Honeyware: a web-based low interaction client honeypot

Yaser Alosefer, Omer Rana
School of Computer Science & Informatics,
Cardiff University,
UK
{Y.Alosefer, O.F.Rana}@cs.cf.ac.uk

***Abstract*—Modern attacks are being made against client side applications, such as web browsers, which most users use to surf and communicate on the internet. Client honeypots visit and interact with suspect web sites in order to detect and collect information about malware to protect users from malicious websites or to allow security professionals to investigate malicious content. This paper will present the idea of using web-based technology and integrating it with a client honeypot by building a low interaction client honeypot tool called Honeyware. It describes the benefits of Honeyware as well as the challenges of a low interaction client honeypot and provides some ideas for how these challenges could be overcome.**

Keywords: Honeyware, client honeypot, honeypot, malicious web-based.

I. INTRODUCTION

A honeypot is a security technology that provides organisations with a way to catch viruses, malware or attackers, as well as acting as an alarm system that can discover attempts to attack a network. Honeypot technology is defined as a 'security resource whose value lies in being probed, attacked or compromised' [1]. There are two main types of honeypot: passive and active. The passive honeypot is a technology that passively waits for attacks in order to detect them, while the active honeypot, also called a client honeypot, interacts with a target web page to identify and determine its potential effect on the browser or operating system.

The advantages of using honeypots are:

- They collect and log data with small amounts of false positives and negatives as it logged data only from the target web page.
- The data has a high value.

The disadvantages of using honeypots are:

- The risk after a system is compromised, which could be an attack on another system on the same network or outside.
- Highly skilled people and their time are needed to operate and analyse the data from honeypots.

The client honeypot architecture is generally divided into three main parts, as shown in Figure 1. The client honeypot also has two varieties: high and low interaction. A high interaction client honeypot is used in a real operating system with real browsers, while a low interaction client honeypot simulates a web browser to interact with the target server, and then scans the server's response to determine whether or not there is malicious code within the response. This paper presents a web-based low interaction client honeypot called Honeyware. The tool interacts with a given URL by simulating a specific web browser through its user-agent string [16] [17] and then downloading the response of the target web site. This response is then examined using the scan engine provided by Honeyware which can detect known malicious signatures. This paper also attempts to overcome some of challenges inherent to client honeypots.

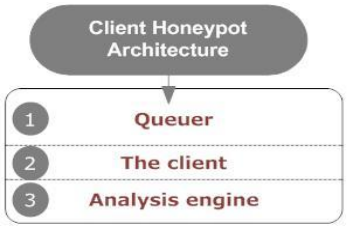

Figure 1 the client honeypot architecture.

II. THE CLIENT HONEYPOT

Attackers are now targeting client applications such as web browsers and media players, whereas they were previously primarily targeting servers. Therefore, there is a requirement to determine their modes of attack and to capture their malicious scripts and tools to analyse and improve security. Instead of passively waiting for attackers, an active honeypot will go and search for the attackers. The client honeypot interacts with the server to study it and determine if an attack has happened. It can interact with websites using HTTP as well as other protocols such as FTP.

A. *Client honeypot architecture*

The client honeypot architecture is divided into three main sections: the queuer, the client and the analysis engine (see figure 1). Each part is necessary to the client honeypot.

1) *The queuer*: the queuer is responsible for creating a server list for the client to interact with. There are several techniques used to create URL lists, including search engines, Blacklists, Phishing and spam messages, Typosquatting domains and Instant messaging.
2) *The client*: the client is the component that makes requests and interacts with the servers collected by the queuer.
3) *The analysis engine*: the analysis engine is responsible for determining and checking the state of the client honeypot to see if an attack has occurred or not.

III. RELATED WORK

A low interaction client honeypot uses a lightweight simulation to simulate a client when interacting with a server. The aim of this is to detect malicious code present on a server by interacting with the server and examining its responses to determine if there is any malicious code present. The reply is examined by checking the string for the page, known as the malicious code string. One disadvantage of this is that it cannot detect unknown malware because it uses a signature database to match them with the responses from the server. However, the low interaction client honeypot also has some advantages such as its ease of use and deployment and its speed in reporting results as compared to the high interaction honeypot.

