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

badrul@um.edu.my omarzakaria@um.edu.my
james_cwy@yahoo.com

Abstract

This paper discusses on the development of the Honeyd@WEB. Honeyd@WEB is a system that can deploy low-interaction, production, dynamic and manageable virtual honeypots via a web interface. It runs open source programs, such as P0f (a passive fingerprinting tool) and Honeyd (a low-interaction honeypot). Honeyd@WEB can automatically determine; how many honeypots to deploy, how to deploy them, and what they should look like to blend in with the environment. The first part of this paper highlights the basic security concepts of honeypot and honeynet. The second part of this paper explains the Honeyd@WEB system. Finally, the strengths and the weaknesses of the system are discussed.

Keywords: Keyword: - Honeypot, honeynet, open source, network security, Honeyd@WEB

Introduction

Today, information is a vital element in every aspect of life. Up-to-date and correct information are the key to any successful in businesses, academia, government, personal finances or leisure activities. While this has been truer for hundreds of years, it has never been as true as in the last half of the 20th century with the invention of the modern digital computer. Security is one of the hottest issues in network today. Worries about security have soared because of the increasing magnitude of electronic commerce occurring over the Internet and the swiftly evolving business trend towards telecommuting. Therefore, more sensitive and critical information is crossing the world than ever before.

The security problem is viewed as a problem to protect information assets, both private and public. Providing solutions and safeguards are fairly straightforward. The greater problem is to educate user about the risks and the consequences if the information is stolen, compromised or lost; user roles in a secure environment and the tools that are available. Administrators find that combining together all security policy that constrains both users and attacks is time-consuming and costly. Users also become irritated at the heavy security policies that make their work more difficult for no obvious reasons and causing bad politics within the company.

Planning an audit policy on huge networks takes up both server time and resources. Most of the time, administrator does not take into account of audited events. A common attitude among users is that if no confidential work is being performed, why bother implementing security. Organisations will need to de-

termine the price they are willing to pay in order to protect their data and other valuable assets. This cost must be weighed against the costs of losing the information, hardware and disrupting services. The solution is to find the correct balance. If the data needs minimal protection and the loss of that data is not going to cost the company, then the cost of protecting the data will be less. If the data is sensitive and needs maximum protection, then the opposite is normally right.

There are organisations dedicated to breaking into computer networks. Besides this, there are literally thousands of computer attackers that roam the Internet and Bulletin Board Systems (BBSs) trading information on how to hack certain systems or networks. The recent expansion of the World Wide Web has provided attackers with nearly unlimited places to play and prey on administrators that are ignorant of basic network security. Administrators and designers must ask themselves the following questions: What is the worst thing that could happen if a malicious person had privileged access to their computer network without any authorisation? Could this event embarrass or compromise the company, the clients, or the country? How much productivity would the company lose if all of its hard drives were erased?

The computer networks of an organisation are considered to be the backbone of the organisation. The information available is crucial to the organisation and will affect its decision-making capability. It is important to secure the organisation's networks as much as possible to prevent hackers from penetrating their networks. If the attacker is able to do so, significant losses in terms of cost will incur and confidential data will be exposed. In some cases, deliberate information compromise has resulted in the loss of lives.

In addition to the basic network security infrastructure like firewall and normal security actions, organisations that have sufficient funds can deploy honeypots and honeynets technologies in their organisations. These emerging technologies play a vital role as it provides some information on the attackers' steps when they are compromising the network. This information is essential and gives us an insight of what an attacker can do to a network that has been compromised. Although every technology created by man has its benefits and disadvantages, it cannot be denied that this technology plays vital roles in our lives. By identifying what the attackers can do and are doing, vulnerabilities of the network can be discovered. Thus, certain precautions can be taken to increase the security of the network to prevent the same attacks from recurring.

