Web Application Honeypot: Glastopf

Sarosh Petkar
CSEC 731

I. INTRODUCTION

Web services are an integral part of modern day web applications and are defined as software systems that are made available over the Internet. This ever growing architecture brings with it new vulnerabilities, and are often the primary target of attackers; they are most likely to exploit unsanitized user input with attacks such as Buffer Overflow, Cross Site Scripting (XSS) and SQL/XML Injections. Attacks against web applications make up more than 60% of the total number of attacks conducted on the Internet. As a result, organizations cannot afford to allow their websites to be compromised, thereby, leading to leakage of customer's data. Thus, in order to prevent and study such attacks honeypot systems are used. Honeypot technologies are termed as security resources whose main focus is to be probed and compromised while at the same time collect high value data.

One of the widely used honeypots is Glastopf, it is a low interaction python-based honeypot that possesses the capability to exploit web application weaknesses such as SQL injection, remote and local file inclusion. Another web services honeypot called as 'WSpot' emulates SOAP (Simple Object Access Protocol) in order to log and register all activities but faces similar shortcomings as Glastopf. [1]

In this paper, a low-interaction client honeypot is discussed that can bite; it emulates certain vulnerabilities that can be exploited using various XSS and SQL exploitation methods. The principle behind it is very simple: reply to the attack, using the response the attacker is expecting from his attempt to exploit the web application. It does this by, exposing paths indicating the existence of a known vulnerability to search engine crawlers.

The next few sections describe the background related to the field of interest and then we will dive into the implementation of the model. Several tests will be conducted to gauge the honeypots performance.

Keywords: honeypot, web application, injection, Glastopf

II. BACKGROUND

Honeypots are defined as entrapment systems that are aimed at deceiving malicious users into launching a variety of attacks against servers. These decoy systems are intentionally left vulnerable in order to collect detailed information about the attackers. They are deployed in order to provide some level of protection for organizations or even in some cases as research units to study and analyze the methods employed by attackers to inject malicious software.

Honeypots can be classified as either low-interaction, high-interaction or hybrid. Low interaction honeypots are used to provide limited interaction for an attacker. All the services offered by such a honeypot are emulated as their sole purpose is to analyze the request and determine its true nature. The advantages of it are its increased speed and low resource consumption.

Whereas, High interaction honeypots are fully functional systems that imitate actual systems. This type of honeypot excels at detecting new attacks but at the same time they are more time consuming. Finally, Hybrid honeypots incorporate classification methods used by the other two categories in a cost effective large-scale environment. This system tends to outperform high interaction honeypots. [2]

Interaction	Installation	Deployment	Information gathering
Low	Easy	Easy	Limited
Medium	Involved	Involved	Variable
High	Difficult	Difficult	Extensive

Table 1: Tradeoffs of honeypots

Related work in this area involves the use of a low interaction client honeypot to simulate a client when interacting with the server. An example of such a tool is HoneyC, Monkey-Spider and SpyBye. Similarly, another low-interaction client honeypot named Honeyware possesses the ability to simulate different web browsers, to scan target files by virtue of different scan engines. According to the author a few advantages of it are that it can rectify the use of IP tracking to delay malicious activity by visiting the website multiple times to give the appearance of a normal human behavior. [9]

Other related work in this field involves the use of a low-level interaction honeypot called "Honeyd@Web". It has the ability to do a very low level of interaction with the web applications. The deployment of this honeypot was able to mimic a web server to outside world and log the attacks. [5]

Another honeypot is the High Interaction Honeypot Analysis Toolkit (HIHAT), a major drawback of this system is that in the case that it is hijacked, there is no mechanism to gain control of the system. Finally, Google Hack Honeypot (GHH) generates pages from Google hacking database, which provides a security hole that can be detected by a scanner tool. This is an efficient technique to deceive malicious actors. [6]

III. METHODOLOGY

The model consists of a low-interaction honeypot called Glastopf that provides features such as emulating a real server and replies to the attacker using the response the attacker is expecting. It supports multi-stage attacks and can effectively trace varied web application attacks. The way the tool works is, when the request to the honeypot is a general HTTP request then the honeypot will provide a regular response. In the scenario, there is an indication of any threat then the honeypot will emulate the attack and send a response pretending that the attack had been successful.

