. SQL injections are captured as 'unknown' pattern. The figure-5 below shows the event that was captured for a single SQL that is injected to with website.

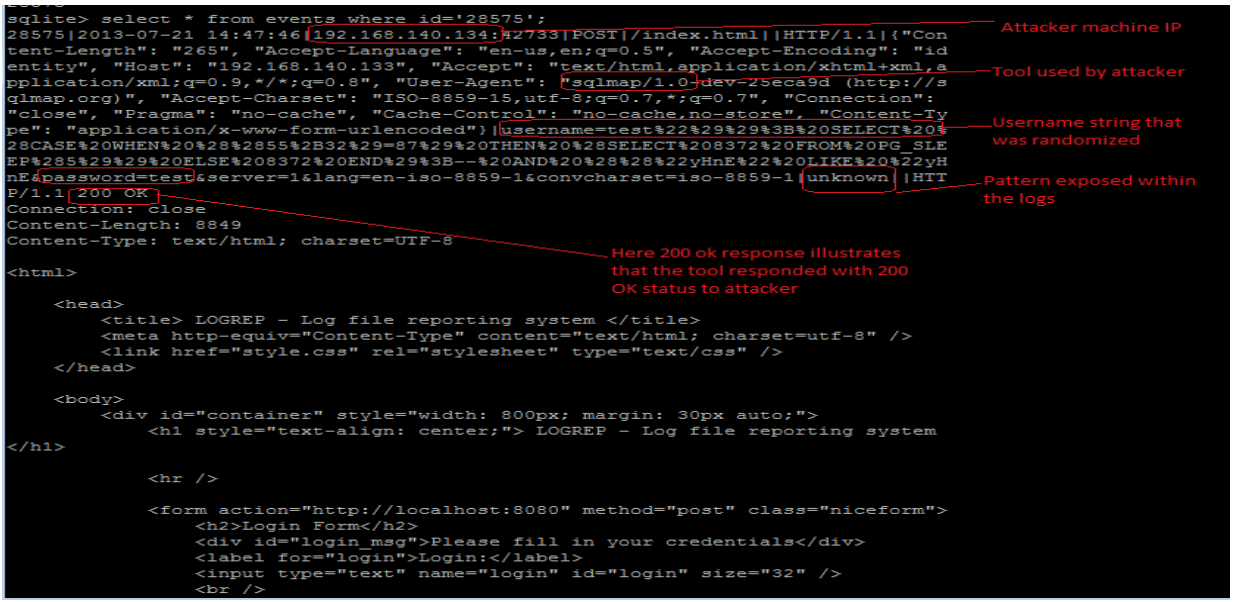

Figure-5: Glastopf sqlite database

B. Cross-site scripting (XSS):

Cross-site scripting attack is one of the known attacks that are performed by the malicious users on the web applications. This is one of the attacks that are listed in the OWASP (Open source web application security project). In order to replicate the cross-site scripting attacks W3AF tool was used to simulate the attacks to the honey pots deployments. W3AF tool is used at this point to inject cross site scripts in form of regular expressions to the application running on the honey pot deployment. Here it is assumed that the end user is been redirected to the web application honey pot since it's detected that he is a malicious user.

W3AF initiated scan to web application with selection of 'XSS' plugin. Figure-6 shows the regular expressions that are captured within the Dshield logs. The logs that are generated after the attacks are then extracted with the use of an apache scalp script that reads the apache log and extracts the results in HTML form.

Cross site scripting attack patterns	No. of logged events (Dshield)	Pattern (Dshield)	No. of logged events (Glastopf)	Pattern (Glastopf)
Manual injection: <script>alert(document.cookie)</script>	1	XSS	1	Comments
Manual injection: <script>alert('This is a alert message')</script>	1	XSS	1	Comments
Manual injection: index.html?title=<meta%20http-equiv="refresh"%20content = "0;">	1	XSS	1	Comments
Automated scan: W3AF executed with selection of 'XSS' plugin	890	XSS	1391	Comments

Table-4: Cross-site scripting attack pattern(s) in Dshield and Glastopf

Table-4 demonstrates the different cross-site injections that were performed on the Dshield and Glastopf web honey pots. The first three injections were performed manually whereas W3AF was used to perform the automated scan using the plugin as 'XSS'.

During the manual and automated scan Dshield captured the logs successfully within the Apache logs. These apache logs were then run through the python script 'apache-scalp' which calculated the impact as 5 and pattern as “XSS” i.e. cross-site scripting. In order to look for the attackers information Apache logs can be used which also shows the time event.