Honeypot

The basic idea of a honeypot is quite old and was used already for quite a long time. Prior to honeypots, there was the seminal narrative by Clifford Stoll of monitoring and tracking an intruder (Stoll, 1998). Stoll described how he created a complete but non existent government project with realistic but false files which attackers spent an extended period of time downloading and analysing, thus providing an opportunity to monitor and trace the attackers. The original Honeypot computer systems are documented in the two proceedings that are presented by Bellovin and Cheswick (Bellovin, 1992, Cheswick, 1992). Although, the word "Honeypot" is a new phrase but the technology is not new and is getting more and more crucial. Possible definitions of what a honeypot is:

Spitzner, L. (2003) defines the term "Honeypot" as follows:

> A honeypot is a resource whose value is being in attacked or compromised. This means, that a honeypot is expected to get probed, attacked and potentially exploited. Honeypots do not fix anything. They provide us with additional, valuable information.

Honeypots are important because they allow interaction between the attacker and the honeypot. All traffic from and to a honeypot is considered to be unauthorised activity. Compromised

honeypots are not a threat to the security of the network but rather it aids us by collecting the ~~data~~ generated data. All data collected by a honeypot is consequently interesting data. Data gathered by a honeypot is valuable and can lead to a better understanding and awareness which in turn can assist us in increasing overall network security. Ironically, honeypots are not as complex as Honeynets. Honeynet is a network of computer systems consisting of multiple honeypots, intrusion detection systems and firewall components. All inbound and outbound data is controlled, captured and analysed. These systems provide real operating systems and real services that closely resemble the actual conditions found in many organisations today. Each system within the network is really a honeypot, a system designed to be attacked. Exclusively, it is a high interaction honeypot designed primarily for research, to gather information on the enemy. This high interaction can teach us a great deal about intruders, everything from how they break into systems to how they communicate and why they attack systems. Nevertheless, these honeypots are fully functional systems. Therefore, the following aspects need to be considered while setting up the honeypot (Even, 2000; Sink, 2002):

- The honeypot system should appear as generic as possible. This is one reason sniffer based systems are used in conjunction with honeypots. If a Microsoft NT based system is deployed, it should appear to the potential intruder that the system has not been modified or they may disconnect before much information is collected.
- The honeypot is designed so that an attacker cannot easily use the honeypot as a launch point for further attacks against networks.
- The honeypot should appear to contain genuine and interesting information, to entice attackers to stay on the site while tracking their moves.
- Honeypots may be set up in front of a firewall, in the demilitarised zone (DMZ), or behind a firewall. In general, it is best to set up a honeypot behind a firewall. In addition, the closer the honeypot is to actual servers (which are likely behind a firewall), the more likely it is to tempt intruders.
- Being careful in the type of traffic that an intruder can send back out to the Internet is vital as to avoid becoming a possible launch point for attacks against other entities on the Internet. (One of the reasons for installing a honeypot inside of the firewall)
- Making the honeypot an interesting site is necessary by placing “Dummy” information or making it appears as though the intruder has found an intranet server. Expect to spend some time making the honeypot appear legitimate so that intruders will spend enough time investigating and perusing the system so that forensic information can be gathered as much as possible.

System Administration and Networking Security (SANS) Institute (SANS, 2002) also cites Even (2000) and Sink (2002) goals for setting up a honeypot:

- Learns how intruders probe and attempt to gain access to systems. The general idea is that since a record of the intruder’s activities is kept, it is possible to gain insight into attack methodologies to better protect the real production systems.
- Gathers forensic information required to aid in the apprehension or prosecution of intruders. This is the sort of information often needed to provide law enforcement officials with the details needed to prosecute. Someone with far superior skills who is out there and poised to attack the system may only take a few hours after it is up, to discover it.

