

Research on data control mechanism in the honeyfarm environment

Jian Li[†], Jing-Feng Xue and Chun Shan,

School of Software, Beijing Institute of Technology, Beijing, 100081, China

E-mail: 13811328115@139.com

There is always the potential of malware or attackers using a honeyfarm to infect or attack non-honeyfarm systems. Thus, each organization must implement data control of a compromised honeypot to minimize the risk. We have proposed a data control mechanism that exploits honeyfarm gateway, intrusion detection system, and reverse firewall to achieve this goal. We demonstrate that the data control mechanism not only can effectively identify the outbound traffic with attacks, but enables dynamic containment of malware inside a honeyfarm.

Keywords: Honeypot; Honeyfarm; Data Control.

1. Introduction

With the constant growth of the Internet, network security issues have become increasingly serious. With automated techniques attackers can scan specific network ranges of the Internet searching for vulnerable systems with known weaknesses. Once these attackers have compromised a machine, they install a so called zombie on it. This allows an attacker to remotely control this machine and many compromised machines which can be remotely controlled by an attacker, is called a botnet. Botnets pose serious threats, they are used as a launching platform for various types of network attack such as bulk-email, distributed denial-of-service (DDoS) and identity theft.

The honeypot has emerged as an effective tool to discover and understand the tactics and motives of attackers. A honeypot is a security resource whose value lies in being probed, attacked, or compromised. Honeypots have many unique advantages, including dramatically reducing false positives, working in encrypted environments, and the ability to capture new behavior. Though effective, a single honeypot only provide a limited local view of network attacks. Deploying a large number of distributed honeypots in different network domain can provide a broader view. Unfortunately, the more honeypots you deploy, the more resources are required. To address these challenges, Lance Spitzner

presents honeyfarm^[1], a new concept with tremendous potential. It represents one of the newest methods for large deployments of distributed honeypots.

Instead of deploying large numbers of distributed honeypots, simply deploy honeypots in a single and consolidated location. This single network of honeypots becomes a honeyfarm. Once Deployed a redirector to each network you want monitored, the redirector[2] transports an attacker's probes or unauthorized activity to a honeypot within the honeyfarm, without the attacker ever knowing it. Unfortunately, honeyfarms have a great deal of inherent risk in them. The risk being that once attackers take over one of the honeypots within the honeyfarm, they can then use that honeypot to attack other non-honeypot systems. To minimize the risk, each organization must implement data control of a compromised honeypot. Data control defines how activity is contained with the honeynet without an attacker knowing it. The purpose of data control mechanism in the honeyfarm is to prevent attackers and malware using honeyfarm to attack or infect other non-honeypot systems. What's more, with the data control mechanism we can understand the native behavior of malware.

2. Architecture of Honeyfarm

According to the idea of Potemkin virtual honeyfarm[3], we have presented a honeyfarm system mainly consists of honeyfarm gateway and virtual honeypot. Each of the components will be described below.

2.1. Honeyfarm gateway

The honeyfarm gateway supports four distinct functions: it must manage the containment of inbound and outbound traffic, implement network address translation, forward DNS requests, and interface with administration component. Each of the functions will be described below, and the data control will be discussed in Section 3.

2.1.1. DNS proxy

An attacker may be able to detect a honeypot simply by initiating a HTTP GET command, and seeing if it is blocked. In order to avoid detecting the honeypot within a honeyfarm, we must allow outbound DNS requests. The simplest approach to enabling DNS within a honeyfarm is to adjust the containment policy to allow honeypots to send out DNS requests directly to domain name servers on the Internet. However, this will give the potential for malware to attack domain name servers. To address the risk, the gateway allow outbound DNS requests to be forwarded to a dedicated DNS server.

2.1.2. *Network address translation*

A honeypot within a honeyfarm created due to a reflected connection can have any IP address, which most likely one outside the honeyfarm's address space. If a honeypot attempts to initiate a HTTP GET command, response packets should be routed to the real Internet host, not the honeypot within the honeyfarm. We address this issue by implementing network address translation (NAT) at the gateway. When a honeypot created by a reflected connection need to communicate with Internet hosts, we rewrite the source address to be inside the honeyfarm's address space so response packets will reach the gateway.

