AN IMPLEMENTATION FOR A WORM DETECTION AND MITIGATION SYSTEM

Hamad Binsalleeh and Amr Youssef

Concordia Institute for Information Systems Engineering
Concordia University,
Montreal, Quebec, Canada, H3G 2W1
{*h_binsal,youssef*}*@ciise.concordia.ca*

ABSTRACT

In this paper, we present an integrated system for the detection and mitigation of zero-day scanning and mass mailing worms. The detection engine of our system utilizes the Domain Name System (DNS) anomalies of the worm traffic; an idea that has been noted by several security researchers. Once a worm is detected, the firewall rules are automatically updated in order to isolate the infected host. An automatic alert is also sent to the user of the infected host. The system can be configured such that the user response to this alert is used to undo the firewall updates and hence helps reduce the interruption of service resulting from false alarms. The developed system has been tested with real worms in a controlled network environment. The obtained experimental results confirm the soundness and effectiveness of the developed system.

1. INTRODUCTION

An Internet worm[1] is a self-propagating malware program that automatically replicates itself to vulnerable systems and spreads across the Internet. These worms propagate themselves by exploiting a security vulnerability in certain versions of service software to take control of the victim machine and copy themselves over to other vulnerable machines. Unlike computer viruses that typically spread from one computer to another by attaching themselves to either data files or executable applications (and hence their spread is limited by the speed by which these infected files can be transmitted from one system to another), worms, in contrast, are capable of autonomous migration from one system to another via the network without the assistance of external software.

Typically, a worm-infected host scans the Internet for vulnerable systems. It chooses an IP address, attempts a connection to a service port, and if successful, lunches the attack. The above process repeats with different random addresses. As the number of compromised machines increases, more copies of the worm can work together to reproduce themselves. Since the hosts that are vulnerable to a worm typically account for a small portion of the IP address space, worms rely on high-volume random scan to find victims. Thus, the scan traffic from tens of thousands of compromised machines can congest networks and an explosive epidemic can therefore be developed across the Internet causing a Denial-of-Service (DoS). Although most known worms did not cause severe damage to the compromised nodes, they could have altered data, removed files, stolen information if they had chosen to do so.

Internet worms can be classified into: vulnerability worms, mass-mailing worms, instant messaging worms, and peer-to-peer worms [1, 2]. Irrespective of its type, an Internet worm is usually composed of several components, which collaborate together to achieve the goal of the worm designer. In particular, every worm contains two main components: the target locator module and infection propagator module. Some other optional components such as remote control, update interface, life-cycle manager, and payload routines can be used according to the worm function during its life cycle.

In the following section, we explain some Domain Name System (DNS) anomalies associated with the traffic generated by vulnerability worms and mass-mailing worms [3, 4, 5].

2. DNS ANOMALIES OF WORMS

A typical user usually uses human readable domain names for accessing various services on the Internet. These domain names have to be translated to its associated IP addresses before initiating any communication. This translation process is initiated once a network node requests DNS service for domain name lookup. In response to this request, the DNS server responds by sending a DNS response message, which contains a complete description of the domain name stored in the DNS server database from which the lookup has been requested. Once the DNS response message is received, the requesting host retrieves the desired IP address and then starts its communication with its target host using the retrieved IP address.

On the other hand, scanning worms do not contact DNS servers for domain name mapping. Instead, to locate their victims,

[1]The term worm was casted by John Brunner in 1975 in a scientific fiction novel entitled: *The Shockwave Rider.*

these worms generate IP addresses directly by themselves without consulting any DNS server. Thus, and as noted in [4], the presence of new connection requests that are not preceded by DNS queries can be considered as a sign of worm infection. Other legitimate connections that can be initiated by benign hosts without any prior DNS queries (e.g., embedded IP addresses hard coded inside HTTP, VoIP and instant messages protocols, IP addresses of servers running inside the local network, and reserved IP addresses that are used publicly over the Internet as well as within local networks for broadcasting and multicasting) have to be identified as legitimate and their associated IP addresses have to be added to the safe list.

