2011 International Conference on Network and Electronics Engineering
IPCSIT vol.11 (2011) © (2011) IACSIT Press, Singapore

Taxonomy of Hybrid Honeypots

Hamid Mohammadzadeh.e.n [1], Masood Mansoori [2] and Roza Honarbakhsh [3]

[1] Faculty of computer Science & Technology, University of Malaya, Kuala Lumpur
[2] Faculty of Computer Science and Software Engineering, University of Canterbury, New Zealand
[3] Advanced Informatics School, University Technology Malaysia, Kuala Lumpur
Email: [1] h.network@gmail.com, [2] masood.mansoori@pg.canterbury.ac.nz, [3] roza.honarbakhsh@gmail.com

Abstract. End users of computer networks are vulnerable to a growing number of threats posed by sophisticated online attack. The first step in mitigation and protection against online attacks is to understand the nature and tools of the attacks. Honeypots provide a platform by which online attacks can be investigated. They are versatile security tools designed on the principle of deception and deployed to perform a wide range of functionalities. Their functionalities range from data collection and information gathering on attacks targeting server side services or client applications and browsers, to malware collection, attack diversion and SPAM analysis. Based on the type of attack, honeypots with different architectures and interaction levels have been developed. Different components of honeypots have also been integrated to create hybrid honeypots to tackle drawbacks of each type. The aim of this paper is to illustrate the different designs of honeypots, their components and provide a brief study of their security features.

Keywords: Server Honeypot, Client Honeypot, Low Interaction, High Interaction, Hybrid Honeypot.

1. Introduction

The online world is a highly interconnected network of organizations, educational institutions, private LANs, home users and mobile devices. A large number of new users are connecting to the Internet to access online resources and communicate with the rest of the world. Maintaining security of this interconnected network is costly and requires constant monitoring. The security of most networks is manages by utilizing Firewalls, Intrusion Detection Systems (IDS) and Honeypots. Firewalls are the first measure of protection and highly effective in managing access in and out of a network. Intrusion detection systems are alternatively able to monitor connections, inspect packets and match them against database of known attacks or examine them for anomalies. Intrusion detection systems were once thought as the "one answer for all" security tool.

There are inherent design issues in both firewall and intrusion detection system. They produce a large set of logs that need to be analysed and require constant monitoring and administrative attendance. Anomaly based intrusion detection systems are also prone to false positive and false negatives which cause denial of service and attack detection failure respectively. Development of intrusion detection systems faces new challenges by introduction of high speed networks. Tremendous increase in the speed of the data communication and the growth of the networks requires significant processing powers to allow intrusion detection systems to perform Deep Packet Inspection (DPI). The biggest challenge facing intrusion detection system however comes from utilization of IPv6 protocol for online communication. IPv6 utilises IPSec as encryption mechanism which allows hosts to encrypt packets before transmission thus making deploying IDS impractical, as they rely on packet content inspection to detect threats. Honeypots were developed to solve these issues.

2. Background

Honeypots have various security applications and are flexible security tool, capable of detecting variety of attacks. For instance, they are capable of detecting encrypted attacks, diverting attention away from production hosts and servers and averting adversaries away to dummy systems which may or may not exist. In fact, the value of honeypots is in providing early warning about new attacks, and allowing in-depth examination of attackers during and after exploitation.

2.1. Intrusion Deception System and Honeypots

Deception is a novel defences strategy in modern network security. The idea behind deception systems is to provide information or services that deceive the intruder to perform desired actions. Honeypots operate on the principle of deception and therefore can be categorised as deception systems. A honeypot in security community refer to a computer system that is used as a trap to lure an attacker to attack the resources on that particular computer system. Once the attacker performs a malicious activity, information on type, nature and tools of the attacks can be gathered. A honeypot firstly allows providing a controlled environment consisting of interesting data and resources to adversaries to intrigue them to attack and obtain them. Secondly, a honeypot allows collection of information about attackers on multiple layers and based on different interaction levels. Finally, a honeypot can serve as a deterrent against future attacks by inspecting how an attack was performed and securing the operational systems accordingly. This information gathered could be pattern of attack, unknown security holes, utilization of attack tools and techniques and even the keystrokes of an attacker. Analysis of attack logs will result in enforcement of security rules and evaluating existing intrusion detection tools and network firewalls. Honeypots however should not be viewed as a solution to network security but rather they should be seen as compliment to an existing security infrastructure [1].