2.1.3. *Administration interface*

Finally, the honeyfarm gateway provides an administration interface used to maintain the white-list according to monitoring requirements, configure packet modification policy to modify the payload of known attacks, and communicate with intrusion detection system component.

2.2. *Virtual honeypot*

A specialized virtual machine monitor (VMM) spawns a new virtual machine (VM) for each distinct IP address. Once this new VM is ready, it adopts the packet's destination address and handles the request. However, a new VM can incur significant overhead initializing, booting an operating system, and loading application software. If initialization takes too long any inbound connection request may time out. What's more, each VM may consume hundreds of megabytes of memory to represent machine state. Since the host's hardware resources and software resources are limited, creating a large number of VMs would consume all honeyfarm resources.

To reduce the initialization overhead, each host maintains a memory snapshot of a pre-initialized operating system and application environment. When a new VM needs to be created, this snapshot is simply copied, its identity changed to reflect the appropriate network state (IP address, default gateway, and DNS server.). Ideally, a compromised VM would persist long enough for further analysis, logging, or manipulation. To optimize resource, the honeyfarm gateway would reclaim VMs that no longer receiving inbound traffic.

3. **Data Control Mechanism in Honeyfarm**

The best way to implement data control is not to rely on a single mechanism. Instead, implementing data control using several different mechanisms help protect against a single point of failure. We have proposed a data control mechanism that exploits honeyfarm gateway, intrusion detection system, and

reverse firewall to achieve this goal. The data control mechanism is shown in Figure 1.

The honeyfarm gateway implements data control by setting a series of inbound traffic control policy and outbound traffic control policy. Intrusion detection system detects all the inbound and outbound traffic, and makes the use of rules-based network information search mechanism to detect abnormal traffic. In addition, in order to avoid single point of failure, by setting some appropriate rules, the reverse firewall filters outbound connections to ensure that the systems outside the honeyfarm environment will not be destroyed. The following will discuss the inbound traffic control policy and outbound traffic control policy of the honeyfarm gateway, the abnormal flow detection of the intrusion detection system, and the outbound connection filtering of the reverse firewall respectively.

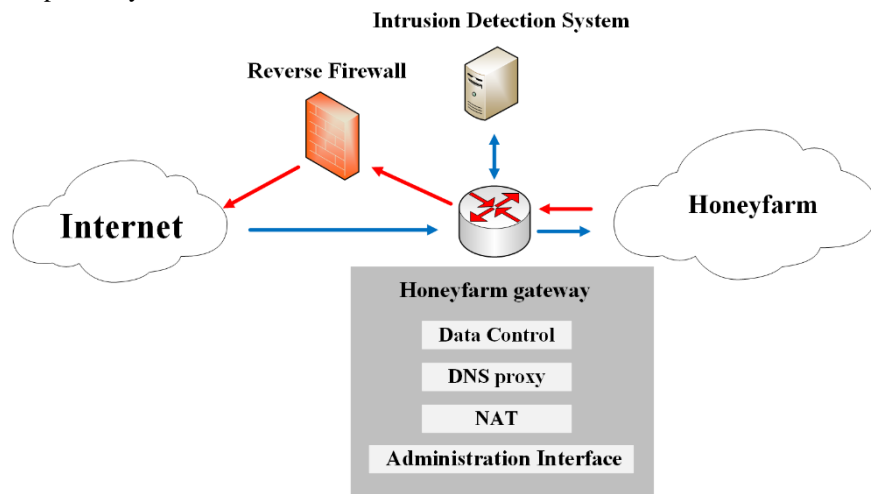


Fig. 1. Data control mechanism in the honeyfarm.

3.1. Inbound traffic control policy

3.1.1. Packet filtering policy

Honeyfarm is a service-oriented architecture, and it provides services for the monitored networks. In order to prevent the honeyfarm from being damaged, the honeyfarm gateway only receive the traffic from the monitored IP addresses and the traffic responding to the honeypots. For the traffic that does not belong to both, the honeyfarm gateway would filter it.