Glastopf is deployed on Ubuntu 14.04 Linux server and its path-based vulnerability signatures 'dorks' serve as bait for attackers. On seeing these paths in search engines the attacker will try to attack it. The system can use predefined injection dorks built for known vulnerabilities, but can also build new dorks from the attacks it sees by automatically adding the paths attackers use to try to access the database. It is set up to investigate deceptive qualities when queried through well-crafted requests. [3]

For experimentation, the first step was to test Glastopf in a virtual sandbox environment and then in the second stage it was left to work its magic in the wild. As part of Phase 1, simulations were performed using various penetration testing and exploitations tools like Nikto, Vega and Skipfish. The attacks focused on vulnerabilities such as Cross-Site Scripting (XSS), Command Injection, Remote File Inclusion and Local File Inclusion as they are the top listed attacks in OWASP.

Tools used for attack simulations:

❏ Nikto: It examines a website and reports back to you the potential vulnerabilities that it found that you could use to exploit or hack the site.

- ❑ Vega: Vega is an open source scanner and testing platform to test the security of web applications. It helps to find and validate SQL Injection, Cross-Site Scripting (XSS), and other vulnerabilities.

- ❑ Skipfish: Skipfish is a web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes.

For simulating the various attacks, browser injections were performed:

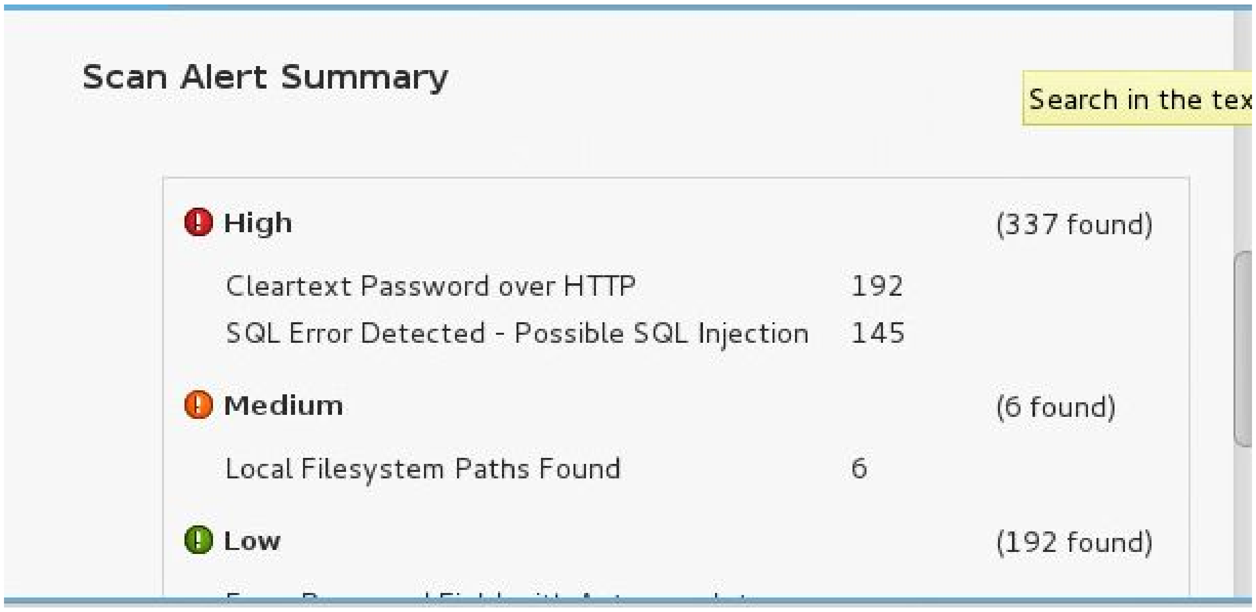
- ➔ Cross-site scripting (XSS) is a type of injection in which malicious scripts are injected into web sites. The end user's browser executes the script without any kind of restriction.
- ➔ Remote File Inclusion (RFI) gives the attackers a way to inject code using remote files on the web application, which can then be executed at the server end.
- ➔ Local File Inclusion (LFI) similar to remote file inclusion, is a technique of injecting a file to the web server in order to expose sensitive information from the server but in this method a local file from the system is used for the attack.