Honeyd@WEB

In order to manage honeypot system using web interface, Honeyd@WEB is created. Through web interface, Honeyd@WEB can deploy low-involvement (low-interaction), production, dynamic and manageable honeypot. It uses a combination of deployment strategies, such as, “Deception Ports on Production Systems” to simulate honeypot services, substituted for well-known services (for instance HTTP, SMTP, POP, DNS and FTP) and “Proximity Decoys” where the honeypot decoys are in close proximity to the production hosts (in the same logical subnet). The risk is minimised because there is no real Operating System present and services are simulated. This deployment is easier to deploy and maintain. Furthermore, the emulated services reduce the risk by containing the attacker’s activity. The attacker will never have access to an operating system to do further damage. However, only limited information is logged. It is also easier for an attacker to detect a low-interaction honeypot in this particular architecture. No matter how good the emulation is, a skilled attacker can eventually detect its presence. Another disadvantage is that it will not allow the researcher to capture any additional data associated with the attack other than the initial probe. The honeypots are deployed in the same logical subnet to distract an attacker from the real targets. They are used as a bait to bind attacking attempts as long as possible and protect the productive environment in the meantime. The primary interest here is to protect the real systems. The purpose of running Honeyd@WEB in the intranet is to detect internal attackers. It is also possible to detect a misconfigured firewall using an internal honeypot. In addition, implementation of the Honeyd@WEB is a great way to detect worms or Trojans.

A similar architecture was used by Georgia Institute of Technology for its Honeynet deployment (Levine, Labella, Owen, Contis, & Culver, 2003). Although its type of deployment was somewhat limited in their ability for data capture and data control, they were highly effective in detecting automated attacks or beginner level attacks against targets of opportunity on the network. Their limitations in data control made it possible for a hacker to fingerprint them as a honeynet. They also offered little to a skilled hacker to attract them to target the honeynet.

For Honeyd@WEB, it provides a web-based approach to setting up and deploying low-involvement (low-interaction), production, dynamic and manageable honeypots. Honeyd@WEB makes use of open source programs specifically P0f and Honeyd. The Arp daemon (Arpd) is also used in conjunction with Honeyd (Anuar & Yao, 2004).

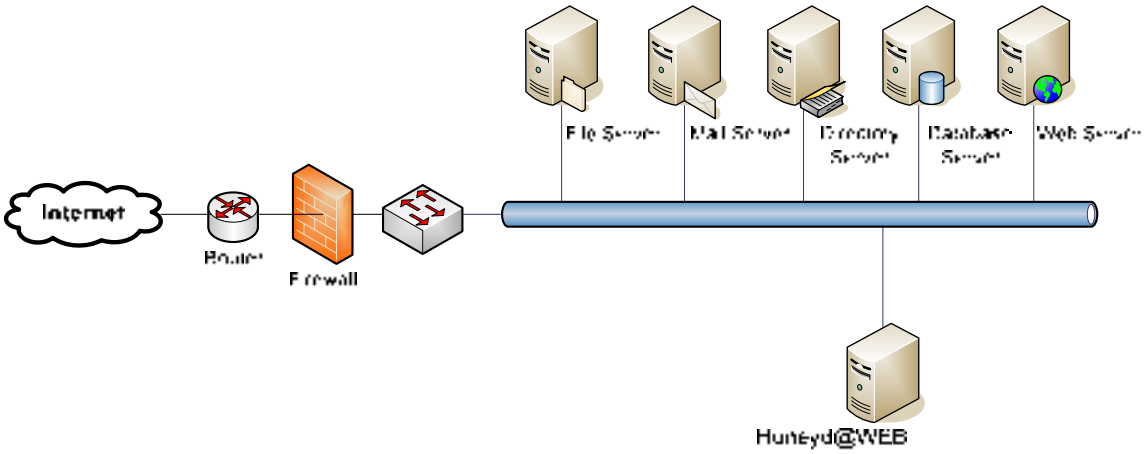

Figure 1: Honeyd@WEB’s Placement

Honeyd@WEB sets and deploys the honeypots using a web-based approach. It runs P0f, Arpd and Honeyd through the web interface. The P0f output is used as a guide in setting the honeypots

templates. The P0f program (P0f, 2003) used is a useful passive operating system (OS) fingerprinting tool developed by Michal Zalewski. P0f performs a passive fingerprinting technique (Smith & Grundl, 2002) based on information coming from a remote host when it establishes connection to a system. The passive OS fingerprinting technique is based on analysing the information sent by a remote host while performing usual communication tasks, such as, a remote party visiting a web page, connections to a machine, or a connection to a remote host with a browser or by other standard means. In contrast to active fingerprinting with tools such as Nmap (Nmap, 1998), the process of passive fingerprinting does not generate any additional or unusual traffic, as a result this process cannot be detected easily.