3.1.2. *Packet modification policy*

Honeypot's value lies in being compromised to observe and study attacks. In order to learn zero day vulnerabilities and new methods used by attackers, the honeyfarm gateway can be configured to modify the payload of known attacks to disable it. Attackers will see their attacks reaching their intended targets, but not be able to figure out why the attacks are failing. They may try different means to attack the honeypot, which maximizes the value of the honeypot.

3.1.3. *Packet dispatch policy*

Packets go through the packet filtering policy and the packet modification policy are prepared to be delivered to final destination. The honeyfarm gateway dispatches packets to VMM that will be responsible for creating a VM to represent the destination IP address of the packet, and the VM handles the requests as though it were the intended recipient.

3.2. *Outbound traffic control policy*

3.2.1. *Packet forwarding policy*

When an attacker breaks into a honeypot, they may initiate connections (ICMP ping or a simple HTTP GET command) to detect a honeypot. To address this challenge, the honeyfarm gateway forward normal traffic to the Internet.

3.2.2. *Packet encapsulation policy*

The source address of the packets forwarded by the redirector called the attack source, under the assumption that this attack source is a central server or bot-master. In order to learn the native behavior of malware, the honeyfarm gateway must response to the attack source. If the honeyfarm gateway forwards the responding packets to the attack source directly, an attacker may be able to detect a honeypot simply by checking the source address of the responding packets. Thus, these packets are encapsulated and forwarded to the redirector by the honeyfarm gateway, and the redirector forwards them to the attack source.

3.2.3. *Packet reflection policy*

Once a honeypot is successfully compromised, it may attempt to attack or infect non-honeypot systems. Thus, a honeyfarm could easily become an accelerator for a malware. To address this issue, the honeyfarm gateway uses packet reflection policy. When the IDS identifies an outbound packet cannot be safely

forwarded to the Internet, the honeyfarm gateway can reflect it back into the honeyfarm which will then adopt the identity of the destination IP address.

3.2.4. Packet dropping policy

In order to hide themselves, the network attackers often break some systems and make them as bots instead of attacking the target through their own systems. Attackers need to conduct a wide range of network scanning to implement the springboard attack. In this case, the compromised honeypot systems will generate a large number of scanning packets with different IP addresses. If the packet dropping policy is not set, the honeyfarm gateway will create honeypots for each scan packet, which will result the depletion of hardware resources in the honeyfarm. To solve this problem, the honeyfarm gateway maintains a list of honeypots that are generated directly or indirectly by each source in the honeyfarm. If the number of generated honeypots exceeds a certain limit, the subsequent packet will be dropped.

3.3. Abnormal flow detection

Abnormal flow detection is implemented by the Intrusion detection system (IDS). The IDS inspects all the inbound and outbound traffic.

For inbound traffic, the purpose of IDS is to identify known attacks and forward them to the honeyfarm gateway. If any packet matches any of the IDS rules, the packet can be modified by the honeyfarm gateway according to the packet modification policy. For outbound traffic, the purpose of IDS is to distinguish traffic with known attacks. The attack traffic would be reflected back into the honeyfarm according to the packet reflection policy, and the normal traffic would be forwarded.

3.4. Outbound connection filtering

Outbound connection filtering is implemented by the reverse firewall. The reverse firewall checks all the packets transmitted from the honeyfarm to the external network and restricts all packets that do not meet the security policy requirements.

In order to avoid a single point of failure, the reverse firewall implement the final layer of data control. By setting the corresponding rules to filter outbound connections to prevent denial of service attacks, distributed denial of service attacks and other security threats. In addition, we can set the threshold to limit the number of outbound connections for further outbound traffic control.

4. Implementation of Data Control Mechanism

The honeyfarm gateway is built on top of the Click modular software router framework[4]. A router implemented in Click consists of a set of packet processing modules called elements. Using a special description language called configuration, elements can be connected together to form a directed graph that represents how packets flow through the processing modules. Upon the base Click installation, our implementation adds roughly 8000 lines of custom element code and nearly 1000 lines for configuration. Intrusion detection system uses network intrusion detection system Snort[5]. The reverse firewall uses the Linux kernel-based firewall iptables[6].