After the testing was completed, phase 2 was initiated. Herein, the honeypot was then made available to the world for a period of 2 weeks. During this time a plethora of attacks were captured and examined. The logs in the database consisted of the IP address of the attacker, time of attack, http response code and others. This information was then used to make a map showcasing the geo-location of the source of the attack.

IV. RESULTS

Phase 1:

Using the aforementioned tools and other techniques, attacks were stimulated by sending a malicious request; the honeypot then processed and logged the request in the database. The Vega and Skipfish results (Figure 1 and Figure 2) were used to understand what kind of attacks would work against the system.


Scan Alert Summary

High		(337 found)
Cleartext Password over HTTP	192	
SQL Error Detected - Possible SQL Injection	145	
Medium		(6 found)
Local Filesystem Paths Found	6	
Low		(192 found)

Figure 1: Vega scan results

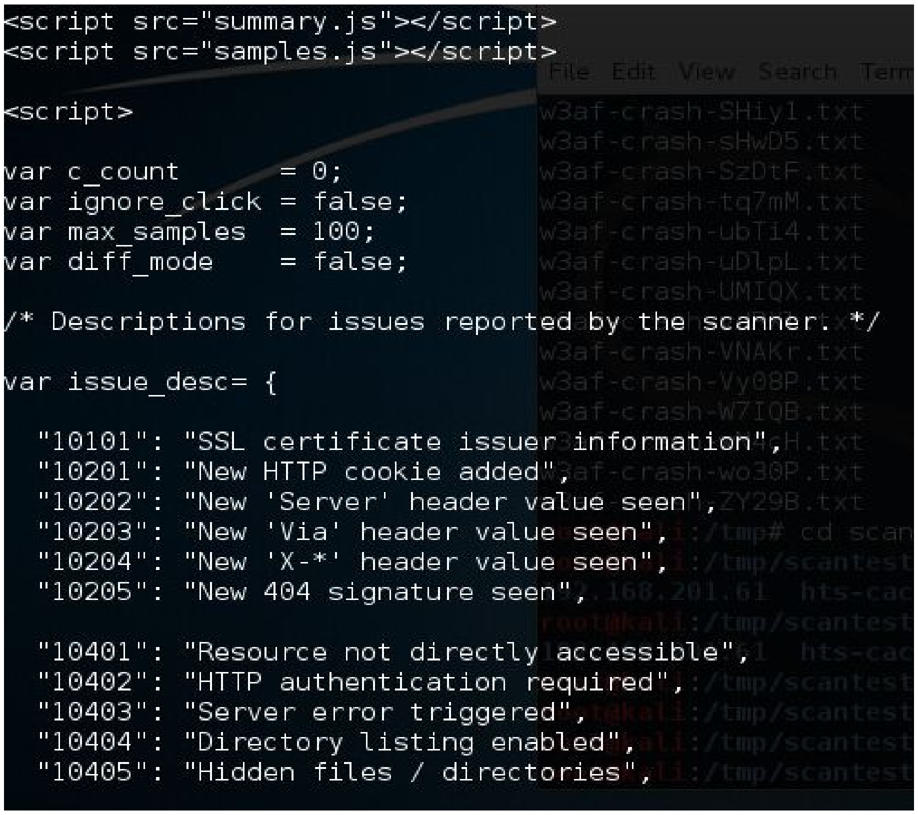
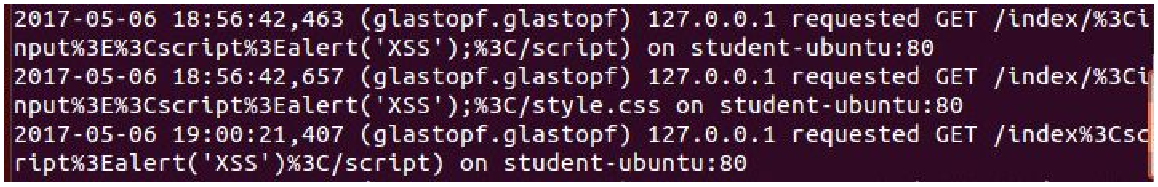

```
<script src="summary.js"></script>
<script src="samples.js"></script>

<script>

var c_count       = 0;
var ignore_click  = false;
var max_samples   = 100;
var diff_mode     = false;

/* Descriptions for issues reported by the scanner. */

var issue_desc= {

  "10101": "SSL certificate issuer information",
  "10201": "New HTTP cookie added",
  "10202": "New 'Server' header value seen",
  "10203": "New 'Via' header value seen",
  "10204": "New 'X-*' header value seen",
  "10205": "New 404 signature seen",

  "10401": "Resource not directly accessible",
  "10402": "HTTP authentication required",
  "10403": "Server error triggered",
  "10404": "Directory listing enabled",
  "10405": "Hidden files / directories",
```