Captured packets contain enough information to identify the remote OS, through the subtle differences between Transport Control Protocol/Internet Protocol (TCP/IP) stacks, sometimes certain implementation flaws. Some additional metrics can be used to gather information about the configuration of a remote system or even its ISP and network setup. P0f actually provides three different detection modes:

- Incoming connection fingerprinting (SYN mode, default): whenever you want to know what kind of system, the person who connects to you runs.
- Outgoing connection (remote party) fingerprinting (SYN + ACK mode): to fingerprint systems that you or users around you connect to.
- Outgoing connection refused (remote party) fingerprinting (RST + mode): to fingerprint systems that reject the traffic.

The main advantage of the passive fingerprinting technique is that it can be used in conjunction with the information logged by firewalls and Intrusion Detection System (IDS). This can give valuable information of the systems used by attackers and potentially help track down the attackers without the attacker knowing about it. Active fingerprinting, on the other hand, will most likely be detected and stopped by the network protection tools at the remote network and could, in worst case, lead to legal proceedings. In addition, P0f can run off-line and sift through large amounts of input data from various logs, such as, firewall logs, IDS logs, and router logs for long periods of time. All this information can be extracted and analysed. Interesting information of the systems connecting remotely to the network can be gathered. The information in the packets being analysed by P0f has often not been changed by the remote network's network devices such as proxies and network address translation.

ARP (Address Resolution Protocol) by itself is a protocol that maps Internet Protocol address (IP address) to a physical machine address that is recognised in the local network. An Arp daemon (also known as Arpd) listens on a specified interface and answers ARP requests for some desired IP addresses. Here, Arpd is used to direct network traffic towards the honeypot and to get Honeyd to respond to all unused IP addresses on the network.

Honeyd (Provos, 2003) is a small open source daemon that can create virtual hosts on a network. The hosts can be configured to run random services and their personalities can be modified so that they appear to be running certain versions of Operating Systems. In addition, Honeyd enables a single host to claim multiple addresses on a LAN for network simulation. It is possible to ping the virtual machines or even to run the *traceroute* command to trace the route used. Any type of service on the virtual machine can be simulated based on a simple configuration file. A different architecture was used in the implementation of the Hit@Me honeypot.

The honeypot Hit@Me was a multi-homed host. One network interface card was used for the connectivity to the Internet (the so-called external interface) and the other one served as an administrative interface. However, Hit@Me was a fine high-involvement honeypot. The Hit@Me strength was that the entire honeypot consisted of only one machine. But this was also its weak-

ness. Through this one-machine design, it was not possible to have any influence on the traffic that was targeting or leaving the honeypot. To minimise the risk of becoming an attacking machine, this sort of honeypot should only be run under surveillance. It was then guaranteed that an operator could shutdown or reset a virtual machine if abnormal conditions occurred. A detailed description of the Hit@Me honeypot implementation is available (see Baumann and Plattner, 2002).

Other implementation likes Honeyd is Honeytraps. Honeytraps architecture was discussed by Yasinac and Manzano (2002) through a draft article in the Internet. Honeytraps are systems (honeypots or honeynets) that are designed to be compromised. Honeytraps were developed with two different architectures, serial and parallel. The first is the serial architecture. This architecture works by placing the honeytrap between the Internet and the production systems. In this configuration, the honeytrap acts as a firewall. All recognised users are filtered to the production system while attackers are contained in the honeytrap. The attackers' activities are monitored and all the information collected is routed to another system that is protected by a firewall, to ensure the integrity of the data. The serial architecture forces the attackers to go through the honeytrap to attack the production system thus exposing all attackers to the honeytrap monitoring techniques. Nevertheless the drawback of honeytrap is resource intensive. One of the important characteristics of honeytraps is that they need not deal with real users, thus reducing the volume and complexity of monitoring. However, in the serial connection, the honeytrap must handle all traffic going into the production system and re-route the authorised user to the production system. This architecture runs the fundamental risk that the intruders that are attracted into the honeytrap may subsequently attack the production systems successfully in spite of the best containment efforts of the honeytrap.

