Intrusion Detection System using Virtual Honeypots

Prof. Smita Jawale (Department of Computer Engineering, VCET)
Rishi Mehta, Vivek Mahalingam, Niyoshi Mehta

(Department of Computer Engineering, VCET, Mumbai University, India. Email id-
rishi91991@gmail.com,vivekk251@gmail.com ,niyoshimehta@yahoo.co.in)

Abstract:
The most significant drawbacks in the existing intrusion detection systems (IDSs) are traffic overload, unknown attacks, false positives and false negatives. We propose the design of AAIDHP (An Architecture for Intrusion Detection using Honey Pot), for nullifying the drawbacks of the existing systems. As a component, the honeypot cooperates with IDS, which increases flexibility, configurability and security of IDS.A honeypot will let the user make an attempt to intrude the system, thereby observing the intruder's activity and creating intruder signatures. However, the major limitation of the honeypots technology is that nowadays there are tools to detect honeypots .In order to hide honeypots, we propose the use of 'virtual honeypots' that is based on virtualization technology. We also present the definition of the honey pot, the description of this approach and a discussion of design.

1 Introduction
The advancements in the distributed computing technology has enabled a high level of interconnectivity among the machines, thereby creating revolution in communication and reducing workload by distributed work processing. This interconnectivity emphasizes the long standing problem of providing security in a distributed system by introducing many more possible attacking points. This has resulted in a tremendous increase in the intrusions and thereby leading to enormous economic loss as well as data loss. Also, identity theft is a major problem in this. Hence, we propose the inclusion of honeypots as a mechanism for intrusion detection. A honeypot is a decoy system employed in the production system for tracking the activity and characteristics of the intruder.

Because the primary objective of a honeypot is to detect enemies without being known to them, it is important to hide its existence. However, as several studies have reported [10], exploiting the property of consecutive addresses allocated to the honeypots, they can be easily traced rendering them useless. In fact, there exist some anti-honeypot tools that intelligently probe IP address space to locate Internet security sensors including honeypots.

An apt solution to this problem is concept of Darkpots, consisting of large number of virtualized honeypots. These virtual honeypots have non consecutive IP addresses that are unused in the production network. Darkpots enables us to deploy a large number of honeypots within an active IP space used for a production network; thus detection is difficult using existing probing techniques. Apart from that, by virtually classifying the unused IP addresses into several groups, Darkpots enables us to perform several monitoring schemes simultaneously. This function is meaningful because we can adopt more than one monitoring schemes and compare their results in an operating network.

2 Definition of honey pot
A honey pot can be defined as a "decoy" system that has a non-hardened operating system or one that appears to have several vulnerabilities for easy access to its resources. The decoy system should be set up in a similar manner to those of the production servers in the corporation and should be loaded with numerous fake files, directories, and other information that may look real. By making the honey pot appear to be a legitimate machine with legitimate files, it leads the hacker to believe that they have gained access to important information

In a word, honey pot provides an environment where intruders can he trapped or vulnerabilities accessed before an attack is made on real assets.

We propose the IDS with honey pot as its component solves all the problems mentioned in section

A honey pot is designed to he compromised, not to be used for production traffic. Any traffic entering or leaving the network is suspicious by definition. This concept of no production traffic greatly simplifies the data capture and analysis.

A honeypot is a host that has no real purpose, other than to capture unauthorized activity. So honey pot reduces this problem by not having any true production traffic.

False negatives are another challenge. Because there is little or no production activity within a honey pot, the honey pot reduces false negatives by capturing absolutely everything that enters and leaves itself. This means all the activity that is captured is most likely suspect

As **to unknown** activity, even if IDS misses it, we have captured the activity. We can review all of the captured activity and identify the attack.

3 Main technologies of the AAIDHP (architecture)

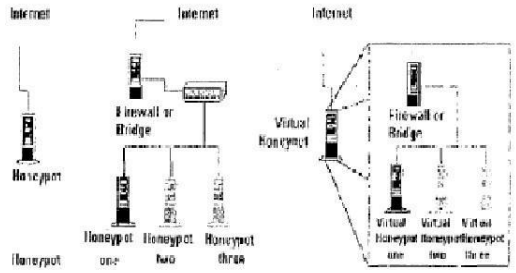

Fig. AAIDHP.