4.1. Inbound traffic control

The inbound traffic is divided in two parts, one is forwarded by the redirector, and the other is responding to the honeypots. The control process is shown in Figure 2.

The honeyfarm gateway maintains a whitelist that records the all the IP addresses of the monitored network, and a non-attack address list that records the normal traffic destination addresses. In order to avoid detecting and damaging the honeyfarm gateway, it only receive traffic from the whitelist and the non-attack address list.

For the traffic that is forwarded by the redirector, the honeyfarm gateway decapsulates the packet and forwards to the IDS. Once identified known attacks, the honeyfarm gateway decides whether to modify or forward it. Finally, virtual machine monitor will create a VM to represent the destination IP address of the packet, and subsequent packets to the same IP address can then be delivered directly to that VM. For the traffic that is responding to the honeypots, the honeyfarm gateway uses destination address network address translation and dispatches it to the corresponding honeypot.

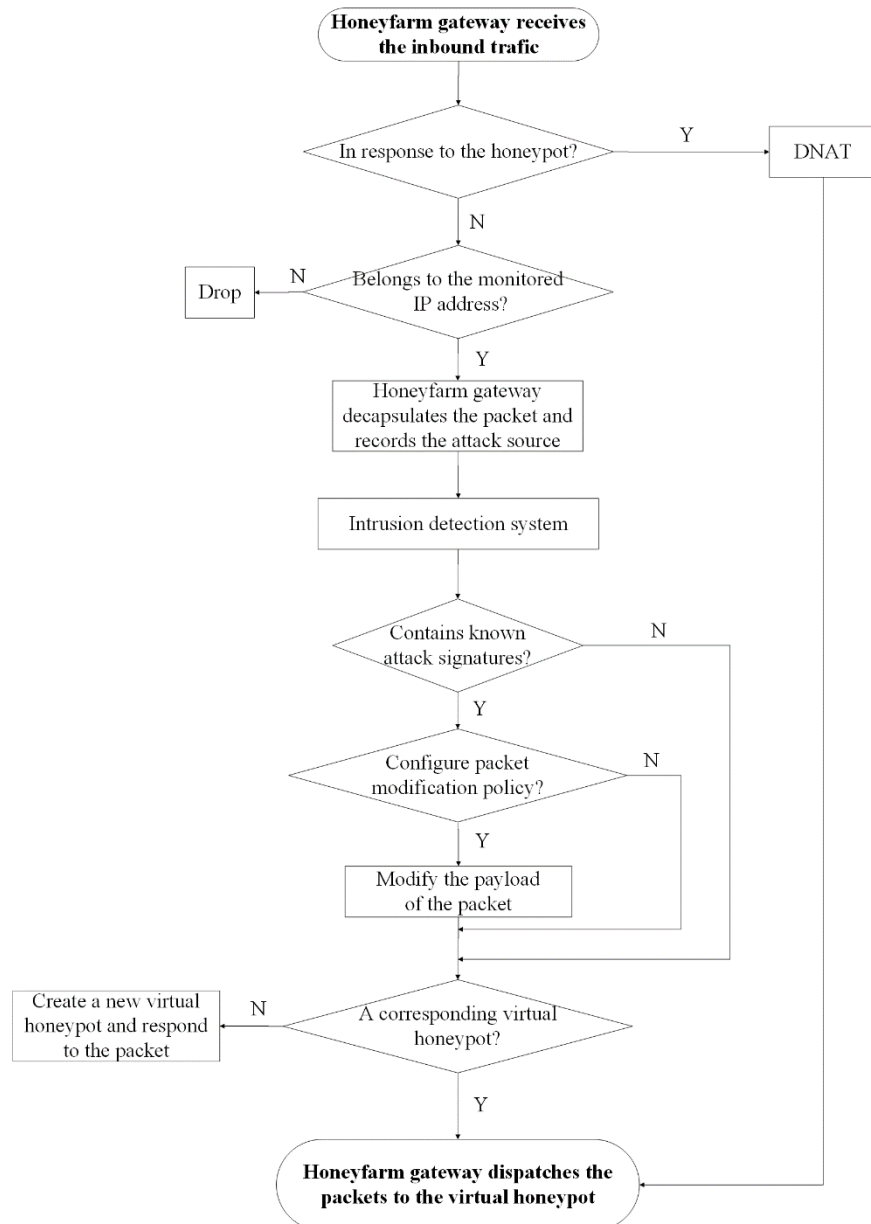


Fig. 2. Inbound traffic control process.