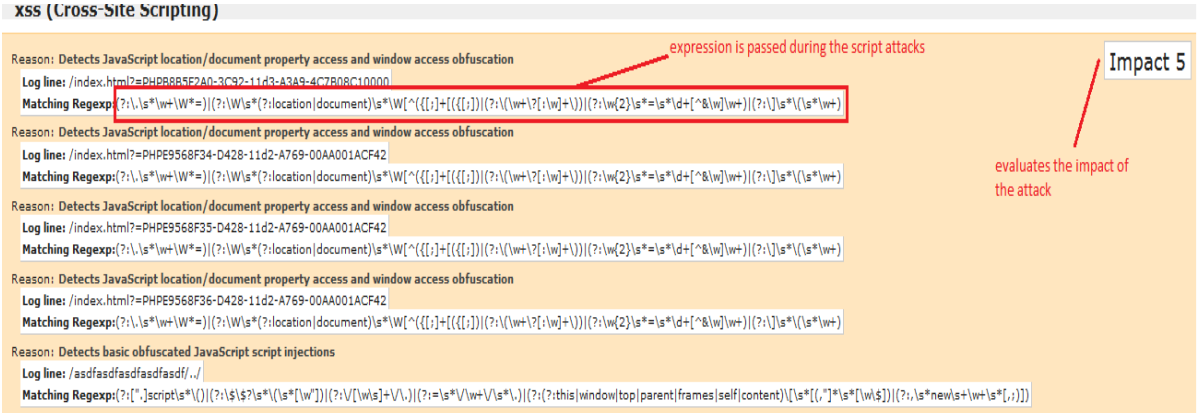

Figure-6: Dshield logs as a result from the automated w3AF scan

In case of Glastopf, all the cross site scripting attacks were captured successfully but with the pattern name as 'comments' which is not very useful. This pattern name does not correspond to the actual cross-site injection which makes it harder for the administrator to identify the attack. The only way administrator can identify that the attack was performed is through the tool information like W3AF that can be found within the logs.

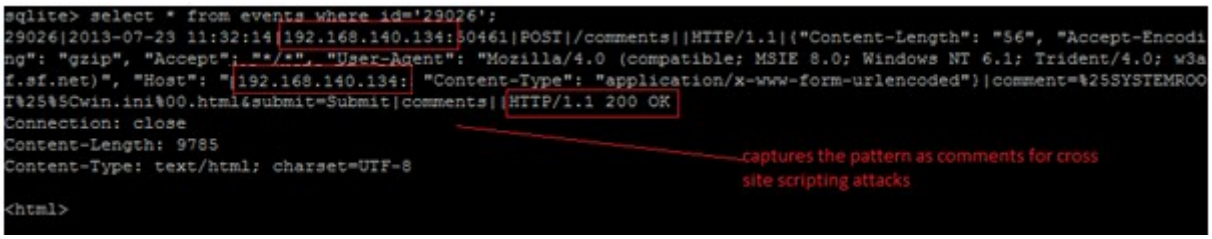

Figure-7: Glastopf log event captured as a result from the automated w3AF scan

Glastopf also successfully logs the attacker’s information in form of IP address and time of event in one place.
The other observation that is related to the number of logged events demonstrates the different numbers of logging information that was exposed within the logs. In case of Dshield less number of logged events is shown since it takes every request as a HTML request and responds to it whereas on the other hand Glastopf differentiates the CSS (custom style sheets) from the HTML and thus increasing the logged events count.

C. Remote and local file inclusion attacks

For simulating the remote and local file inclusion attacks browser injections were performed. RFI (Remote File Inclusion) give the attacker’s way to inject code using remote files on the web application which can then be executed at the server end. This makes it very risky for the sensitive information to get exposed. LFI (Local File Inclusion) is similar to remote file inclusion as this is also a technique of injecting a file to the web server to expose the sensitive information from the server but in this technique a local file from the system is used for the attack. A common example that can be used to simulate such attack is by injecting the local or remote files to the web address with the help of “?” character like after the web application URL include the address "? lib_path=/etc/passwd'.
For automated scans Nikto 2.1.5 was used with plugin for remote file inclusion and local file inclusion.