We use some technologies to increase flexibility, configurability and security of the AAIDHP. Attackers have no influence on interacting with the compromised honey pot simultaneously.

3.1Data Control

When an attacker breaks into a honey pot, they may initiate connections out of the network for a variety of reasons (download toolkits, setup automated bots, IRC chats, send emails, etc). And the honey pot will he commonly used as a bouncer. The purpose of Data Control is to ensure attackers cannot apply the honey pot to attack or harm other systems. Data Control mitigates risk. We will implement two technologies

3.1.1 Connection-limit: We contain how many outbound connections an attacker can initiate from a honey pot. We use iptables to set how many times an attacker can initiate a TCP, UDP, ICMP outbound connection. The iptables is the user space command line program used to configure the Linux 2.4.x and 2.6.x IPv4 packet filtering rule-set. It is targeted towards system administrators. One nice feature of iptables, when the TCP limit has been met, it does not affect any of the UDP, ICMP or OTHER traffic, until their limits have been met also. On average, the AAIDHP allows five to ten outbound connections per an hour. Every time a connection is initiated outbound, the ipables counts them. When the limit is reached, iptables blocks any more connections from the honey pot. In general, when a worm infects the honey pot, there will he lots of iptables logs. At that time, thousands of scans arc going out every second. In a short time, there will be many logs with different DST. So the worm can be detected.

3.1.2 Rate limit: After the intruder controls the honeypot completely, he will probably launch denial of service attack (DOS) to other hosts in the secured network. So we need a defense in depth against DOS attacks. As an aid of Connection- limit, we use Rate-limit to restrict the spread of these attacks. In computer networks, rate limiting is used to control the rate of traffic sent or received on a network interface. Traffic that is less than or equal to the specified rate is sent, whereas traffic that exceeds the rate is dropped or delayed. A device that performs rate limiting is a rate limiter .For example if the attack is employing lCMP packets or TCP SYN packets, the system can be configured to specially limit the bandwidth with those types of packets. This will allow some of these packets that may belong to legitimate network flow to go through. And DOS is avoided meanwhile.

3.2 Multi-level log mechanism (MLLM)

The purpose of the MLLM is to log all of the attacker's activity. This is the whole purpose of the honey pot, to collect information. Without it, the honey pot has no value. The key to MLLM is collecting information at as many layers as possible. Single layer is not secure and no single layer tells us everything. The AAIDHP has identified two critical

layers of MLLM. The honey pot captures the attacker's activity. There is detailed information of attacks such as the processes started, compiles, file adds, deletes, changes, and even key strokes etc in the system logs. This information is critical, as its own first indication of what an attacker is doing. Obviously the system logs cannot be kept on the honey pot exposed to the hacker. Thereby we transmit them via UDP to a remote machine named "Remote Log Server". Attackers cannot see, nor sniff these packets. But more advanced attackers will compromise the "Remote Log Server"n an attempt to cover their tracks. So the second element is capturing every packet and its full payload as it enters or leaves the honey pot. The "Sniffer Server" can do it and writes down all the packets in the bin& log files. In this way, even if backers have broken into the "Remote Log Server" and destroyed all the logs in this host there are still intruder's behaviors in those binary log files.

4 Architecture of the AAIDHP

The Architecture of the AAIDHP is shown in Figure. This figure shows eight essential components of the architecture: "Remote Log Server", "Sniffer Server", "Honey Pot", "IDS", "WWW Server", Switch, Router and Fire Wall. "IDS" is the host for intrusion detection and "WWW Server,,,, is the secured host in the network. Switch is used for the Data Control mentioned in section 3.2 and Router for the Route Control. There is another function to set up the Router here. It creates a network environment that more realistically mirrors a production network. So the trap of the honey pot is not easy to be found. In this paper, we work hard at the integration of the honey pot with IDS and Fire Wall. We want to buildup a cooperative system to detect intrusion.