Figure 2: Skipfish scan results

XSS Simulation:

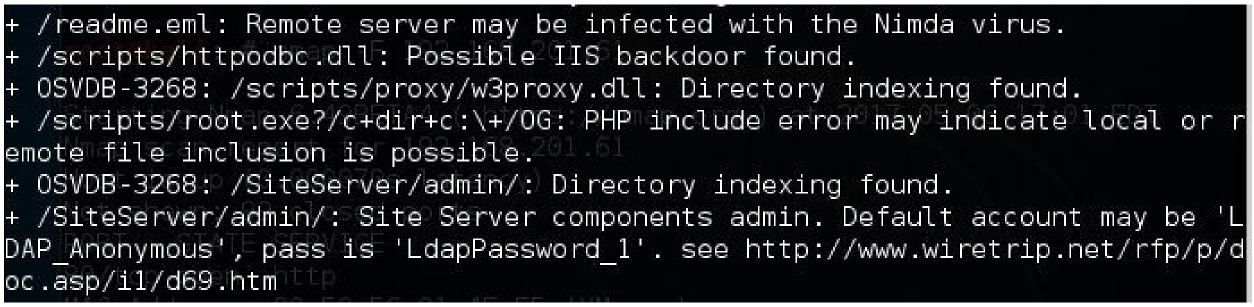
On performing the Manual Injection: <script>alert('XSS'')</script>



```
2017-05-06 18:56:42,463 (glastopf.glastopf) 127.0.0.1 requested GET /index/%3Ci
nput%3E%3Cscript%3Ealert('XSS');%3C/script) on student-ubuntu:80
2017-05-06 18:56:42,657 (glastopf.glastopf) 127.0.0.1 requested GET /index/%3Ci
nput%3E%3Cscript%3Ealert('XSS');%3C/style.css on student-ubuntu:80
2017-05-06 19:00:21,407 (glastopf.glastopf) 127.0.0.1 requested GET /index%3Csc
ript%3Ealert('XSS')%3C/script) on student-ubuntu:80
```

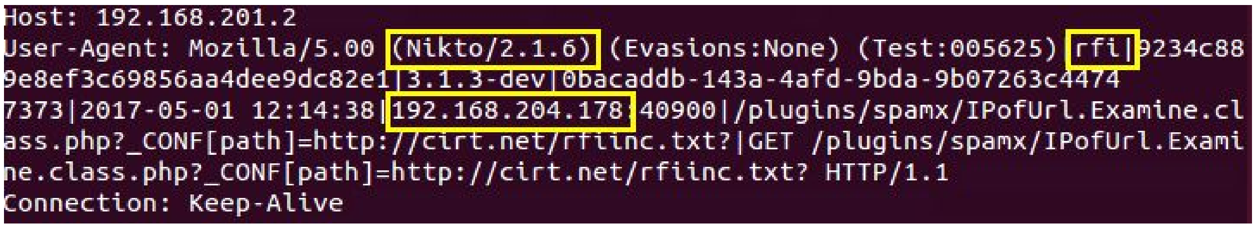
Figure 3: Glastopf log for XSS

RFI Simulation:



```
+ /readme.eml: Remote server may be infected with the Nimda virus.
+ /scripts/httpodbc.dll: Possible IIS backdoor found.
+ OSVDB-3268: /scripts/proxy/w3proxy.dll: Directory indexing found.
+ /scripts/root.exe?/c+dir+c:\+/OG: PHP include error may indicate local or r
emote file inclusion is possible.
+ OSVDB-3268: /SiteServer/admin/: Directory indexing found.
+ /SiteServer/admin/: Site Server components admin. Default account may be 'L
DAP_Anonymous', pass is 'LdapPassword_1'. see http://www.wiretrip.net/rfp/p/d
oc.asp/i1/d69.htm
```