There are a number of low interaction client honeypots have been developed, including:

1) *HoneyC*: developed by Christian Seifert [18], this open-source tool uses the three previously mentioned components to create a list of websites by performing a search via the Yahoo! Search engine. Then, it visits each website identified and downloads them to, finally, examine the web page code via Snort, an IDS tool that contains a signature database of malicious strings.
2) *SpyBye*: written by Niels Prvos [14], this works like HoneyC but uses the ClamAV anti-virus engine to check web pages instead of Snort.
3) *Monkey-Spider*: developed by Ali Ikinci [15] at the University of Mannheim, this tool is free to use under the GPL license.

IV. CHALLENGES OF LOW INTERACTION CLIENT HONEYPOTS

Honeypots help security researchers to study the techniques and objectives of web-based malware. However, there are still some challenges to be overcome to allow their full benefits to be achieved:

- *IP tracking*: this technology is widely used in web-based tools to track the IP address of visitors. For example, according to Seifert [3] the Mpack [12] tool tracks visitors' IP addresses and only attacks the visitor with malicious scripts after a number of visits to the website. If a client honeypot tries to visit a malicious website running the Mpack tool with the IP tracking feature enabled, it will not detect any malicious behaviour and may assume the site is clean, but if the client honeypot visited it a number of times, the IP tracking feature will trigger the malicious behaviour. Therefore, the challenge is to solve this problem to prevent false negatives.
- *Geolocation-dependent*: this feature, provided by a number of malware tools, will cause the malware only to affect visitors from certain countries, while behaving normally with visitors from other countries.

V. HONEYWARE

Honeyware is a new low interaction client honeypot tool which aims to combine the benefits of web-based tools that run on local or remote servers with the ability to access the tool from a web browser, which is important for many different devices such as PCs or mobiles. Another important function of this tool is that it gives the user the ability to test the target server with almost all the available web browsers and to scan the target with five different scan engines. In the future it will also scan the target with an intrusion detection system such as Snort [13].

Honeyware is written in PHP and is an open source tool. It was developed to solve some problems and challenges in low interaction client honeypots. Its features are:

1) *Web browsers*: Honeyware has the ability to simulate different web browsers including: Internet Explorer, Firefox, Opera, Chrome, Safari and Konqueror.
2) *The Scan engine*: Honeyware has the ability to scan target files with different scan engines to determine whether the target has malicious code or not. Some scan engines are able to detect viruses and malware

that another product cannot. The scan engines that Honeyware supports are:

- **AVG** version 7
- **F-Prot** version 5.2.1.4252
- **Avast** version 1.3.0
- **ClamAV** version 0.95.2
- **AntiVir** version 7.8.0.0

An example of the benefits of using different scan engines is the exploit with the CVE number CVE-2007-5601. The vulnerability is in the RealPlayer software; according to the CVE [4].This vulnerability was scanned with the five different scan engines and only two out of the five scan engines (AVG and Avast) detected it.

3) *Crawling*: Honeyware supports search engine crawling to obtain URLs to scan. Honeyware supports two main search engines: Yahoo! and MSN.
4) *Honeyware Client*: Honeyware also implements an important client tool that can interact with the target server to retune the files downloaded to Honeyware and examine them. This feature is very important for solving geolocation-dependent problems.

A. *Honeyware overcoming client honeypot challenges*

Honeyware was built to solve some of the challenges inherent in a low interaction client honeypot, as mentioned in section 4, in order to implement a useful tool. Attackers take advantage of these problems to prevent their detection by low interaction client honeypot tools, and so deliver malicious binaries to a wide range of people. There are already some tools available for low interaction client honeypots, but it is hoped that Honeyware will improve upon these.

The first challenge is the use of IP tracking to delay malicious code's activity according to the number of website visits. There are two possible ways to solve this:

1) Visit the website multiple times to give the malicious web page an appearance of usual human visitor behaviour. This way, the honeypot will need to download the target files and delete them in a loop. This method is not reliable as it requires more time to do the loop, and causes more traffic.
2) Visit the target web page as a spider which connects to the target web page and makes a HTTP request to the target but does not save the target response files to the Honeypot machine. This would be much faster than the first method and it could also send multiple web spider requests simultaneously.