Honey pot is by no means the only method to collect data, however it has the advantage of reducing false negatives. Even if IDS misses some attacks, we can identify the attack according to MLLM. IDS can protect against these threats the next time. Traditional IDS is purely defensive. But in

AAIDHP**,** there is enough information about threats that" exist. New tools and attack patterns can be discovered. Hence, future compromise can be predicted. We use the information captured by the honey pot to correlate with the IDS"S logs. IDS can carry on frequency analysis, source analysis and statistical analysis of given theme and so on. New methods and ways of intrusion can be learned too. Further more, IDS maybe know who invade into the system. By these means, the capability of defense will be improved. The honey pot system can cooperate with Fire Wall. The system will refuse the visit of the intruder whose IP address is set in the Fire Wall **as** blacklist by the honey pot. According to the destroy degree, the term of refusing the malicious visit can be short-term or long-term. By combining data from multiple systems, these data can be used for such things **as** early warning and prediction, statistical analysis, or identification of new tools or trends. The main characteristics that we would like to achieve in the AAIDHP are flexibility, configurability and security.

4.1Flexibility: Honey pot creates a network environment that more realistically mirrors a production network.

4.2Configurability: IP trap, Data Control and Route Control can be deployed dynamically.

4.3Security: Intruders can be trapped in the honey pot before an attack is made on real assets. It is obvious that AAIDHP solves the information overload, unknown attacks, false positives and negatives. At the same time, it also increases flexibility, configurability and security of IDS.

5 DARKPOTS

The economic impact of viruses, botnets, and other malware is one of the most serious Internet security issues that needs to be addressed urgently .The annual worldwide economic damages from malware exceeded $13 billion in 2006. From the viewpoint of an organization connecting to the Internet, it is important for network administrators to be able to detect malicious activities in order to protect their network, because it is well known that many computers today are prone to various attacks originating from viruses or botnets even though end users are encouraged to keep applying latest security patches or install anti-malware software. It is a challenging task to correctly detect malicious activities, often consisting of only a few packets, from the large number of packets that carry data traffic. A straightforward approach to resolve this issue is to grasp the trends of network incidents by monitoring the logs of a firewall or a sensor server that is located at the unused IP address space (also called darknet)[15] of the network. Because the packets destined for a darknet are not legitimate in nature, we do not need to employ expensive deep packet inspection (DPI) schemes for distinguishing malicious packets from legitimate ones. While monitoring, a darknet can effectively provide information about the malicious attacks attempted on a network in a passive manner, we cannot understand the detailed information about each attack unless we actively analyze the actual connections originating from the attack sources. A honeypot is a system that aims to detect and analyze malicious attacks attempted on a network in an interactive manner. Because a honeypot works as if it is a victim computer, it is able to trace the process of infection, detect malwares, and inspect the entire picture of botnet activities from the viewpoint of infected hosts. Honeypots have attracted considerable attention as a promising approach to analyze malicious activities in a controlled environment .Because the primary objective of a honeypot is to detect enemies without being known to them, it is important to hide its existence. However, as several studies have reported, it is known that exploiting the unique characteristics of hosts working on a consecutive IP address range can easily reveal the existence of honeypots. In fact, there exist some anti-honeypot tools that intelligently probe IP address space to locate Internet security sensors including honeypots. In order to tackle this problem, we propose a system called Darkpots that consists of a large number of virtualized honeypots using unused and nonconsecutive IP addresses in a production network. Darkpots enables us to deploy a large number of honeypots within an active IP space used for a production network; thus, detection is difficult using existing probing techniques. We note that we can dynamically change a set of unused IP addresses that we use for monitoring the network. We implement a prototype of Darkpots and empirically evaluate its effectiveness and feasibility in a high-speed campus network. Although using nonconsecutive IP addresses as a darknet is not a new to the best of our knowledge, ours is the first attempt that extends this concept to an interactive honeypot, implements the system, and evaluates its effectiveness in a production network. Using the framework of Darkpots, we compare the three different monitoring schemes: passive monitoring (darknet), reflector monitoring (sensor), and interactive monitoring (honeypot).We demonstrate how the interactive monitoring is effective, compared to the other schemes. We also demonstrate that using nonconsecutive IP addresses is useful not only for hiding the existence of honeypots but also for extracting more information. That is given a fixed number of unused IP addresses, honeypots located at distributed IP addresses could collect a larger number of malware species as compared to those located at concentrated IP addresses. Thus, the randomness has statistical advantages against the biased IP selection. Our objective is to run a large number of honeypots in a production network without affecting legitimate traffic. This work leverages virtualization technologies to build DarkPots. For a set of unused IP addresses given by a network operator, Darkpots first checks the two-way traffic patterns to ensure their vacancy. After vacancy checking, Darkpots change the switching rule of a forwarder, which runs on top of a programmable switch such as OpenFlow switch That is, the forwarder switches packets for unused IP

addresses to the honeypot slice subnet connecting to physical honeypots; it switches other legitimate traffic to succeeding routers. Note that the rule of the forwarder can be periodically changed because unused IP addresses could be updated by IP address renumbering.