The second architecture of the Honeytraps is using parallel architecture. Alternatively, the parallel configuration allows the honeytrap to be independent of the production system. As with the serial configuration, the information gathered about attackers' activities in the honeytrap is re-routed to a separate, protected system. This architecture is less resource intensive so it can be implemented in a system with fewer resources. As with the serial architecture, there are several drawbacks with the parallel honeytrap architecture (Yasinac & Manzano, 2002).

Firstly, for the honeytrap to be useful during the forensic process, Honeytrap and production systems must have been attacked independently. Configuring the honeytrap so that it is likely that an intruder would enter or probe the honeytrap before or shortly after entering the production systems is tricky and again leads us into possible entrapment scenarios (Yasinac & Manzano, 2002).

Secondly, under the parallel honeytrap architecture it is likely to be more difficult to connect an intruder in the honeytrap and the production system using parallel configuration. Honeytrap and the production system are connected using indirect connection (separated connection).

The Web Interface

Honeyd@WEB uses a web-based approach in setting up and deploying the virtual honeypots. The web interface allows the following:

- Run P0f to gather necessary network information.
- Run the arp daemon to listen to a specified range of unused IP addresses.
- Run Honeyd to deploy the virtual honeypots.
- Set and create the honeypots' template for Honeyd to use.

P0f is used to gather network information that may be useful when it comes to setting up the honeypots to match the current environment as much as possible. The output generated by p0f

will act as a guide for the next phase; setting up the virtual honeypots. Two options are available when it comes to setting the honeypots.

For the first option, a recommendation of honeypots to be deployed is given. The recommendation given is based on the network information gathered by p0f. Based on this information, certain honeypots are recommended for deployment as to match the current production environment. However, the characteristics of each honeypot have been predefined in a general context. A decision needs to be made whether to deploy the recommended honeypots, which is depending on the requirements. For the second option, the type of honeypots to be deployed can be selected. This allows some form of control over the templates settings; the name of the honeypot can be specified, services to emulate can be selected and the unused IP address that it should take can be assigned. If needed, the honeypots configured can be edited and deleted. In other words, everything will require some form of configuration. Nevertheless, it will take some time and effort when it comes to the configuration of the characteristics of the honeypots.