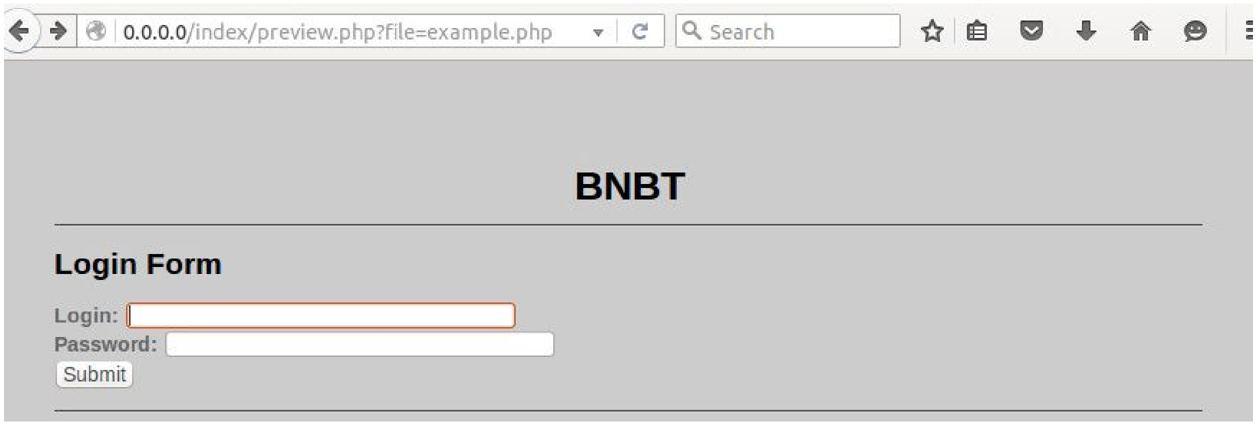
Figure 4: Nikto performing RFI.



```
Host: 192.168.201.2
User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:005625) rfi|9234c88
9e8ef3c69856aa4dee9dc82e1|3.1.3-dev|0bacaddb-143a-4afd-9bda-9b07263c4474
7373|2017-05-01 12:14:38|192.168.204.178:40900|/plugins/spamx/IPofUrl.Examine.cl
ass.php?_CONF[path]=http://cirt.net/rfiinc.txt?|GET /plugins/spamx/IPofUrl.Exami
ne.class.php?_CONF[path]=http://cirt.net/rfiinc.txt? HTTP/1.1
Connection: Keep-Alive
```

Figure 5: Glastopf log from database after Nikto scan.

LFI Simulation:

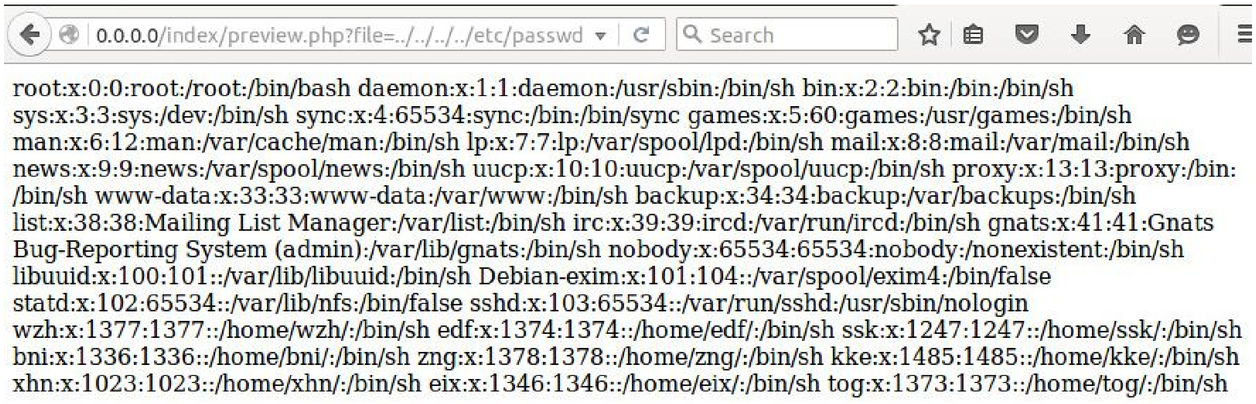


BNBT

Login Form

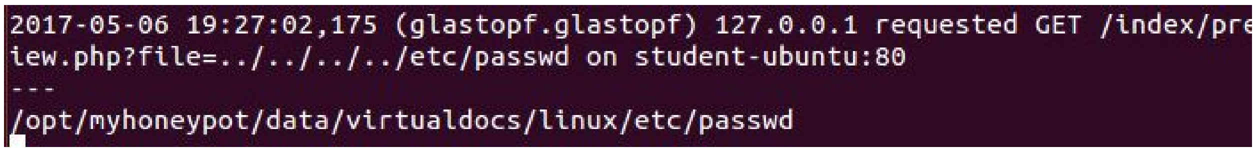
Login:
Password:
Submit

Figure 6: Glastopf homepage



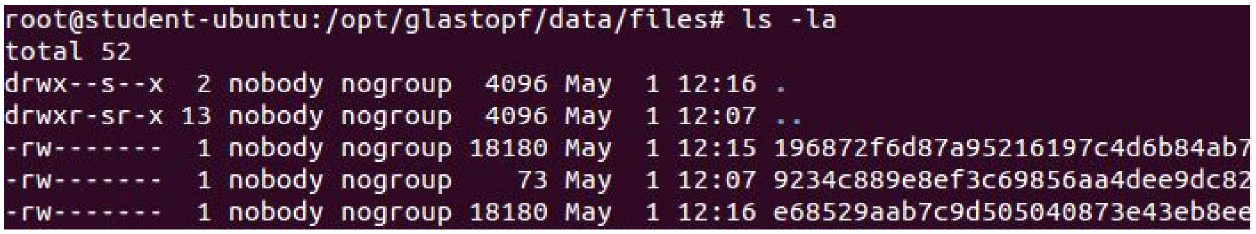
```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:
/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh Debian-exim:x:101:104::/var/spool/exim4:/bin/false
statd:x:102:65534::/var/lib/nfs:/bin/false sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
wzh:x:1377:1377::/home/wzh/:/bin/sh edf:x:1374:1374::/home/edf/:/bin/sh ssk:x:1247:1247::/home/ssk/:/bin/sh
bni:x:1336:1336::/home/bni/:/bin/sh zng:x:1378:1378::/home/zng/:/bin/sh kke:x:1485:1485::/home/kke/:/bin/sh
xhn:x:1023:1023::/home/xhn/:/bin/sh eix:x:1346:1346::/home/eix/:/bin/sh tog:x:1373:1373::/home/tog/:/bin/sh
```

Figure 7: Local File Inclusion Attack



```
2017-05-06 19:27:02,175 (glastopf.glastopf) 127.0.0.1 requested GET /index/pre
iew.php?file=../../../../etc/passwd on student-ubuntu:80
---
/opt/myhoneypot/data/virtualdocs/linux/etc/passwd
```

Figure 8: Glastopf log of LFI log.



```
root@student-ubuntu:/opt/glastopf/data/files# ls -la
total 52
drwx--s--x  2 nobody nogroup  4096 May  1 12:16 .
drwxr-sr-x 13 nobody nogroup  4096 May  1 12:07 ..
-rw-------  1 nobody nogroup 18180 May  1 12:15 196872f6d87a95216197c4d6b84ab7
-rw-------  1 nobody nogroup    73 May  1 12:07 9234c889e8ef3c69856aa4dee9dc82
-rw-------  1 nobody nogroup 18180 May  1 12:16 e68529aab7c9d505040873e43eb8ee
```

Figure 9: Shellcode stored.

By virtue of manual injections, all the cross site scripting attacks were captured successfully with the contents of the script visible. (Figure 3)

After successful exploitation with the help of some manual injections and Nikto, it can be seen that Glastopf's log records the attack pattern as 'RFI' and 'LFI' (Figure 5). These logs show additional information related to the attacker's activity such as time of event, tools used, pattern, HTML track, response code and also detailing the different parameters used during the attack. Additionally, the tool has the capability to store the shellcode in the database (Figure 9), which can be used for future analysis.

Phase 2:

In Phase 2, the honeypot was made available to the world for a period of 2 weeks. After this period, the logs were filled with data related to the plethora of attacks that were used against the honeypot.

The results suggested that the most common attacks performed against the system were XSS, RFI and LFI (Table 2). These attacks originated primarily from UK, Brazil, Denmark and other countries (Figure 10).