Similarly, the procedure used by mass mailing worms to deliver their malicious emails is different from the normal procedure followed by the normal email communication protocols [3, 5]. Normally, each host that wants to send an email has to communicate with the mail server responsible for delivering emails to its intended recipients. Mail servers in turn communicate with DNS servers for determining the target recipient email servers. A special type of communications has to take place between the mail server and the DNS server. Mail server has to initiate Mail eXchange DNS query (MX query), which contains the fully qualified domain name. Afterwards the mail server sends this query to the DNS server in order to retrieve the IP address of the mail server that is responsible for the target host mail box. Then, the mail server starts communicating with the remote mail server to deliver the email. In contrast to the above procedure of normal email delivery, mass-mailing worms usually use its own SMTP engines to bypass local email servers security measures. However, they still rely on the DNS servers for locating the respective mail servers of their intended victims. Creating host-based MX requests is a violation of the typical communication pattern because these requests are supposed to only take place between mail servers and DNS servers.

3. SYSTEM MODULES

Fig. 1 shows the five main components of our worm detection and mitigation system. In this section, we explain the main functionality of each one of these components and illustrate its interaction with the other components.

3.1. Traffic Sniffing Module

The main objective of this module is to capture all the local area network (LAN) traffic and log it, in real time, to the detection system database. In our implementation, we used Snort Network Intrusion Detection System (NIDS) for implementing this module. By configuring Snort in the sniffing mode, we are able to collect all the network packets and log them, using libpcap, into TCP dump file format.

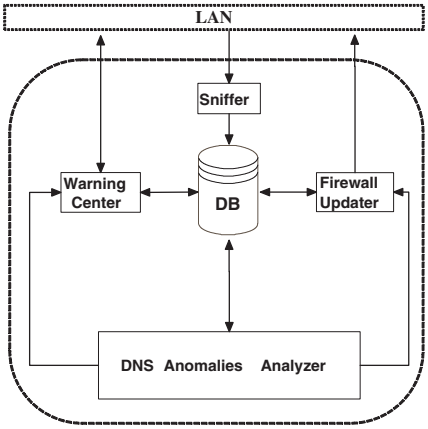

Fig. 1. Main components of the proposed system

3.2. Database Module

In order to have a scalable and flexible packet analysis module, the sniffed LAN traffic is stored inside a database. We used the database design schema of Snort system to interact with the collected network traffic. This provides an efficient representation of network traffic and helps us to analyze the stored data efficiently. The stored database tables are then managed using MySQL open source Database Management System (DBMS).

3.3. DNS Anomalies Analysis Module

Following the same architecture proposed in [4], this module has been decomposed into two components that collaborate together to perform the process of detection and analysis of network traffic: Packet Processing Unit (PPU) and DNS Correlation Unit (DCU). The PPU has two main functions: building a local DNS cache, and extracting embedded IP addresses. Each DNS cache candidate consists of 4 tuples including source IP address of the DNS record, the domain name that is recorded in the request, the reply of the query, and the Time To Live (TTL) attribute that indicates when we have to discard the current DNS record. The process of extracting embedded IP addresses is achieved by applying regular expressions functions to each packet payload in order to match and retrieve any numeric IP address representation within a string. The DCU is responsible for validating each new connection to check whether it is legitimate or not. It takes the PPU processed information, i.e., the DNS cache list, the embedded IP addresses list, and the white list. Then, it retrieves every new network connection from the database and compares the source and the destination IP addresses with every single entry inside all the legitimate lists. If any matching entry is found, the DCU skips the current connection and starts analyzing subsequent new connection requests. Otherwise, the DCU identifies this connection as illegitimate and adds it to the malicious activities