However, with both methods the malicious web page will detect the multiple requests to the web page and could block the honeypot or behave as benign. One idea to solve this is to use a client to make the multiple requests and then to use the Honeyware tool to scan the target machine. In this scenario, Honeyware would be able to compare both results and check if the web page used any techniques to mask its malicious behaviour. This scan would take some time to complete as it needs to request the web page twice, once through Honeyware and once through the client. The client would need to request the web page multiple times using a Wget tool in spider mode, using the web spider method of interaction with the target and not saving any response to the local machine. Figure 2 shows the IP tracking scenario using Honeyware. It is essential, whichever method is used, always to request the malicious web page from the same IP address so that the IP address will be tracked.

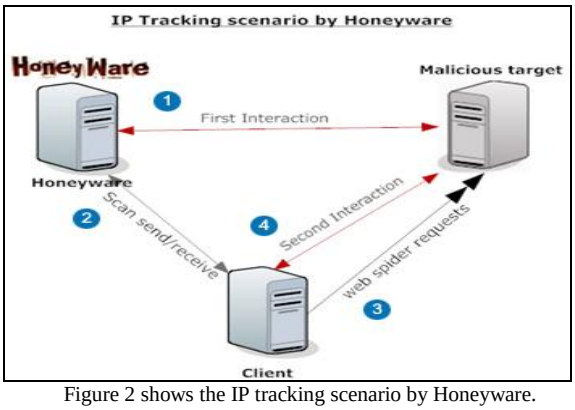

Figure 2 shows the IP tracking scenario by Honeyware.

The second challenge is the use of geolocation to only attack visitors from certain countries. The Swiss security blog [5] conducted an experiment using a honeypot to find sites that collected FTP credentials from different zombies by attacking them first from a malicious website and then exploiting a vulnerability on the victim's PC to capture his FTP credentials and send them to a control server. The results showed the top country attacked was the USA, the Honeyware operator could therefore install a client in the USA, even if the main Honeyware server is elsewhere.

Using this technique, Honeyware can successfully detect websites that use geolocation-dependent features provided by tools such as Mpack, by placing Honeyware clients in locations that appear to be of high value to the attackers.

B. *Honeyware architecture*

Honeyware is able to scan any target server with a valid URL. The user can choose the web browser that Honeyware will simulate. The benefit of this is that some attackers use web-based tools such as Mpack [12] to attack only visitors using a certain browser, or even a

certain version of a browser. For example, Internet Explorer version 6 suffers from a remote code extension vulnerability (CVE-2006-5579 and MS06-072) [20][21].The attacker can check the visitors' web browser version and then only infect them if they use the same vulnerable web browser version.

To simulate a web browser by Honeyware, each web browser uses a specific user agent which is a string sent first to the target server in order to visit it. The user agent includes important information such as the web browser name, version and name of the host operating system. The benefits of a user agent are important in different situations to different people. For example, using the user agent to detect the browser and OS of visitor's operating system and browser and then redirects the user to a specific exploit that works specifically against the victim's system which is done by the Mpack web-based exploit tool [12]. Figure 3 shows Mpack's attack method using visitor browser product and version.

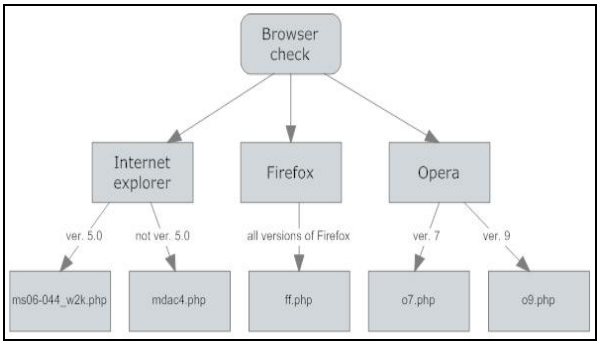

Figure 3 Mpack attack methods

The attacker can use this to their advantage to feed each browser with a vulnerability; web browser detection can be easily done by using JavaScript. [10] Figure 4 shows the Honeyware architecture.

Honeyware is a web-based tool built using HTML and CSS; it provides many features, such as the ability to save all scan and crawling results into a database to create archives; it can also delete scan or crawling results. The main page, as shown in Figure 5, has two main parts; the scan form and archives. The user can initiate a scan easily using the form and can change the default settings, such as the web browser or scan engine. The page also displays the scan archive, giving the name of the scan and some information such as the web browsers and scan engines used.