Remote file inclusion attack patterns	No. of logged events (Dshield)	Pattern (Dshield)	No. of logged events (Glastopf)	Pattern (Glastopf)
Manual injection: "?lib_path=/etc/passwd"	1	RFI	1	RFI
Manual injection: "?lib_path=/etc/ passwd %00"	1	RFI	1	RFI
Automated scan: Nikto executed with selection of 'RFI' plugin	43	RFI	34	RFI

Table-5: Remote file inclusion attack pattern(s) in Dshield and Glastopf

Local file inclusion attack patterns	No. of logged events (Dshield)	Pattern (Dshield)	No. of logged events (Glastopf)	Pattern (Glastopf)
Manual injection: "?Sample=C:\\test\\exploit"	1	LFI	1	LFI
Manual injection: "?Sample2=C:\\test1\\test.php"	1	LFI	1	LFI
Automated scan: Nikto executed with selection of 'LFI' plugin	12	LFI	23	LFI

Table-6: Local file inclusion attack pattern(s) in Dshield and Glastopf

These simulations of the results into the execution of the scripts over the server could result in damaging the server files or could delete them as well. Dshield and Glastopf both were able to simulate the local file and remote file inclusion attacks. Glastopf in its database captures these patterns in form as RFI and LFI which could be queried from the database along with the trail of events that were executed against the web application. There are different events logged by the honey pots whereas Glastopf is more dynamic in nature and also captures the HTML information within the request and response from the honey pot server. The success response codes “200 OK” confirms that the attack were successful.

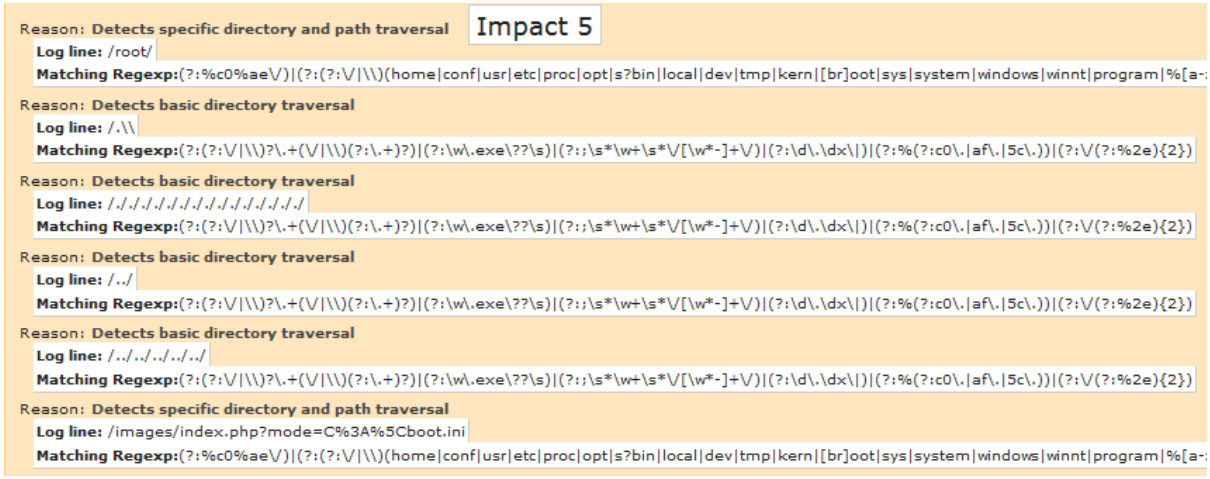

Figure-8: Dshield logs as a result from the automated Nikto scan for Remote file inclusion

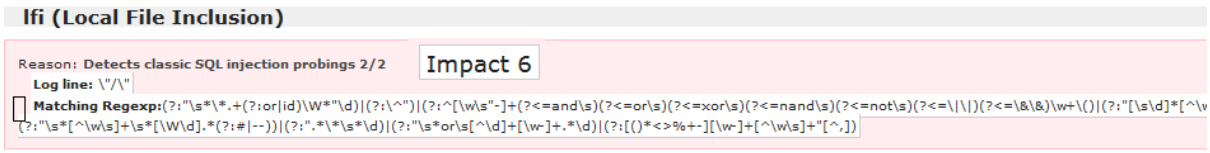

Figure-9: Dshield logs as a result from the automated Nikto scan for Local file inclusion