4.2. Outbound traffic control

The outbound traffic is divided in two parts, one is the responding traffic to the attack source, and the other is the traffic initiated by the honeypots. The control process is shown in Figure 3.

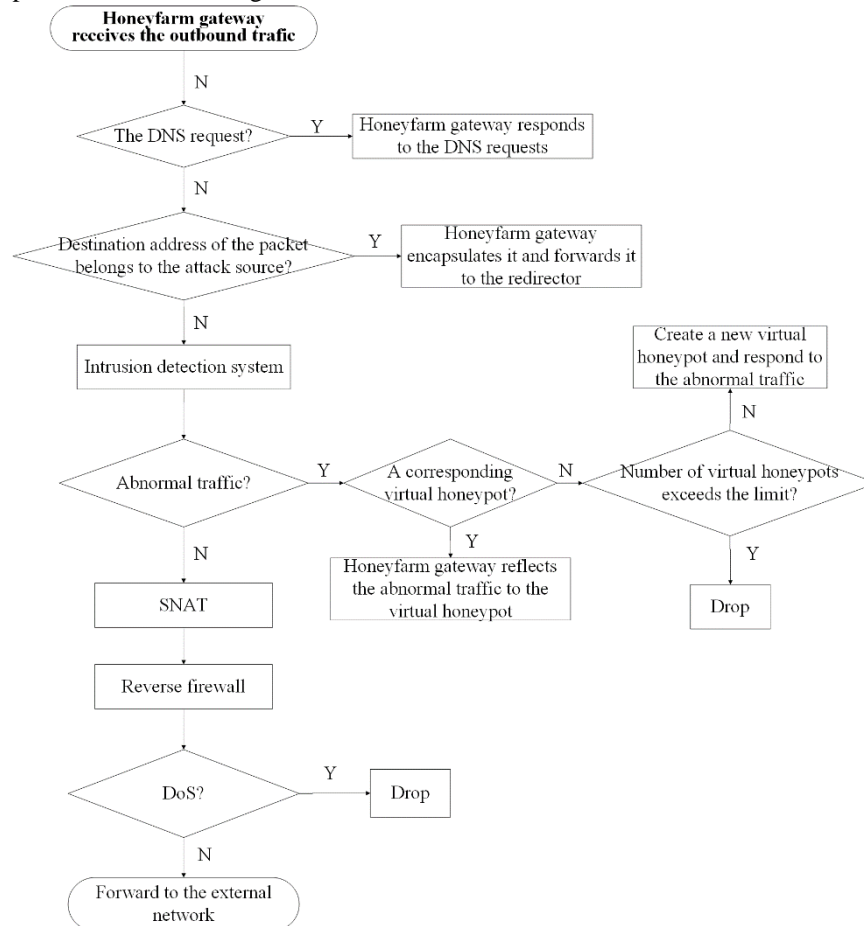


Fig. 3. Outbound traffic control process.

The honeyfarm gateway maintains a flow table that records the attack sources, and a history table that records the state of instantiated honeypots.

For the traffic responds to the attack source, the honeyfarm gateway encapsulates the packet and forwards to the corresponding redirector. After that, the redirector decapsulates it and forwards to the attack source. For the traffic initiated by the honeypots will first be detected by Snort to distinguish abnormal traffic. The abnormal traffic that cannot be safely forwarded to the Internet

would be reflected back by the honeyfarm gateway. Packets with attacks track the IP address of honeypots via the history table so that an outbound packet is subject to the packet reflection policy. However, due to the limited hardware resources of the honeyfarm, the honeyfarm gateway will drop the subsequent packets when the number of honeypots exceeds a certain value. Finally, the normal traffic will be forwarded to the reverse firewall for further filtering.