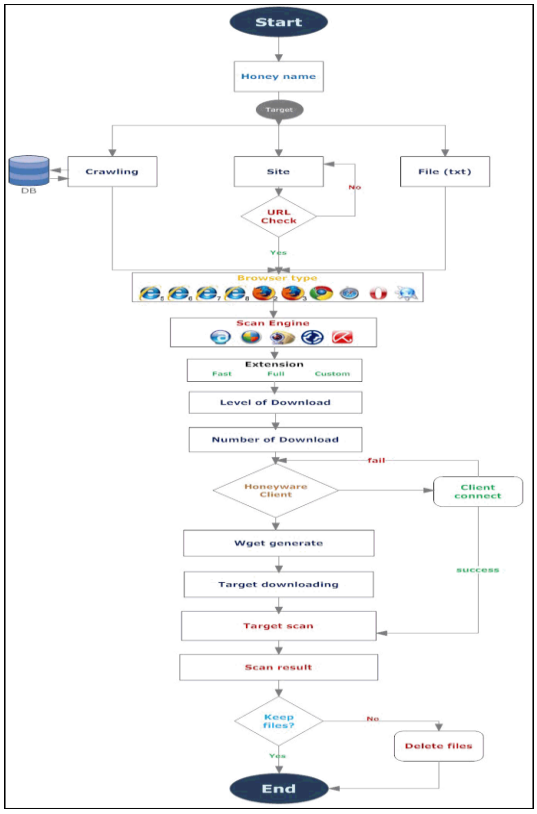

Figure 4 Honeyware architecture.

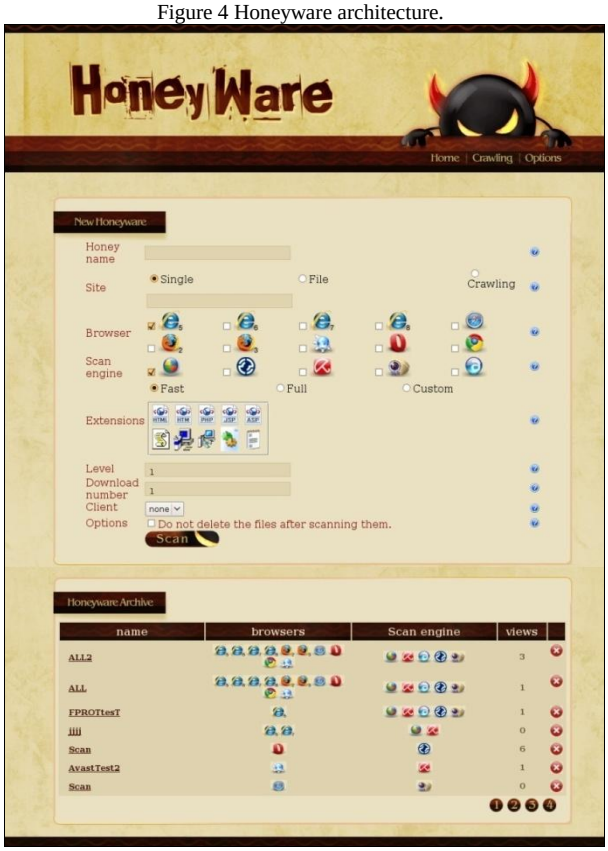

Figure 5 the Honeyware main page.

Figure 6 shows an example of a result page for a scan which was performed on a known exploited page on a local web server in order to test the scan engines and the Honeyware tool for their ability to detect the exploit. The result page also shows the target files downloaded by Honeyware allowing them to be checked manually to get a better understanding of the target server.