Generally honeypots can be used for two main purposes; Production purposes and Research purposes. Fig.1 illustrates the general overview of honeypots and its classification.

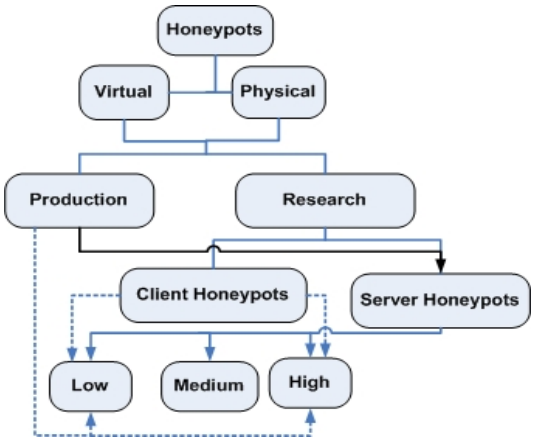

Fig. 1: General overview of honeypots' classification.

3. Traditional Honeypot's Architectures

Based on the purpose of deployment, honeypots can also be divided into server-side or client-side honeypot.

3.1. Server Honeypots

Server honeypots capture attacks by exposing open ports and vulnerable server-based services and passively wait for an attacker to find, engage and attempt to exploit those vulnerabilities. Honeypots that fall into the category of server honeypots provide different levels of interaction with attackers. Interaction generally refers to the level of activity that is granted to attacker and emulation is how much a system can pretend to be a system/service it is not. These honeypots are generalised into three different categories of low, medium and high interaction [2].

Server honeypots are either deployed alone or in a controlled space generally referred to as a Honeynet. A honeynet consists of multiple connected honeypots in a network environment where inbound access is easily granted and traffic is monitored and controlled using intrusion detection systems, packet sniffers and firewalls. Outbound connection is however limited or disabled to prohibit a compromised system to be used

as a launchpad against other hosts in outside networks. Server honeypots do not necessarily have to be located in a single geographical location but can be deployed as distributed honeypots in large numbers in multiple sites. Distributed honeypots are generally designed based on client-server architecture and transfer attack information back to a centralised location for analysis and statistical purposes. Fig. 2 depicts an example of a distributed honeypot architecture.

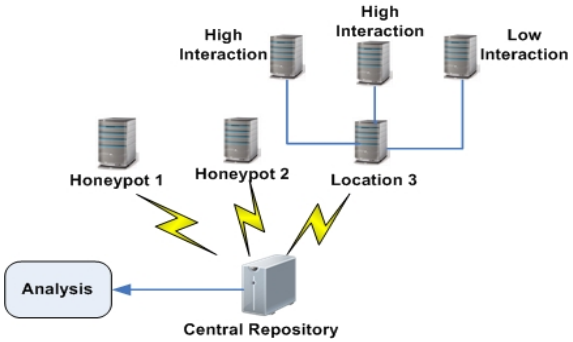

Fig. 2: Distributed Honeypot Architecture

HoneyPharm [3] is a client/server system that manages, reports, and analyses all distributed honeypot instances to one central repository. This type of hybrid honeypot is divided in two sides; server node and client node. Nepenthes honeypot is installed on the client side to collect the malware and related data. This data is then sent to the server, which is the central repository. The data which is sent by client side also includes information such as the place where the malware was disseminated and the time when the malware was captured.

3.2. Client Honeypots (Honeyclients)

Traditional honeypots are server-based systems that passively wait on a network to be attacked. Passiveness of these honeypots contributes to their limited view of the attacks launched on other systems and on an end user's applications such as web browsers. Client honeypots take a different approach to identify attacks on client systems. They actively crawl into web servers, retrieve website contents and by monitoring disk read/write activity or matching the content against known exploits, determine if a website is hosting malicious content. If any changes occur after each visit, it may be concluded that the web server has installed a malware or inserted a code to exploit vulnerability either in browser or operating system through the browser. The website retrieval process is done using either real or simulated browsers or by crawlers. Website retrieval and analysis is the most efficient mechanism used by all client honeypots [4].