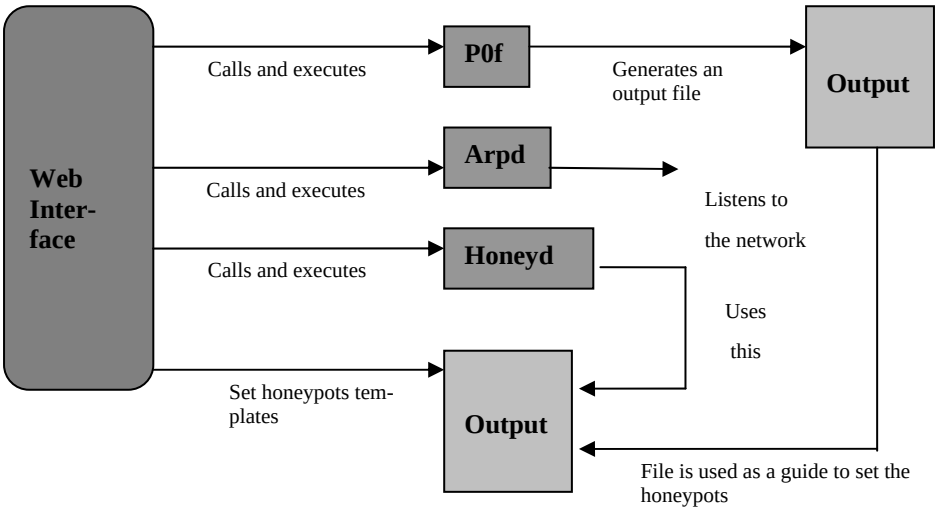

Figure 2: Web Interface's Functions

It is suggested that the recommendation approach be used instead of configuring each of the honeypot. Nonetheless, the final decision is up to the individual using it. For logging and analysis purposes, snort will be used to log the attacks made. Having to have configured the necessary templates and deploying the virtual honeypots through Honeyd@WEB, the Analysis Console for Intrusion Databases (ACID) can be used to view and analyse the attacks attempted on the virtual honeypots. Nevertheless, the level of dynamicity and adaptation that Honeyd@WEB has is quite restricted. Further efforts are being taken to improve the dynamicity and adaptation level of Honeyd@WEB. Figures 3 and 4 illustrate the web interfaces of the Honeyd@WEB.

Figure 3: Main Interface (Top Half)

Figure 4 : Main Interface (Bottom Half)

Why is Honeyd@WEB Recommended?

The motivation of developing Honeyd@WEB using open source programs is because they are freely available on the Internet and can be used without any major issues. Honeyd@WEB is just a prototype and has certain limitations but extra efforts are being made to add more features and to enhance it further. It also provides an insight of the possibility of building a dynamic honeypot using a web-based approach. Using a web based approach; Honeyd@WEB is able to provide a web interface to accomplish all that is necessary in deploying virtual honeypots. It is simple in nature and quite straightforward.

Honeyd@WEB will allow virtual honeypots to be deployed through a web interface. Necessary tools and programs can be run as well using the web interface. Virtual honeypots can be deployed either by following the recommendation given; recommended honeypots to deploy or allowing the individual to choose what type of honeypots to deploy and how to deploy it by selecting the services necessary that it should emulate, and giving the individual more flexibility. Analysis and logging of attempted intrusions are also made possible through the use of snort and ACID.

Honeyd@WEB's Strengths

Honeyd@WEB uses a web-based approach to deploy the honeypots, thus allowing everything to be done through the web interfaces from running the necessary tools to configuring the honeypots. It also provides a method of setting up dynamic virtual honeypots through Honeyd that matches the current environment as much as possible. As the network environment changes, the changes are also reflected in the recommended honeypots to be deployed because they are always based on the latest Operating Systems information gathered by P0f that passively monitors the network.

Virtual honeypots can be viewed, added and deleted if desired. Recommended honeypots can be deployed using one of the P0f archive log files or using the current and latest network information gathered by P0f. Honeypots can be managed (configured), where the user can create the preferred honeypots if desired.

Logs generated by Honeyd can also be viewed through the web interface. In addition, certain groups of data can be edited, namely the services to emulate, the personalities (Operating Systems) to mimic and IP addresses available for use. This is made possible as each data group has its own table.

Honeyd@WEB provides flexibility whereby allowing the number of honeypots that can be deployed to be changed easily. Furthermore, the number of services that each honeypot can emulate can be modified as well. Besides that, Honeyd@WEB makes use of open source security tools like Snort. Snort, has a proven track record when it comes to intrusion detection. Snort integrated together with ACID can provide quicker, more efficient and effective analysis of intrusion attempts.

Simplified management and administration is possible using the web interfaces to assist in setting up the virtual honeypots and maintaining them via Honeyd@WEB which can acts like a control centre. Furthermore, Honeyd@WEB can be run from anywhere within a Local Area Network (LAN) via browser.

Honeyd@WEB's Weaknesses

The dynamic property of Honeyd@WEB is limited. This is because if there are certain changes in the network, it requires the user to make a decision whether there is a need to re-deploy the virtual honeypots based on the latest information or not make any changes at all but just keep the

current virtual honeypots configuration. Honeyd@WEB uses a simple method when it comes to giving the recommendation of what type of honeypots to deploy.