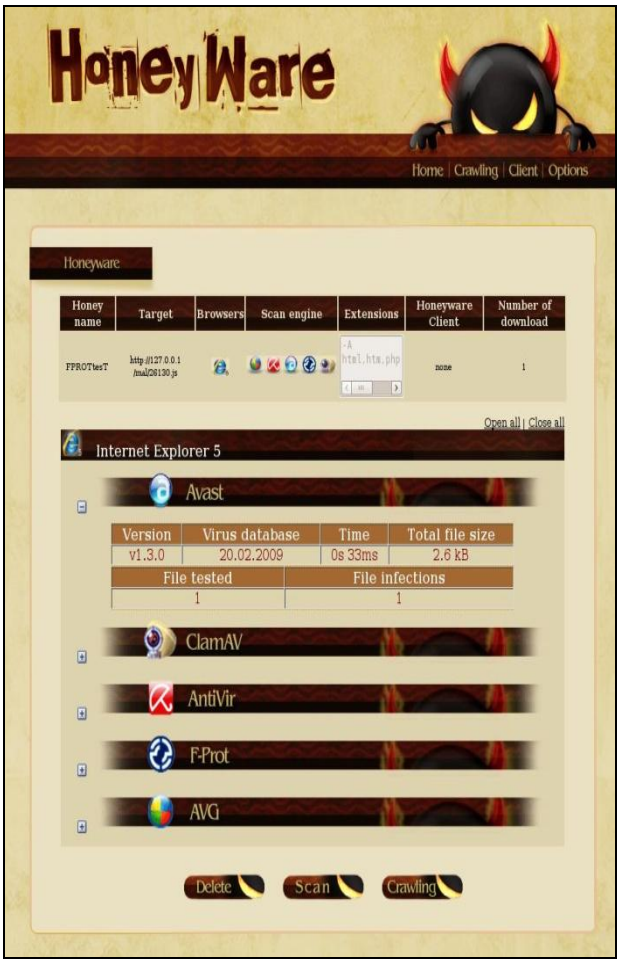

Figure 6 the Honeyware result page.

VI. THE HONEYWARE CLIENT

Honeyware provides a new and powerful feature for low interaction client honeypots. As previously discussed, the Mpack tool allows visitors from specific countries to be targeted [3]. To overcome this, the Honeyware client, which can be set up in different locations, was created. The structure of the Honeyware client is simple; it is a single PHP file and an empty directory. The Honeyware client supports some of the features that can be used from the Honeyware tool:

1) *Connection encryption.*
2) *Connection state*: Honeyware can be queried for the state of the client; this is done to add new clients to Honeyware.
3) *Target interaction*: The main purpose of the client is to connect to a target server so the server will interact as the client's location.

The Honeyware client will interact with Honeyware via the web server, port 80, and this port has to be open so the connection can be established between Honeyware and its clients. However, the target server can easily scan the visitor's port address. Because the normal end-user does not need to open port 80, if the port is open then the malicious server may assume it to be a honeypot and so interact safely with the visitor. There is an easy solution for this which has to be implemented while installing the Honeyware client; it uses iptables, a free and open source tool that is shipped with most major Linux distributions. iptables can be configured to only allow connections from the IP address of the Honeyware server and to reject any other attempts to connect, thus preventing the attacker scanning port 80 to see if the Honeyware client exists. Figure 7 shows the communication between Honeyware and its clients.

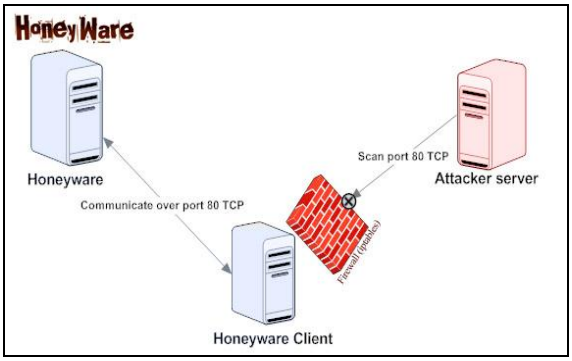

Figure 7 communication between Honeyware and its clients.

VII. HONEYWARE EXPERIMENT

To validate Honeyware, an experiment was conducted using Honeyware as a low interaction client honeypot to scan a collection of malicious and benign URLs to see its the effectiveness of Honeyware compared to Capture-HPC [12], a high interaction client honeypot.

The experiment scenario involved 94 URLs collected from a search engine of which 84 were malicious and 10 benign. As previously mentioned, Honeyware uses anti-virus software to scan a URL after it has been downloaded. A URL will be considered by Honeyware if one or more of the antivirus programs

detect any malicious code or files in the downloaded URL. Figures 8 and 9 show the experiment results.