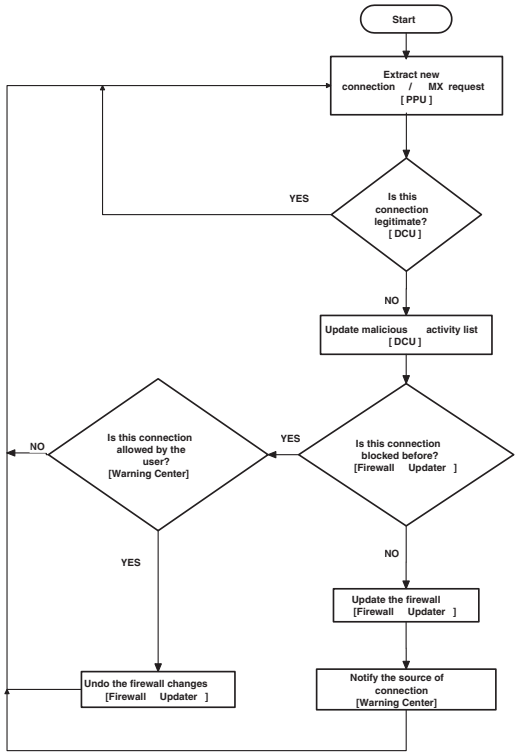

Fig. 2. Worm detection process

list. Furthermore, the DCU module sends a request to the firewall management center to block this connection. It also sends another request to the warning center to notify the user of the infected host. Similarly, the DCU identifies hosts infected by mass mailing worms by locating any MX queries that are initiated from hosts which are not authorized to issue these kinds of queries.

3.4. Warning Center

This module is responsible for alerting end-users about any suspicious activities that originate from their systems. When using our worm detection system, an additional software component is installed on each end-user host. The warning system interacts with these software components using UDP connections. Whenever a connection is flagged as suspicious by the worm detection system, the warning system extract the source IP address and port number, the destination IP address and the port number, protocol, and the timestamp of this connection. Then the warning center checks the history of notification messages for the same connection. Only connections that do not have records within the notification history generate new alarms.

3.5. Firewall Updater

The objective of this component is to automatically mitigate the effect of worms by isolating the infected machines (or subnets). Whenever a connection is flagged as malicious, the firewall is automatically updated to block all future connections originating from the suspected host. The firewall updater module is connected by Secure Shell (SSH) protocol directly to the firewall system and is responsible for updating the IP table rules. The system can be configured such that the user response to the alert sent by the warning center can be used to undo the firewall updates and hence helps reduce the effect of false alarms. Fig. 2 shows the interaction between the various system components during the worm detection process.

4. EXPERIMENTAL RESULTS

In order to test the effectiveness of the proposed system, we carried out two sets of experiments. The objective of the first set of experiments, which was carried out without any real worms, is to identify the causes of false positives, i.e., the cases in which a legitimate user connection is identified by the worm detection system as being generated by a worm. The objective of the second set of experiments, which was conducted using some real worms in a controlled network setup, is to test the robustness of the overall system, i.e., to verify that the developed detection system, including both the automatic firewall updater and warning center, can not be over run by the scanning speed of modern Internet worms. It should be noted that our system decision is based on deterministic observations. Hence, by design, false negatives do not exist. Fig. 3 shows the network setup used throughout our experiments. It consists of two different network segments that are separated by a network router. In the first segment, as shown in the lower part of the figure, we have a set of seven desktop computers[2] that are linked together by an Ethernet hub. Our detection system is part of this segment. On the other side, we have the DNS server, which is configured to function as a local DNS for the first segment. The LAN and the DNS server are connected to the university network via

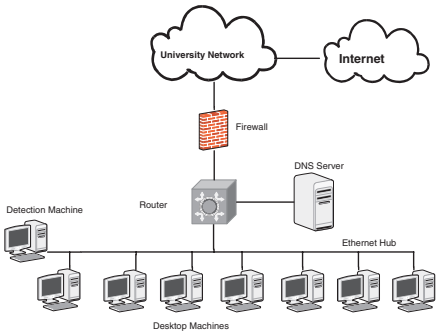

Fig. 3. Experimental network setup

[2]These computers are the normal workstations used by graduate students to carry out their daily research activities.

a firewall. The firewall is configured using Linux system for controlling the outgoing and incoming data flow. Our test network is then connected to the Internet through the university network. The detection machine, which runs the DNS analysis engine, continuously monitors all the traffic that is generated or received by any local system within the test network. The DNS server is responsible for providing the DNS services for all systems inside the LAN. The firewall system is a configurable software program that is implemented inside a Linux system, which can be controlled remotely by our detection system in such a way that allows new rules to be added and the existing ones to be modified dynamically.