6 DESIGN AND ARCHITECTURE OF DARKPOTS

This section describes our proposed system, DarkPots. We first present the high-level overview of the system

6.1 High-level Overview

Darkpots composes of three primary components: vacancy checker, forwarder, and analyzer. Vacancy checker monitors all the packets and checks the network traffic patterns for each unused IP address to ensure its vacancy. We note that vacancy checker uses the information obtained from a network administrator as a source of unused IP addresses. It further double-checks the traffic patterns to ensure the accuracy. The forwarder is set at a point of presence (PoP) in a network, i.e., it is set between the gateway router(s) and the Internet. Upon receiving a packet destined to an unused IP addresses, it forwards the packet to the Analyzer, which is a set of servers that work as a sensor or a honeypot. Other legitimate packets are forwarded to succeeding gateway router(s). The analyzer processes the received packets in several manners. As shown later, the analyzer plays three different roles: passive sensor, active sensor, and honeypot.

6.2 Vacancy Checker

An unused IP address can be broadly classified into the following two cases:

(a) An IP address that is explicitly filtered by gateway router(s).

(b) An IP address that is not explicitly filtered by gateway router(s) but not assigned to any of the hosts in the network.

In case (a), all packets to unused IPs are blocked at the gateway firewall. Therefore, intercepting the packets should cause no effects. However, from our empirical test, we found that the number of IP addresses categorized into case (a) in our network was not as high as we expected. By carefully examining case (b), we can obtain more information about unused IPs in the network. We note that simply using the list of unassigned IP addresses, given by a network administrator, could cause false positives because an end-user might use an unassigned IP address by wrong configuration. In addition, the list might not have been up-to-date. To ensure the accuracy of the list of unused IP addresses, we double check their vacancies by carefully monitoring the traffic patterns. The procedure of compiling the unused IP addresses can be summarized as follows.

1) Obtain the list of blocked and unassigned IP addresses from a network administrator.

2) Check the patterns of traffic flows that are associated with the above IP addresses by monitoring two-way packet streams.

3) If the traffic patterns of an internal IP address are completely one-way basis, we add the address to the unused IP list. The list of unused IPs is synchronized with the forwarder.

6.3. Forwarder

A key role of the forwarder is to classify packets as legitimate and bogus. Bogus packets are the ones that are destined for unused IP addresses. Upon receiving an IP packet, it first checks whether the IP address of the packet is listed in the unused IP lists (i.e., bogus). If the IP address is listed in either of the lists, it rewrites the destination MAC address to the corresponding analyzer server and sends it out on a wire. As discussed in section III, we examine two allocation methods: distributed allocation and concentrated allocation.

6.4. Analyzer

The implementation of the analyzer can be done in three modes: passive sensor (A), active sensor (B), and interactive honeypot (C). The analyzer is a system that receives forwarded packets and logs for further analysis. In some cases, the analyzer sends a response to incoming packets for additional inspection of the sensor's behavior. In order to establish a connection for forwarded packets, we create virtual interfaces on each analyzer by assigning sub interfaces to a physical network interface: i.e., eth0:1.Each virtual interface is assigned an unused IP address collected from the forwarder. We note that this virtualization technique enables us to run most existing honeypot software.

Passive Sensor: The passive sensor mode acts like a sensor machine in the darknet. It never responds to any incoming packets. However, all the inbound packets are logged for further analysis.

Active Sensor: The active sensor mode waits for an incoming TCP SYN packet and responses it to with a TCP SYN/ACK packet. After sending the SYN/ACK packet, it discards the connection and will not send any packets. Thus, it creates half-open TCP connections. The active mode is useful in examining the potential establishment of attacks. The active mode acts like a reflector, as described in, and responds SYN/ACK packets as typical legit servers do. The active mode analyzes packets in response to SYN, to distinguish SYN flooding against actual infection behavior, because typical SYN flooding aims to increase TCP half-open connections for denial of service and not try to establish a three-way handshake. The disadvantage of the active mode is that it can be used as a reflector by malicious users in intermediating DDoS attacks.