Honeyware detected 83 of the 84 malicious URLs and failed to detect 1 URL on 4/11/2009. After the antivirus software was updated, the test was conducted again on 8/11/2009 and it detected the URL which it had failed to detect the first time. Of the benign URLs, it did not detect any malicious files. On the other hand, Capture-HPC scanned the same 94 URLs but was only able to detect 62 of the malicious URLs, it classed 23 URLs as benign and gave an error for 9 URLs.

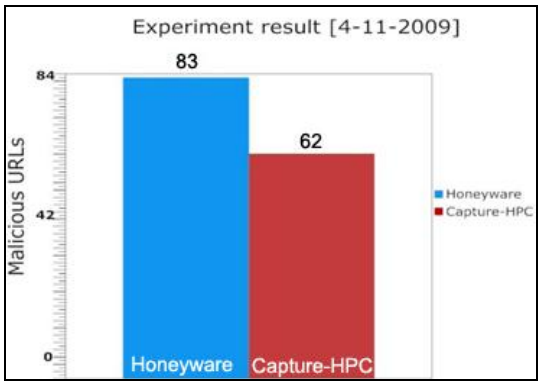

Figure 8 experiment results on 4th November.

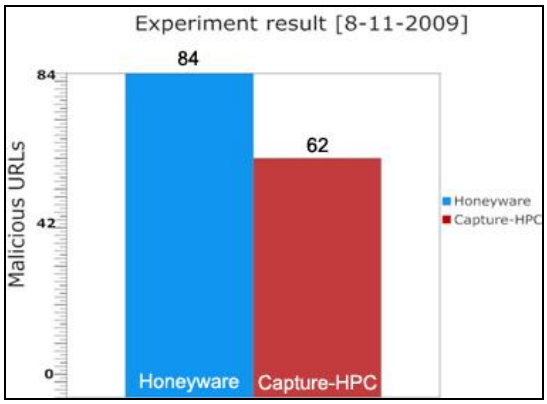

Figure 9 experiment results on 8th November.

Honeyware is only able to detect known attacks as it uses antivirus software as the scan engine and this needs to be updated regularly to get valid results, as was shown by this experiment when the Avast antivirus software detected the new URL which was not previously detected on the 4th November. On the other hand, Capture-HPC failed to detect 22 malicious and 9 error URLs. We believe that Capture-HPC is only able to detect URLs which are triggered just by visiting a web page, in the way most drive-by download attacks work, whereas some of the more experimental URLs needed some interaction from the user to trigger their malicious code to ensure the user was human and not a client honeypot. Other web pages try to fool the user into downloading the malicious application by pretending to be a useful benign application. Capture-HPC can only detect the behaviour of a web page scanned without any interaction, while Honeyware is able to download all the files and links that point to other sources so it can download html pages, exe applications and other extensions that can be customised by the analyst. Figure 10 shows how well each antivirus program performed.

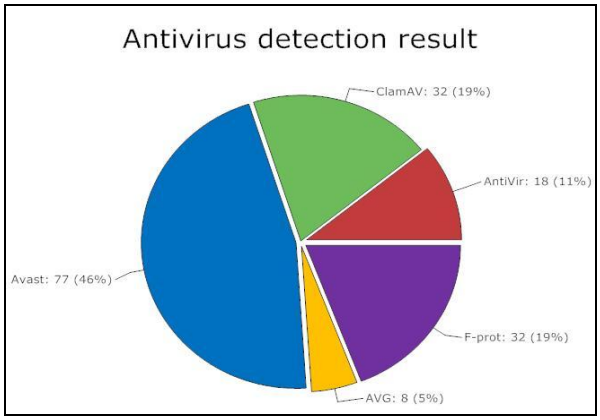

Figure 10 antivirus detection results.

This experiment did not test the Honeyware client as it is hard to find live malicious websites which use geolocation-dependent or IP tracking tools such as Mpack because most known publicly available malicious URLs are down and the few of them which are live use traditional malicious attacks. However, an internal experiment was conducted using Mpack and other location and IP tracking scripts and this found that the Honeyware client works successfully to detect these types of malicious websites. In the case of IP tracking, Honeyware sends a request to the target and then the client sends web spider requests to increase its IP tracking, meaning Honeyware can compare both responses to detect any differences.