We have run our system for 17 days without interruption. Before starting the experiment, all the laboratory machines were restarted and the DNS cache list within every system was flushed to make sure that no DNS queries have been resolved before starting the experiment. Fig. 4 shows the total number of new connections that originate from the workstations during the test interval where all connections with the same source and distinction IP address are counted once. If a connection with a given target IP address was observed in a given time, it is not counted again if it was observed in any subsequent time. From the results above, it is

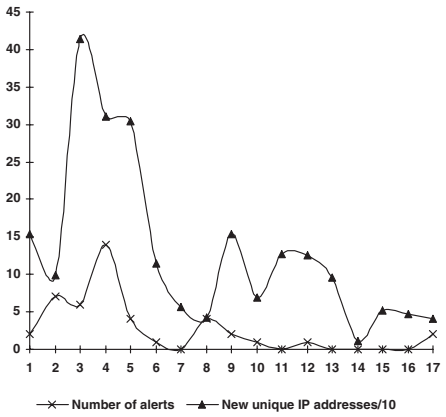

Fig. 4. Number of unique observed connections and number of false alarms

clear that number of false positives can be greatly reduced by carefully designing the safe lists. The network setup used throughout the second set of our experiments also consists of two different network segments that are separated by a network router. In the first segment, three desktop computers were infected by three Internet worm: two of them are scanning worms (W32.Sasser and W32.Blaster,and the third one is a mass mailing worm W32.Netsky). The firewall is configured to block all the traffic originating from these infected machines. While Sasser and Blaster generated about 41150 and 36978 scans per day, respectively, Netsky resulted in about 59441 MX queries. All these anomalies were successfully captured by our system and the infected host was isolated from the first connection attempt.

5. CONCLUSION AND FUTURE WORKS

Throughout this work, we have developed an integrated worm detection and mitigation system. The detection algorithm is based on the observation that DNS activities associated with both scanning worms and mass mailing worms violate the typical communication pattern followed by normal users. Similar to [4, 5], our system can detect all zero-day scanning and mass mailing worms that cause the prescribed DNS anomalies and it can detect slow as well as fast scanning worms after only a single malicious worm connection attempt. It should be noted that, unlike [4, 5] where the authors performed offline processing of TCP dump, our system runs in real time. In addition to this, the automatic firewall update feature mitigates the damage that can result from waiting for a manual action to be performed by system administrators and isolate infected hosts or subsets. Moreover, its user friendly GUI is carefully designed to facilitate both the system configuration and the monitoring processes. Careful investigation of different Internet protocols can lead to discovering other anomalies associated with worm activities and hence discovering larger families of worms. Also, integrating the developed system with other statistical based systems will result in lowering the number of false alarms and improving the detection capabilities of the system. Mitigating the effect of irresponsible users that always confirm that the suspected connection attempt originating from their machine are legitimate (either because of their negligence, ignorance or because they do not want to interrupt their work) is an interesting multidisciplinary research problem. In order to reduce effect of false alarms on the continuity of service to the suspected LAN users, techniques such as virus throttling can be applied before totally isolating suspicious hosts and subnets.

6. REFERENCES

[1] J. Nazario, *Defense and Detection Strategies against Internet Worms*. Norwood, MA, USA: Artech House, Inc., 2003.

[2] P. Szor, *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.

[3] Y. Musashi, R. Matsuba, and K. Sugitani, "Indirect detection of mass-mailing worm-infected PC terminals for learners," in *ICETA2004*, 2004.

[4] D. Whyte, E. Kranakis, and P. van Oorschot, "DNS-based detection of scanning worms in an enterprise network," in *Network and Distributed System Security Symposium (NDSS'05)*, February 2005.

[5] D. Whyte, P. van Oorschot, and E. Kranakis, "Addressing SMTP-based mass-mailing activity within enterprise networks," *Computer Security Applications Conference*, vol. 22, no. 06, pp. 393 – 402, December 2006.