Honeyd@WEB relies on the open source programs that have been proven and developed by the open source community. Honeyd@WEB utilises quite a number of open source programs; specifically P0f, Honeyd, Snort, MySQL and ACID. Currently, Honeyd@WEB's capabilities are limited to the Linux platform only as Linux can only support open source programs.

Careful configuration and proper installation are essential to ensure the smooth running of Honeyd@WEB. This is due to the fact that it employs a number of open source programs and some of these programs could depend on other open source programs to work correctly. Therefore, it is vital that every single configuration step is followed exactly. If one step is missed or configured wrongly, Honeyd@WEB would not be able to work.

Future Research

The dynamic property of Honeyd@WEB can be further improved which includes; firstly, the gathering of information about the network in terms of the unused IP address range and services running on the network.

Secondly, this development can be enhanced through the algorithm used, which can provide recommendations and options on how the honeypot can be deployed. This algorithm will help the user in deciding what Operating Systems to emulate, what services to emulate, and what IP address it should have. The user just needs to select an option. In addition, the user is able to change or alter any relevant information if desired.

Thirdly, the adding of the number of relevant security tools that can assist in the analysis of intrusions and decision making process. Then, integrating all these security tools to provide more efficient and effective analysis of intrusions; thus allowing reaction time to be much swifter.

Finally, artificial intelligence technique can be used to create dynamic and intelligent honeypot. As a result, the honeypot is able to think and learn by itself and fend for itself. In addition, increasing its ability to be able to operate and adapt in a research environment that tends to have higher levels of interactions.

Conclusion

Honeypots and honeynets are new fields in the sector of network security. At present, there is a lot of ongoing research and discussion all around the world about them. Together, they are indeed an emerging technology that has great potential in the world of network security. To be able to defend networks, hackers' techniques must be learned and identified. This has been made possible through the deployment of honeypots that are less complex compared to honeynets that have varying levels of complexity depending on the deployment technique. As technology continues to strive, many have realized and recognized the importance of honeypots and honeynets in discovering hackers' techniques. Using this technology, vulnerabilities that exist in a computer network can be identified based on the attack methods used.

Although a honeypot does not address and solve a particular problem, it is an instrument that plays a unique part in the overall security architecture. A honeynet is said to be an extremely controlled network with honeypots as part of the honeynet architecture. Systems are set up in such a way that they mimic typical organisations' network environment. Problems and vulnerabilities exposed are very real to the environment. Honeypots provide highly valuable and beneficial data that are much easier to interpret. These data give an insight of what the attack patterns are like, allowing faster analysis and reaction time. Honeypots technology has evolved as time progresses and contributed much to the research of the studies on the blackhat community.

Honeyd@WEB has provided an approach of deploying low-involvement (low-interaction), production, dynamic and manageable honeypots using a web interface; using P0f that passively monitors the network and Honeyd that sets up the virtual honeypots to be deployed. As the network environment changes, the changes are also reflected in the recommended honeypots to be deployed because they are always based on the latest Operating Systems information gathered by P0f. Virtual honeypots deployed using Honeyd can be created automatically or configured through Honeyd@WEB. In addition, Honeyd@WEB has provided a method of running the necessary tools – P0f, Arpd and Honeyd through the use of the web interfaces available. The open source programs used to develop Honeyd@WEB specifically P0f and Honeyd has proven to be excellent tools in their own functional framework. The usage of Snort and ACID has made the analysing of the intrusion attempts made on the virtual honeypots to be quicker, more efficient and effective. There are numerous brilliant open source programs that have made a great deal of contributions to the area of network security such as Snort, Honeyd, P0f, Nmap and many more.