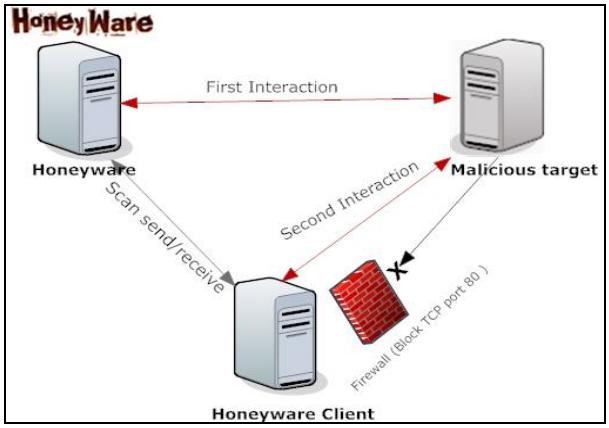

Figure 11 the Honeyware scan scenario.

VIII. HYBRID SYSTEM

Honeyware is a low interaction client honeypot, which means it can detect known malware and it can simulate operating systems or services. While a high interaction client honeypot like Capture-HPC has many of the same features, Honeyware is able to detect unknown attacks but it takes a long time to complete each scan. Capture-HPC takes about 17 seconds to scan each URL whereas Honeyware takes an average of one minute because it downloads the URL first and scans it using its scan engine.

The experiment showed that Honeyware was able to detect all the malicious URLs, but it takes a long time to finish scanning all the URLs and to get the results back to the investigator, but Capture-HPC scanned all 94 URLs in about 30 minutes. The idea of the hybrid system is that it combine the high and low interaction client honeypots. The benefit of this hybrid system is that Capture-HPC is fast enough to scan URLs and is able to detect unknown malicious websites, while Honeyware can detect URLs that need some interaction from an end-user, which cannot be detected by Capture-HPC.

The hybrid system starts by scanning all URLs with Capture-HPC and then forwards all benign URLs from Capture-HPC to Honeyware to scan. This hybrid system would be fast and would detect all our experimental URLs, as the Capture-HPC will detect the 62 URLs quickly and then forward the benign and error URLs, of which there were 32, to Honeyware which would be able to detect the remainder of the malicious URLs. Figure 12 shows the idea behind the hybrid system.

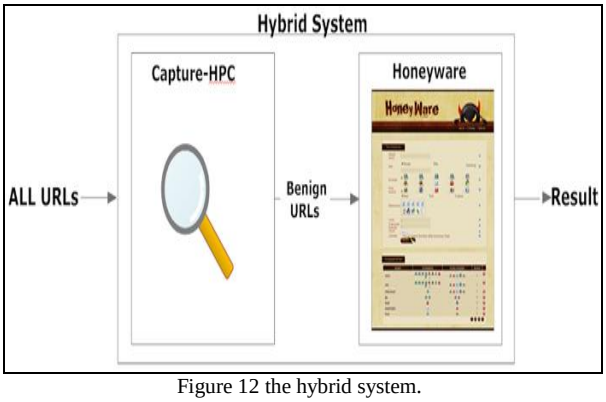

Figure 12 the hybrid system.

IX. HONEYWARE FUTURE WORK

Honeyware was developed as a web-based low interaction client honeypot which integrate the web technology to client honeypot. Therefore, there are some future works which will be implementing in future versions of Honeyware. The following points are the future for Honeyware:

1. *Plug-in simulation*: Some malicious websites will exploit vulnerability within plug-ins installed on the web browser, such as Flash Player, RealPlayer and Adobe Reader. There are some exploits available for such plug-ins and therefore the attacker can check the visitor plug-in through some plug-in detectors [7][8][9] and exploit the user if the valuable plug-in is present. An example of plug-in vulnerabilities is CVE-2009-0376 and CVE-2009-0375; these vulnerabilities affect RealPlayer version 11, allowing remote execution of an arbitrary code for an attacker to successfully exploit this vulnerability. However, if the attacker fails to exploit the previous vulnerability, then it will cause a denial-of-service condition.
2. *Intrusion detection system (IDS):* Honeyware uses scan engines to scan the target server. In addition, the use of such IDS as Snort [23] will add an excellent feature to Honeyware.
3. *Honeyware Crawling*: The current version of Honeyware supports two of the main search engines, Yahoo! and MSN, and it will be helpful if other search engines were added to Honeyware to give the user the ability to search across different search engines. Search engines such as Google and Ask are examples that can be added to Honeyware.
4. *Improve Honeyware client.*

X. CONCLUSION

Honeyware applies web-based technology to the low interaction client honeypot. In a hybrid system, the idea is to pass all URLs to a high interaction client honeypot and then to pass URLs found to be benign to Honeyware to validate the results and to provide another layer of detection. This technique will, in theory, be quite slow, but further investigations will be conducted into the hybrid system and its effectiveness in the real world as it will achieve more accurate results at a cost to the detection speed.