5. Evaluation

In this section, we evaluate the validity of our data control mechanism in the honeyfarm.

5.1. Experimental environment

Our experimental test environment uses a Dell PowerEdge R330 server with a 3GHz Xeon processor and 8GB of physical memory. The honeyfarm gateway is based on the Click 2.0.1 version running in kernel mode. VMM is based on a version of Xen 3.0 and we use Debian GNU/Linux 5.0 as Xen guest operating system for creating the honeypots. Snort and iptables are deployed in the RedHat Linux 9 respectively.

We have monitored five public IP addresses, five honeypots are installed in the honeyfarm to monitor those IP addresses. The operating system and service configuration of the honeypots are shown in Table 1. Monitoring began in May 2016 and lasted for five months.

Table 1. Honeypots information and service configuration.

Honeypot information	Honeypot service configuration
Unpatched Ubuntu 10.04	FTP, SSH, Telnet, SMTP, HTTP, RPC
Unpatched Windows XP SP3	MSRPC, NetBIOS-SSN, Microsoft-DS
Unpatched Windows Server 2003	FTP, HTTP, RPC, NetBIOS-SSN
Patched Windows 7	MSRPC
Patched Windows 8	MSRPC

5.2. Experiment analysis

5.2.1. Comparative analysis

Due to the different types of operating systems, and the variety of open services, the Ubuntu 10.04, the Windows XP SP3 and the Windows Server 2003 which provide more services always attract more attacks. As shown in Figure 4, we record the number of attacks in 30 days without the packet modification policy in the honeyfarm gateway. After configuring the packet modification policy, we record the number of attacks in the next 30 days. The number of attacks on each

honeypot is less than before, indicating that the packet modification policy can effectively modify the attack signatures in the inbound traffic.

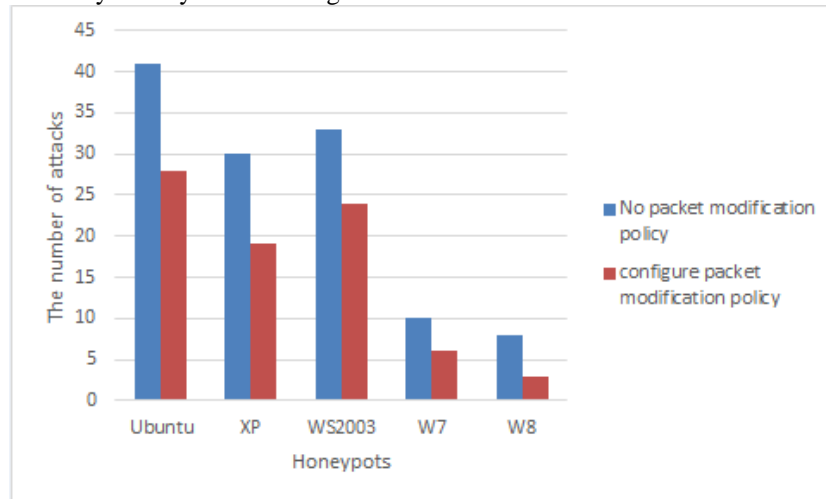


Fig. 4. The number of captured attacks.

An attacker will attempt to use a honeypot as a springboard to attack other product systems when he has compromised it. The most common form of attack is to use the compromised honeypots to initiate a network scan. As shown in Figure 5, we stopped the intrusion detection system between July 8 and August 7. After deploying the intrusion detection system in August 8, the number of honeypots in the honeyfarm is increased markedly, indicating that the packet reflection policy can reflect the abnormal traffic identified by the intrusion detection system back to the honeyfarm.