4. Hybrid Honeypot Designs

4.1. Design 1 - Client-Server Hybrid Honeypot

A major factor that significantly lowers the efficiency of server honeypots in capturing attacks is their passiveness in attracting attackers through exposure of their vulnerable services. Client honeypot on the other hand, expose client vulnerabilities and actively seek threats that would exploit those vulnerabilities. The system [5] proposes integrating the active property of client honeypots into server honeypots to overcome their passiveness. The system utilises modules that interact with web servers to increase exposure of its vulnerable services by letting the IP and host information of the honeypot be recorded by websites that are suspected to be operated by hackers or organisations that participate in malicious activities. This exposure increases a honeypots online fingerprint and results in a higher number of captured attacks. Interaction with web servers through integrated client honeypot module also creates traffic which decreases the probability of detection through detecting of lack of traffic in and out of a honeypot. Features and capabilities of hybrid honeypot system are:

- Incorporation of client side honeypot modules into server-based honeypots.
- Utilising crawler and customised browser for interaction with web servers.
- Increasing exposure of honeypot through active interaction with web servers.
- Exposure and monitoring server side services and vulnerabilities through a server honeypot.
- Increasing deception by generating traffic.

4.2. Design 2 - Low and High Interaction Hybrid Client Honeypots

Low interaction client honeypots differ from their high interaction counterpart on two critical components: visitor and analysis engine. The visitor in a low interaction client honeypot consists of simulated browser or crawler while a high interaction client honeypot employs real browser to retrieve website content. Low interaction client honeypots also employ pattern matching through antivirus or snort signatures for exploit detection. High interaction client honeypots on the other hand rely on system state after each visit for attack discovery. PhoneyC [6] and Capture-HPC [7] are examples of low and high interaction client honeypots respectively. A hybrid design of low and high interaction client honeypot is depicted in [8] where a collection of website URLs are first visited by a high interaction client honeypot to provide higher detection speed. The websites that are marked as benign are passed to low interaction client honeypot for further analysis. It should be noted however, that low interaction client honeypots are generally known for their higher speed while high interaction honeypots for their detection rate. The reverse scenario in [8] is a property of the unique design that relies on multiple antivirus engines for detection, hence slower performance speed. Other architectures are designed on the principle of initial fast detection of low interaction honeypots and confirming the detection results with a high interaction.

The research [9] focuses on a hybrid design to optimize speed as well as detection accuracy. In the proposed design, low-interaction client honeypot are deployed to crawl the web at high speed. The initial detection phase is performed using low interaction honeypots. However since they produce false positive and false negative, a certain percentage of results are forwarded to the high interaction client honeypot to increase detection accuracy.

4.3. Design 3 - Low and High Interaction Hybrid Server Honeypots

Similar to a hybrid client honeypot, hybrid server honeypot designs aim to overcome the disadvantages of each low and high interaction server honeypots by combining their functionalities. This is due to the fact that each type provides a set of advantages and drawbacks which can be eliminated to a certain level by a hybrid design. As the name implies, a low interaction server honeypots provides a low level of interaction for attackers and fails to support complex functions and commands, hence gathers less information and is susceptible to detection. However, they can be deployed in large numbers and require few resources and management overhead. They are also safe to be left unattended since they simulate operating system services and cannot be taken over to attack other hosts on the network or Internet.

High interaction server honeypots comprise of real operating systems and services which are deployed in virtual machines or physical systems. Since an entire operating system is present, there is very little risk of honeypot detection, however there exists a risk of the system being used to attack other hosts upon compromise. They also require significant resources in terms of processing power, RAM and constant administrative monitoring to be deployed in large numbers. On the plus side, a large set of data can be gathered from attackers. To overcome the limited logging capabilities, and risk of detection of low interaction server honeypots and overcome the administrative and resource demanding nature of high interaction server honeypots, Hybrid architectures have been proposed and implemented.

The main purpose of research [10] is to combine the strong points of low and high-interaction honeypots to mitigate the drawbacks of each type. A large number of low interaction server honeypots can be deployed to increase exposure of vulnerable services and act as a gateway to high interaction server honeypots. A small number of high interaction honeypots need to be deployed to maintain connection and gather information on limited number of attacks that are more sophisticated and involve a human attacker. In this system, low-interaction honeypot Honeyd, acts as lightweight proxy that exposes vulnerable services and listens on open ports. If an attack is determined to be a port scan, it is ignored and dropped. However, once a real attack has been detected with finalised three way handshaking, the connection is transferred to a high interaction honeypot without attacker's notice. Consequent activities take place between the attacker and the high interaction honeypot.