Interactive Honeypot: An interactive honeypot emulates the vulnerabilities of major OSes. We can adopt Nepenthes (version 0.2.2)[19] as an implementation of honeypot. Nepenthes emulates major vulnerabilities like MS04-012 (TCP/445, RPC-DCOM) and MS02-039 (1434/TCP, SQL Server). We enabled a logging function module to record download attempts (logging download) and successful downloads (logging submission), while capturing all packets using tcp dump. In the last two modes, the analyzer spoofs a source IP address to immediately before the

received packet of the destination IP address when responding to the Internet.
Therefore, the analyzer is transparent to attackers or botnets, because it responds like a legitimated host.

7 Conclusion

Thus we see the benefits of using honeypots in intrusion detection system and using virtual honeypots that enables hiding of the honeypots and thereby creating efficient attacker's signatures and detecting intruder's characteristics deeply .While honeypots may not become the complete, self reliant security measure, however this mechanism aims at enhancing the present security features. That helps in reduction of losses due to intrusion to a greater extent. While user interface is one area where there can be lots of scope as far as honeypots is concerned.

REFERENCES

[I] Richard A.Kemmerer and Giovanni Vigna. Intrusion Detection: A Brief History and Overview. Reliable Software Group, Computer Science Department, University of California Santa Barbara. SECURITY & PRIVACY-2002
[2] Stefan Axelsson. Intrusion detection systems: a survey and taxonomy. Technical Report 99-15, Depart. Of Computer Engineering, Chalmers University, March 2000.
[3] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: alternative data models. In Proceedings of the 1999 IEEE Symposium on Security and Privacy, pages 133-145. IEEE Computer Society, 1999.
[11] T. Lane and C. E. Bradley. Temporal sequence learning and data reduction for anomaly detection. In Proceedings of the Fifth ACM Conference on Computer and Communications Security, pages
150-158, 1998.
[12] T. Lane and C. E. Bradley. Temporal sequence leaming and data reduction for anomaly detection. ACM Transactions on Information and System Security, 2295-331, 1999.
[13] C. Kreibich and C. Kanich and K. Levchenko and B. Enright and G. M. Voelker and V. Paxson and S. Savage, "Spamcraft:
An Inside Look At Spam Campaign Orchestration," Boston, USA, April 2009.
[14] C. Economics, "Malware Report: The Economic Impact of Viruses, Spyware, Adware, Botnets, and Other Malicious Code," Tech. Rep., Computer Economics, 2007, Tech. Rep.
[4]Stephanie Forrest, S. A. Hofmeyr, A. Somayaji, and T.A.Longstaff. A sense of self for unix processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, pages 12&128. IEEE Computer Society, 1996.
[5] S. A. Hofmeyr, Stephanie Forrest, and A. Somayaji. Intrusion detect using sequences of system calls. Journal of Computer Security, 6:151-180, 1998.
[6] P. Helman and J. Bhangoo. A statistically base system for prioritizing information exploration under uncertainty. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 27:449466, 1997. W. Lee, S. J. Stolfo, and P. K. Chan. Learning patterns from mix processes execution traces for intrusion detection. In Procsedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, pages 5&56. Menlo Park, CA:
AAAI Press, 1997.
[8] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In Proceedings of the Seventh USENIX Security Symposium, 1998.
[9] Anup Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In Proceedings of the Eighth USENIX Security Symposium, 1999.
[1o] T. Lane and C. E. Bradley. Sequence matching and leaming in anomaly detection for computer security.
In Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, pages 4349. Menlo Park, CA. AAAI Press, 1997.

[15] "The darknet project," http://www.cymru.com/Darknet.
[16] J. Ullrich, "Dshield," http://www.dshield.org.
[17] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The Nepenthes platform: An efficient approach to collect
malware," Lecture Notes in Computer Science, vol. 4219, p. 165, 2006.
[18] L. Spitzner, "The honeynet project: Trapping the hackers,"
IEEE Security & Privacy Magazine, vol. 1, no. 2, pp. 15–23, 2003.
[19] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in Proc. Network and Distributed Systems Security Symposium, vol. 1. Citeseer, 2003, pp. 253–285.