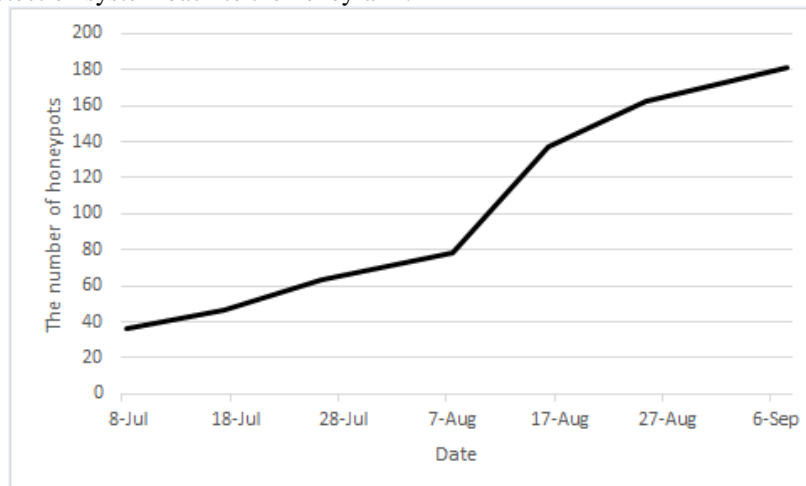


Fig. 5. The number of honeypots.

5.2.2. Worm analysis

During the last 5 months of monitoring, our honeypots captured Flame, Morto, Blaster and some other worms, we use the honeyfarm to analysis the propagation behavior of those worms. Morto spreads the fastest, takes about 7 seconds to infect the next honeypot, while Flame and Blaster spreads relatively slowly, infecting the next honeypot takes about 30 seconds. Morto uses the remote desktop protocol (Remote Desktop Protocol, RDP) to obtain the target remote desktop access, because the RDP password of honeypots is empty, so Morto spreads fast. Blaster spreads through Remote Procedure Call (RPC) vulnerabilities, which are slow to propagate due to the length of the vulnerability scanning and penetration process.

By configuring the honeyfarm gateway, one attack source can generate up to 128 honeypots. Thus worms have freedom to infect other honeypots in the honeyfarm. Figure 6 shows the worm propagation behavior, we found that the propagation time of all worms is exponential. Through statistical analysis, Morto first infected 128 honeypots, it takes about 50 seconds.

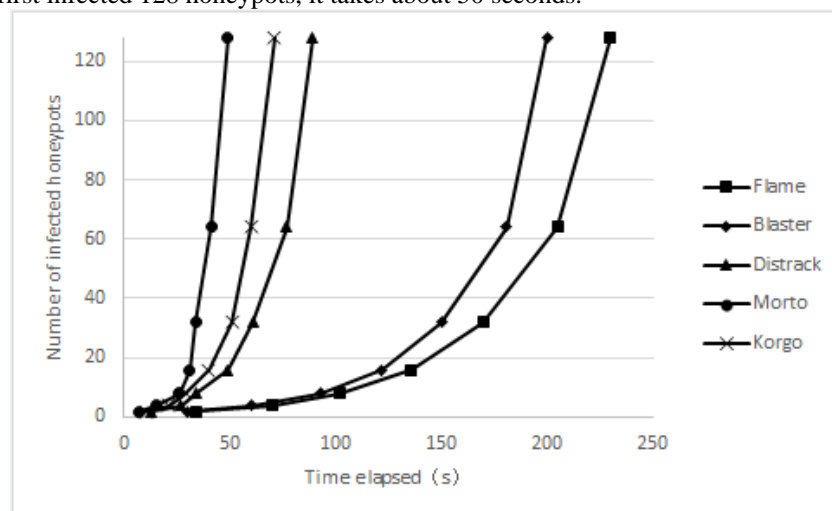


Fig. 6. Worm propagation behavior.

5.2.3. Delay analysis

Deployment of the intrusion detection system and the reverse firewall will inevitably lead to network latency. We use a honeypot to construct a TCP source, while using an Internet host to construct a TCP receiver, and the receiver has two hops from the source. The TCP source sends a 4.2GB video file to the TCP receiver repeatedly. We increase the TCP packet size from 400 bytes to 1400

bytes and measure the end-to-end delay in the TCP source for 10 times. The average result is shown in Figure 7.