The future of honeypots will be dynamic and intelligent honeypots. Slowly, honeypots that needs configuring will become obsolete. Future honeypots will be a plug-n-play solution. By integrating the capabilities of honeyd with the capabilities of a passive fingerprinting tool such as p0f, dynamic honeypot is not impossible. However, this solution is not entirely perfect; it can be improved further in terms of the level of interaction between these two applications. Besides that, determining what information to use and how to use this information is vital as it will decide the effectiveness and the efficiency of this so known as dynamic honeypot. Dynamic honeypots may not be an entire perfect solution. There are works towards to this potential area of research. Dynamic honeypots can only be improved and enhanced from time to time and slowly but surely it will achieve all the purposes for which it was built (Anuar & Yao, 2004).

References

"ACID" (2003). *Analysis console for intrusion databases.* Retrieved June 20 2005, from http://www.cert.org/kb/acid/

Anuar, N. & Yao, C. W. (2004). Dynamic and manageable honeypot: Honeyd@WEB. *Proceedings of 1st International Conference on Informatics 2004*, Kuala Lumpur, Malaysia, 28 – 30 July 2004.

Baumann, R. & Plattner, C. (2002). *Honeypots*. Diploma Thesis, Swiss Federal Institute of Technology. Retrieved May, 20, 2005 from http://security.rbaumann.net/download/diplomathesis.pdf

Bellovin, S. (1992). There be dragons. *Proceedings of the Third Usenix Security Symposium*, Baltimore MD.

Cheswick, B. (1992). An evening with Berferd in which a cracker is lured, endured, and studied. *Proceedings of the Winter USENIX Conference*, San Francisco.

Even, L. R. (2000). *Honey pot systems explained*. Retrieved May, 20, 2005 from http://www.sans.org/resources/idfaq/honeypot3.php

Levine, J., Labella, R., Owen, H., Contis, D., & Culver, B. (2003) The use of honeynets to detect exploited systems across large enterprise networks. *Proceedings of the 2003 IEEE Workshop on Information Assurance*, 18-20 June 2003, West Point, NY, pp. 92 – 99.

"Nmap." (2005) Nmap - Free Security Scanner for Network Exploration & Security Audits. Retrieved October, 24, 2005 from http://www.insecure.org/nmap

"P0f." (2003). *P0f*. Retrieved October, 24, 2005 form http://lcamtuf.coredump.cx/p0f.shtml

Provos, N. (2003) Honeyd - A virtual honeypot daemon. *10th DFN-CERT Workshop*, Hamburg, Germany.

Sink, M. (2002). The use of honeypots and packet sniffers for intrusion detection. Retrieved November, 5, 2005 from http://www.giac.org/practical/gsec/Michael_Sink_GSEC.pdf

Spitzner, L. (2003). *Honeypots: Tracking hackers*. Addison-Wesley.

Stoll, C. (1998). Stalking the wiley hacker. *Comunications of the ACM, 31*(5), 484-497.

"SANS." (2002). *SANS.* Retrieved May, 20, 2005 from http://www.sans.org.

Yasinac, A. & Manzano, Y. (2002) *Honeytraps, A network forensic tool (Paper Draft)*. Retrieved May, 20, 2005 from http://www.cs.fsu.edu/~yasinsac/Papers/YM02.pdf

Biography

Nor Badrul Anuar obtained his Master of Computer Science from University of Malaya in 2003. Currently, he is a Lecturer at the Faculty of Computer Science and Information Technology in University of Malaya, Kuala Lumpur. His research interests include Computer Network Security, Open Source and IS/ICT.

Omar Zakaria obtained his B Comp Sc from Faculty of Computer Science & Information Technology, University of Malaya, Malaysia in 1994, and his MSc in Information Security from the Information Security Group, Royal Holloway, University of London, UK in 1996. He is currently a PhD candidate at the Information Security Group, Royal Holloway, University of London, UK. His research area is information security management. He has published a number of papers related to information security areas.

Chong Wei Yao obtained his Master of Computer Science from University of Malaya in 2005. At present, he is working as an IT Specialist. His research interests include Computer/Network Security and OpenSource.