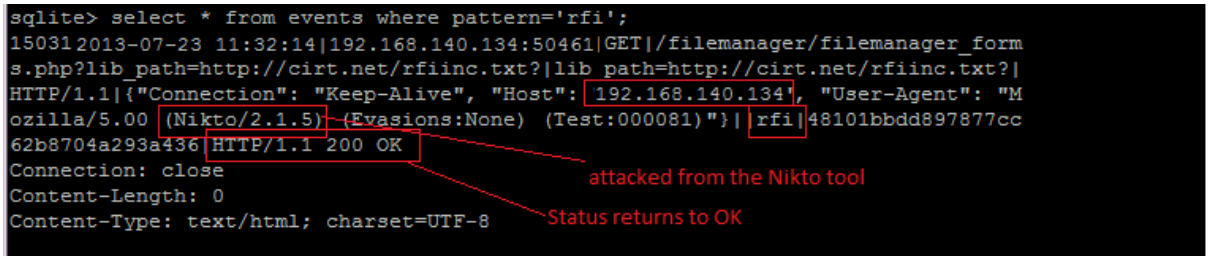

Figure-10: Glastopf logs as a result from the automated Nikto scan for Remote file inclusion

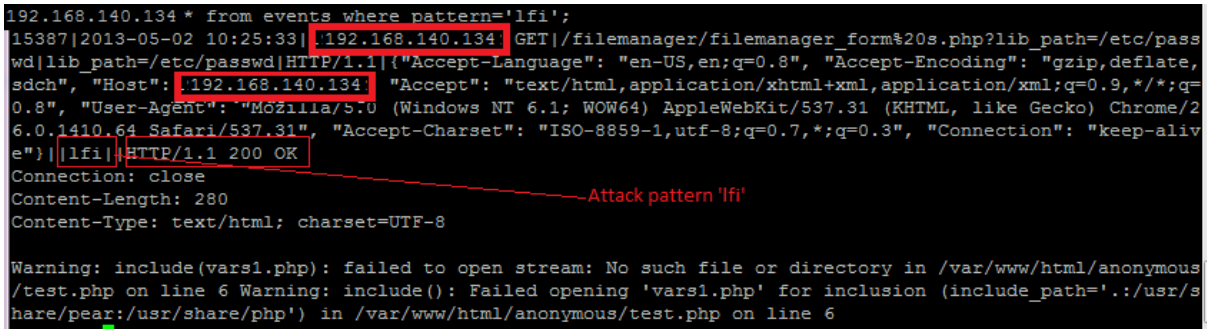

Figure-11: Glastopf logs as a result from the automated Nikto scan for Local file inclusion

Figure 6-11 shows the log snapshots for a single event of attack. Dshield and Glastopf both record the attack pattern as 'RFI' and 'LFI'. However when these logs are compared Glastopf logs show better information in terms of attacker's activity with time of event, tools used, pattern, HTML track, response code and also detailing the different parameters used during the attack. But in case of remote file inclusion Dshield logs more attacks.

V. OBSERVATIONS

The logging information that was exposed from these experiments help us evaluate strengths and limitations of these web application honey pots.

- **Strengths:**
  - Dshield
    - Is highly capable of categorizing the attacks within the logs.(From Table 4)
    - Captures the cross site scripting attack with the name of pattern “XSS” making it easy for the administrator to detect the type of the attack.
    - Third party tools like apache-scalp can be used to hook with the Apache sever logs and extract the reports. These reports show the high level extract of the logs.
    - Logs capture the IP address of the attacker and time of attack as well.
  - Glastopf
    - It is easy to understand the logs that are captured. SQLite database file can be queried to get the information of individual events.
    - It supports SQL injections emulation (From Table 3).
    - Logs capture the IP address of the attacker and time of attack as well.
    - Logs also capture the HTML information that is posted within the request and response from the server.
    - Logs also capture the response code (200 OK) confirming that the attack was successful.

2. **Limitations:**
    - Dshield
        - Does not emulate and log the SQL injections.
        - It is not easy to go through the logs and understand the attackers easily. All the information that is recorded within the Apache log is also not very descriptive. (Figure 6,7,8)
    - Glastopf
        - Captures the SQL injections with pattern name as 'unknown' instead of SQL due to which Administrator has to go through all of them to trace the attacker's activities. (From Table 3).
        - Captures the Cross site scripting with pattern name as 'comments' instead of XSS which again makes it difficult for the Administrator to go through all of them and trace the attacker's activities. But Dshield on the other hand captures the attack pattern as "XSS" making it easier to know the type of attack just by looking. (From Table 4).
        - The logs generated for Remote File Inclusion attacks were lesser as compared to Dshield. (From Table 5).

From the above experiments we found that, Dshield despite of being not updated from 2010 still is efficient in many cases as compared to Glastopf in detecting and logging the pattern of the attack. Although Glastopf provide more information about attacker and can help back in tracing the attacker where Dshield lacked in this area.