As the honeyfarm gateway will pass packets to the intrusion detection system and the reverse firewall for processing, both of them will lead to the delay. Intrusion detection system will match packets with attack signatures, while the reverse firewall simply compares the header of packet with its configuration rules. So the intrusion detection system produce more delay than the reverse firewall. As we use a gigabyte link to connect the TCP source and the TCP receiver, so there is no significant difference in the transmission time between 400 bytes and 1400 bytes. In addition, the TCP receiver is only two hops away from the TCP source, as a result, the end-to-end delay variation of the different size is not significant.

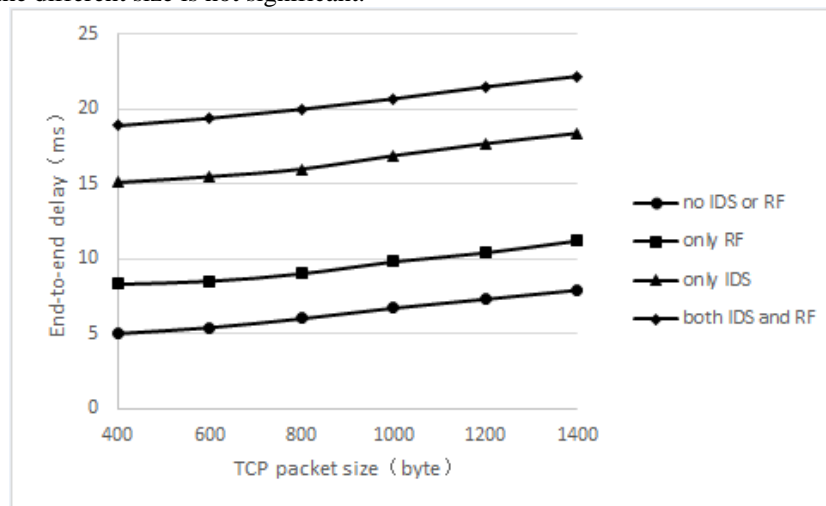


Fig. 7. End-to-end delay.

6. Conclusion

In this paper, we have presented a data control mechanism that exploits the honeyfarm gateway, intrusion detection system, and reverse firewall to minimize the risk by a compromised honeypot within the honeyfarm. We evaluate our mechanism implementation, and argue that it is effective to identify the outbound traffic with attacks, which dramatically reduces the risk of a known outbound attack being successful. What's more, by implementing our mechanism, we are able to safely study and analyze the behavior of malware without the possibility of it leaking out onto the Internet.

Acknowledgments

This work was supported by the Key Project of National Defense Basic Research Program of China under Grant No. JCKY2016602B001 and the Equipment Advance Research Foundation of China under Grant No. 9140A15070415BQ01213.

References

1. Lance Spitzner. Honeybot Farms. 2003. <http://www.symantec.com/connect/articles/honeybot-farms>
2. Xuxian Jiang, Dongyan Xu. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *Proceedings of the USENIX Security Symposium*, August 2004.
3. Michael Vrabie, Justin Ma, Jay Chen, et al. Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm. In *Proceedings of the 20th ACM Symposium on Operating System Principles (SOSP)*, October 2005.
4. Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, M. Frans Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems (TOCS)*, August 2000.
5. Snort. <https://www.snort.org/>
6. IPTables. <https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html>
7. Christian Kreibich, Nicholas Weaver, Chris Kanich, et al. GQ: Practical Containment for Measuring Modern Malware Systems. In *Proceedings of the ACM SIGCOMM on Internet Measurement Conference*, October 2011.
8. Pragya Jain, Anjali Sardana. Defending Against Internet Worms Using Honeyfarm. In *Proceedings of the CUBE International Technology Conference*, September 2012.
9. Sanjeev Kumar, Paramdeep Singh, Rakesh Sehgal, et al. Distributed Honeybot System Using Gen III Virtual Honeybot. *International Journal of Computer Theory and Engineering*, 2012.
10. Zhenxin Zhan, Maochao Xu, Shouhuai Xu. Characterizing Honeybot-Captured Cyber Attacks: Statistical Framework and Case Study. *IEEE Transactions on Information Forensics and Security*, 2013.
11. I.S. Kim, M.H. Kim. Agent-based honeybot framework for protecting servers in campus networks. *IET Information Security*, 2012.