Honeybrid [11] is another hybrid honeypot employing low interaction honeypots to increase exposure of vulnerable server side services. Low interaction honeypots act as front-end role, responsible for filtering the traffic. If an attack is classified as interesting, it is transparently redirected to a high interaction honeypot to

provide full interaction and collect detailed information. Attack redirection is not a new concept and has been previously implemented on Intrusion Detection Systems. Bait and Switch Honeypot [12] is an example of a system that redirects suspicious connections to a honeypot if a traffic matches a Snort signature Rule.

5. Discussion

In order to overcome the drawbacks of a certain system, components from a different system can be integrated to create a hybrid design. Studies on hybrid honeypot designs indicate that integration of different types of honeypots is a modest approach to increase their functionalities. These include, increasing their exposure thus attracting more attacks, easier management, fewer required resources and collection of larger sets of data for analysis. Table-1 illustrates the summary of methods regarding the researcher's methods.

Table 1: Summery of studied designs

Method	Server	Client	Low	High	Result
Design 1	✓	✓	Can be used	Can be used	-Increasing Exposure of Server Honeypots -Avoid Detection of Honeypot
Design 2		✓	✓	✓	-Improving detection speed and accuracy
Design 3	✓		✓	✓	-Increase exposure of vulnerable services. -Capture larger sets of data -Capture detailed information using high interaction honeypots

6. Conclusion

Collecting extensive data on attacks has always been the primary objective of honeypots. Conventional server honeypots suffer from lack of exposure, risk of detection, demand high resources and administrative supervision. Client honeypots on the other hand risk generating false positive alerts, false negative and slow performance speed. A honeypot system consisting of modules of different types of honeypots aims to overcome these issues by combining the best properties of each design.

7. Acknowledgements

We would like to thank Arash Habibi Lashkari for his guidance throughout writing this paper.

8. References

[1] Y. Mai, R. Upadrashta, and X. Su, "J-Honeypot: A Java-Based Network Deception Tool with Monitoring and Intrusion Detection", in *International Conference on Information Technology: Coding and Computing (ITCC'04)*. 2004: Las Vegas, Nevada.

[2] I. Mokube and M. Adams. "Honeypots: concepts, approaches, and challenges". in *Proceedings of the 45th annual southeast regional conference*. 2007. New York, NY, USA.

[3] A. Hassan and M. Al Ali, "Collecting Malware from Distributed Honeypots - HoneyPharm", in *IEEE GCC Conference and Exhibition*. 2011, IEEE: Dubai - UAE.

[4] J. Riden, et al., "Know Your Enemy: Malicious Web Servers". 2008, Honeynet Project.

[5] M. Mansoori, O. Zakaria, and A. Gani, "Improving Exposure of Intrusion Deception System Through Implementation of Hybrid Honeypot". International Arab Journal of Information Technology [Accepted], 2012. **9**(No. 4, 2012).

[6] J. Nazario, "PhoneyC: a virtual client honeypot", in *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. 2009, USENIX Association: Boston, MA. p. 6-6.

[7] C. Seifert. "Capture – honeypot client (Capture-HPC)". 2006 [cited 2010 10 - May]; Available from: https://projects.honeynet.org/capture-hpc.

[8] Y. Alosefer and O. Rana, "Honeyware: A Web-Based Low Interaction Client Honeypot", in *Proceedings of the*

2010 Third International Conference on Software Testing, Verification, and Validation Workshops. 2010, IEEE Computer Society. p. 410-417.

[9] C. Seifert, "Improving Detection Accuracy and Speed with Hybrid Client Honeypots", in *School of Mathematics, Statistics and Computer Science*. 2007, Victoria University of Wellington: Wellington.

[10] Kyi Lin Lin Kyaw, "Hybrid Honeypot System for Network Security". World Academy of Science, Engineering and Technology, 2008(48).

[11] "Honeybrid - Hybrid Honeypot Framework". Available from: http://honeybrid.sourceforge.net/.

[12] A. Gonzalez and J. Whitsitt. "Bait'n'Switch Honeypot". Available from: http://baitnswitch.sourceforge.net/.