Honeyware could also use the idea of a state machine as a state machine map could be built for each URL scanned and then each map could be compared to others to categorize them for further investigation or even to pass certain groups to a high interaction client honeypot in the hybrid system. Honeyware can be downloaded from the SourceForge website at http://www.sourceforge.net/projects/honeyware.

XI. REFERENCES

[1] L. Spitzner, (2002). Honeypots: Tracking Hackers. 1st edition. Addison-Wesley Professiona. ISBN-10: 0321108957.

[2] N. Provos, T. Holz, (2007). Virtual Honeypots: From Botnet Tracking to Intrusion Detection. 1st edition. Addison-Wesley Profession. ISBN-10: 0321336321.

[3] C. Seifert, (2007). Know Your Enemy: Behind the Scenes of Malicious Web Servers. [Online]. Available at: http://honeynet.org/book/export/html/181 [Accessed 1st Mar 2009].

[4] CVE-2007-5601 [Online]. Available at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5601 [Accessed 4th Mar 2009].

[5] The Swiss Security Blog [Online]. Available at: http://www.abuse.ch/?p=737 [Accessed 5th March 2009].

[6] What's a User Agent? [Online] Available at: http://whatsmyuseragent.com/WhatsAUserAgent.asp [Accessed 5th Mar 2009].

[7] Plug-in detection [Online]. Available at: http://developer.apple.com/internet/webcontent/detectplugins.html [Accessed 3rd Mar 2009].

[8] Plugin detection [Online]. Available at: http://www.smeenk.com/article.php?ArticleID=2 [Accessed 5th March 2009].

[9] Browser Plugin Detection with PluginDetect [Online]. Available at: http://www.pinlady.net/PluginDetect/ [Accessed 4th Mar 2009].

[10] Detect browser and display [Online]. Available at: http://drupal.org/node/65903 [Accessed 3rd Mar 2009].

[11] mcrypt_encrypt [Online]. Available at: http://php.net/manual/en/function.mcrypt-encrypt.php [Accessed 1st Mar 2009].

[12] Mpack web-based exploit tool [Online]. Available at: http://blogs.pandasoftware.com/blogs/images/PandaLabs/2007/05/11/MPack.pdf

[13] Snort [Online]. Available at: http://www.snort.org/

[14] SpyBye [Online]. Available at: http://www.spybye.org

[15] Monkey-Spider [Online]. Available at: http://pi1.informatik.uni-mannheim.de/filepool/publications/monkey-spider.pdf

[16] MSDN Library, Understanding User-Agent Strings, [Online]. Available at: http://msdn.microsoft.com/en-us/library/ms537503(VS.85).aspx

[17] Analyze UA tool, [Online]. Available at: http://user-agent-string.info/

[18] C. Seifert, I. Welch, P. Komisarczuk, (2006). HoneyC - The Low-Interaction Client Honeypot [Online]. Available at: http://homepages.ecs.vuw.ac.nz/~cseifert/blog/images/seifert-honeyc.pdf [Accessed 1st Mar 2009].

[19] Free Software Foundation, Inc.,Wget [Online]. Available at: http://www.gnu.org/software/wget/ [Accessed 7th Mar 2009].

[20] CVE-2006-5579 [Online]. Available at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5579 [Accessed 4th Mar 2009].

[21] Microsoft Security Bulletin MS06-072 [Online]. Available at: http://www.microsoft.com/technet/security/Bulletin/ms06-072.mspx [Accessed 4th Mar 2009].

[22] Y. Alosefer, O.F. Rana, "Automated state machines applied in client honeypots," unpublished.

[23] Snort [Online]. Available at: http://www.snort.org/ [Accessed 3rd Mar 2010].