VI. CONCLUSION

Finally it can be concluded, the attacks which were performed on Dshield and Glastopf have clearly revealed that both have their own capabilities and strengths. The logs generated by both of them are helpful and provides enormous amount of information which can be retrieved and can help back in tracing the intruders or malicious hacker. In case of Glastopf, these logs can trace back the location i.e. IP of the hacker which is very unique and important for administrator to take actions.

Glastofp regardless of being updated and having more advanced features like SQL handler plugin it still can't tell the pattern of the attacks for SQL injection and XSS. Whereas Dshield except SQL injection performed fairly well in all attacks as compared to Glastofp especially in remote file inclusion attack where it generated more logs than Glastofp. Also for manual attacks the numbers of logs were same in both cases.

VII. FUTURE WORK AND RECOMMEDATION

Dshield has not been in play since 2010 which is why it has less features as compared to Glastopf. The Best part for both of the web application honey pots is that they are open source and their code can be extended to support more features.

For future work, we can perform more experiments on these two web application honeypots using different tools and attack in order to check their capabilities and explore them at much greater extend. Other features can be introduced to update the Dshield so it can also enhance its capabilities with the existing one.

VIII. REFERENCES

[1] Valli, C. "Honeypot technologies and their applicability as an internal countermeasure", *International Journal of Information and Computer Security, Inderscience Publishers, Geneva, Switzerland*. Pages: 430-436, 2007.

[2] Huang, Y., Huang, S., Tsai, C. and Lin, T. "Web Application Security Assessment by Fault Injection and Behavior Monitoring", *Proceedings of the 12th international conference on World Wide Web, ACM*.
Pages: 148-159. 2003.

[3] Mao, C. "Experiences in Security Testing for Web-based Applications". *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ACM*. Pages 326-330. 2009.

[4] Provos, N. and Holz, D. "Virtual Honeypots: From Botnet Tracking to Intrusion Detection", Addison-Wesley, 2008.

[5] Holz, T. (2007). Learning More About Attack Patterns with Honeypots. Available at: http://pi1.informatik.uni-mannheim.de/filepool/publications/learning-more-about-attack-patterns-with-honeypots-1.pdf

[6] Provos, N. (2004). A Virtual Honeypot Framework. In Proceedings of 13th USENIX Security Symposium, pages 1–14, 2004.

[7] Spitzner, L. (2002). Honeypots: Tracking Hackers. Pearson Education Inc.

[8] Joho. D. (2004). Active Honeypots, M.Sc. Thesis, Department of Information Technology, University of Zurich, Switzerland. Available at: http://www.ifi.unizh.ch/archive/mastertheses/DA_Arbeiten_2004/Joho_Dieter.pdf.

[9] Yagi, T., Tanimoto, N., Hariu, T and Itoh, M. (2010). "Enhanced Attack Collection Scheme on High-Interaction Web Honeypots". IEEE Symposium on Computers and Communications (ISCC). Pages: 81-86.

[10] DShield Web Honeypot Project. Available at: https://sites.google.com/site/webhoneypotsite/.

[11] Vetsch, S., Kobin, M. and Mauer, M. (2010). Glastopf. A dynamic, low-interaction web application honeypot.

[12] Li-juan, Z. "Honeypot-based Defense System Research and Design". *Computer Science and Information Technology, 2nd IEEE International Conference*. Pages: 466-470. August 2009.

[13] Rist, L. 2010. Glastopf: A dynamic, low-interaction web application honeypot. Available at : http://honeynet.org/files/KYT-Glastopf-Final_v1.pdf

[14] Khattab, S., Melhem, R. Moss, D. and Znati, T. 2006. Honeypot Back-propagation for Mitigating Spoofing Distributed Denial-of-Service Attacks. Department of Information Science and Telecommunications,University of Pittsburgh.

[15] Honeypot through Web (Honeyd@WEB): The Emerging of Security Application Integration Nor Badrul Anuar, Omar Zakaria, and Chong Wei Yao University of Malaya, Kuala Lumpur MY.

[16] Yagi, T., Tanimoto, N., Hariu, T. and Itoh, M. 2011. Enhanced Attack Collection Scheme on High-Interaction Web Honeypots. NTT Information Sharing Platform Laboratories, NTT Corporation.

[17] Farneth, J, Dhanju, A. and Blum, J.J. 2012. Analysis of Embedded Web Applications for Honeypot Emulation Programs.

[18] Harper, A., Harris, S. and Ness, J. "Gray Hat Hacking: The Ethical Hackers Handbook", Edition: 3, McGrawHill, 2012.

[19] DShield Web Honeypot Project. Available at: https://sites.google.com/site/webhoneypotsite/

[20] http://glastopf.org/