The most common HTTP methods used by attackers were GET, POST and HEAD; these were used to access files such as setup.py, robots.txt and /etc/passwd. (Table 4)

	Logged Events	**Total Events**
XSS Injections	110	118
RFI	7	9
LFI	16	22

Table 2: Logged Injections in Glastopf

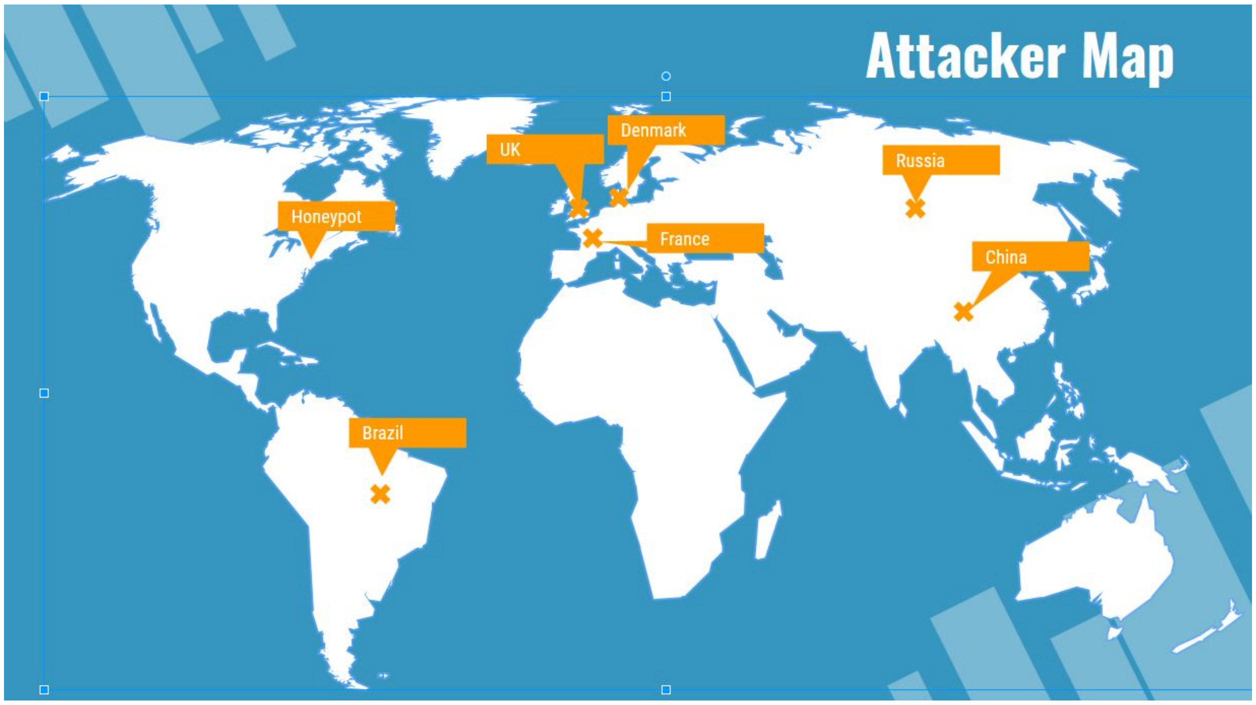


Figure 10: Attacker Map.

IP	Country
139.59.161.xx	UK
222.141.64.x	China
193.109.69.x	Russia
186.225.40.xxx	Brazil

Table 3: Attacker Country and IP

Methods	Status Code	Files
GET	200 (OK)	setup.py
POST	404 (Not Found)	robots.txt
HEAD	400 (Bad Request)	/etc/passwd

Table 4: Commonly used techniques by attackers

Glastopf Strengths:

- It is easy to understand the logs that are captured.
- Logs capture the IP address of the attacker along with the time of attack.
- Logs also capture the HTML information such as the response code (200 OK).

Glastopf Demerits:

- The installed version of Glastopf failed to capture SQL injections.
- Captures the Cross site scripting with pattern name as 'comments' instead of XSS.

V. FUTURE WORK

One major drawback of the tool is that it lacks the ability to gather specific information about the attacker's identity. Hence, for gathering the attacker's information, the honeypot should have the additional feature of using 'LikeJacking'; the purpose of this technique is to get people to click items on a webpage without their knowledge. [8]

Typically, used as a spam mechanism, attackers present a web page that has two layers. The back layer is designed with a Facebook "Like" button configured to follow the cursor. The front end showcases the lure to trick the user. No matter where you click on the webpage, you are actually clicking the Facebook Like button and further spreading the spam. [6]

However, this technique could be used as a defensive utility, wherein, a Facebook page will be accidently 'liked' by the attack if they visit the honeypot. The reason this attack works is that Facebook does not require any confirmation when you click the Like button. Though confirmation would not entirely prevent the attack but would only complicate the attack. The only problem is that this action only takes place in the event that the attacker is logged on to his Facebook account while visiting honeypot. But even then if the attacker removes the 'Like', the Notification pane can be used to gather information.

VI. CONCLUSION

It is critical to first understand and know the enemy in order to defeat him. Thus, in conclusion, honeypots can prove to be a very good starting point in order to protect web applications from malicious users. They learn the exploitation methods deployed by an attacker and report them back to the experts for further analysis.

In this case, Glastopf unlike many other honeypots, focuses on replying the correct response to the attacker and not the specific vulnerability. It uses state of the art mechanisms to capture nefarious activities and then extract information pertaining to those suspicious attackers. This information can help experts unearth attackers that could prove to be dangerous to their cause in the future.

VII. RECOMMENDATIONS

It is better to refuse access to intruders than trying to fool them as a honeypot system may be set up by Blackhats to lure regular users to a fake system in order to gain sensitive information such as account number and credit card numbers. Furthermore, there are certain social aspects and legal issues associated with the use of honeypots. Hence, a decoy system should be deployed with great care, consideration and pre-evaluation of all aspects. [4]

VIII. REFERENCES

[1] A. Ghourabi, T. Abbes and A. Bouhoula, "Design and implementation of Web service honeypot," *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, Split, 2011, pp. 1-5.

http://ieeexplore.ieee.org.ezproxy.rit.edu/stamp/stamp.jsp?tp=&arnumber=6064371&isnumber=6064354

[2] A. Zarras, "The art of false alarms in the game of deception: Leveraging fake honeypots for enhanced security," *2014 International Carnahan Conference on Security Technology (ICCST)*, Rome, 2014, pp. 1-6.
doi: 10.1109/CCST.2014.6987017

http://ieeexplore.ieee.org.ezproxy.rit.edu/stamp/stamp.jsp?tp=&arnumber=6987017&isnumber=6986962

[3] B. Mphago, O. Bagwasi, B. Phofuetsile, and H. Hlomani, " Deception in Dynamic Web Application Honeypots: Case of Glastopf", College of Information Communication and Technology Botswana International University of Science and Technology Palapye, Botswana.

http://worldcomp-proceedings.com/proc/p2015/SAM9766.pdf

[4] C. Rong and Geng Yang, "Honeypots in blackhat mode and its implications [computer security]," *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 185-188.
doi: 10.1109/PDCAT.2003.1236284

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1236284&isnumber=27719

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

[6] S. Djanali, F. Arunanto, B. A. Pratomo, A. Baihaqi, H. Studiawan and A. M. Shiddiqi, "Aggressive web application honeypot for exposing attacker's identity," *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, Semarang, 2014, pp. 212-216.
doi: 10.1109/ICITACEE.2014.7065744

http://ieeexplore.ieee.org.ezproxy.rit.edu/stamp/stamp.jsp?tp=&arnumber=7065744&isnumber=7065431

[7] Tan, Emil. "Glastopf- A Web-application Honeypot." *Edgis*. N.p., 22 Apr. 2014. Web. 12 May 2017.

[8] Wisniewski, Chester. "What Is "Likejacking"?" *What Is Likejacking*. Sophos, n.d. Web. 12 May 2017.

[9] Y. Alosefer and O. Rana, "Honeyware: A Web-Based Low Interaction Client Honeypot," *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, Paris, 2010, pp. 410-417.
doi: 10.1109/ICSTW.2010.41

http://ieeexplore.ieee.org.ezproxy.rit.edu/stamp/stamp.jsp?tp=&arnumber=5463678&isnumber=5463636

[10] https://github.com/mushorg/glastopf