SAFEGUARDING THE PROPAGATION OF FAST SCANNING MALWARE

Abstract: This paper presents a detection and containment mechanism for fast self-propagating network worm malware. The detection part of the mechanism uses two categories of network host activities to identify worm behaviour in a network. Upon an identified worm activity in a network, a data-link containment system is used to isolate the internal source of infection, and a network level containment system is used to block inbound worm datagrams. The mechanism has been demonstrated using a software prototype. A number of worm experiments have been conducted to evaluate the prototype. The empirical results show the effectiveness of the developed mechanism in containing fast network worm malware at an early stage with almost no false positives.

Keywords: Cyber defence; network worm; worm containment; Malware.

1. Introduction
The Internet has provided a medium for communication and sharing of information amongst people, businesses, governments and organisations. Therefore, the Internet must be kept continuous and secured from any form of malicious activities such as unauthorised access to computer system and malware attacks. Malicious software (malware) is a generic term for any software that enters a computer system without the authorisation of the user to perform unwanted actions [1]. A network worm is a malicious software program that propagates across a network by infecting hosts and in some cases launching malicious activities. A scanning network worm propagates by probing pseudo-random addresses looking for vulnerable hosts, which makes the malware highly virulent in nature. Fast scanning network worms are a particularly dangerous sub-class of scanning worms due to the speed at which they propagate across the Internet.

This paper presents a cross-layer detection and containment technique termed SAFE (Safeguard Against Fast Self-propagating Malware), which is an advancement over the mechanism reported in [1]. The rest of the paper is organised as follows. Section 2
summarises related work on detection and containment of network worm malware. Section 3 presents a description of the SAFE algorithm. Section 4 presents the procedure used to evaluate the SAFE mechanism. Section 5 presents the evaluation experiments conducted and the results obtained. Section 6 presents a discussion of the evaluation results. Section 7 concludes this paper and presents possible future work.

2. Related Work
A number of anomaly-based network intrusion detection systems have been developed to identify the presence of worms using datagram header information and payload information.

Gu et al. [2] developed an algorithm, termed DSC, that correlates incoming and outgoing traffic, i.e., if a host received a datagram on port $i$, and then starts sending datagrams destined for port $i$, it becomes a suspect. Jung et al. [3] proposed an algorithm, termed TRW, which identifies a remote host attempt to establish a new TCP connection to a local destination as normal if there is a corresponding TCP reply. On the other hand, failure to establish a successful TCP connection is considered suspicious. Weaver et al. [4] simplified the TRW scheme by considering all new connections to be a failure until a response is received. The algorithm drops a datagram if it does not match an existing and successfully-established connection after a predefined threshold count. Whyte et al. [5] used DNS-based rate limiting to suppress scanning worms in an enterprise network by identifying the absence of DNS resolution before a new connection as anomalous. Shahzad and Woodhead [6] proposed a scheme that uses the absence of DNS lookup action prior to an outgoing TCP SYN or UDP datagram to a new destination IP address to detect worm propagation, and a protocol termed Friends to spread reports of an identified worm event to potentially vulnerable and uninfected peer networks within the scheme. Li and Stafford [7] proposed a worm detector, which they termed SWORD. SWORD comprises two main modules; a Burst Duration Detector (BDD) and a Quiescent Period Detector (QPD). The BDD module comprises a burst detection algorithm, which creates a window for every different size of first-contact connections for detecting fast scanning worm. The QPD module monitors quiescent periods in network activity to ensure that they do not disappear due to constant worm scanning. These techniques consume resources in order to keep track of distinct connection and host information, especially in large networks and they can only slow worm infections [1].

PAYL to detect and generate signatures for zero-day worms. PAYL uses a training phase to create a profile during normal operation, and produces a byte frequency distribution as a model for normal payloads. Based on this information, a centroid model is created and then during the detection phase, the Mahalanobis distance of each datagram payload from the centroid model is calculated. A datagram is considered to be anomalous based on its distance from the normal behaviour. Kim et al. [9] proposed a detection scheme using a standalone device. During the training phase, the mean and standard deviation scores for all datagrams are computed. In the detection phase, a score is computed by counting the number of datagram bytes that fall outside the range defined for each byte. These mechanisms have limitations such as computational complexity, management overhead, high rates of false positives [1] and incur significant delays in deployment and detection [9].

3. The Safe Algorithm
The SAFE mechanism comprises a network layer detection system and a containment system at the data-link layer that work together to provide a countermeasure solution, with a connection maintained between the two systems. The detection system detects anomalies from client hosts and server hosts in a network using different techniques. Client hosts, such as workstations, laptops, tablets and smart-phones, are defined as network hosts which typically consume Internet services, while server hosts, such as web servers and email servers, are network hosts used to serve client requests. The detection system monitors inbound and outbound TCP SYN and UDP datagrams for a window of time with value $T$. This is to determine anomalies that exceed a threshold, which is a maximum allowable count of anomalous datagrams a host can send before $T$ has elapsed. The containment system uses the MAC address of an identified infected host by the detection system to block all traffic originating from the host using MAC address access control.

For client hosts, the detection algorithm observes DNS resolution datagrams and records the IP address of a host that made the resolution and the resolved address in the resolution table. The destination IP address and port of an inbound datagram (excluding a DNS reply) is recorded in the inbound table for both client and server hosts. Outgoing datagram header information (source IP addresses and ports) is associated to entries in an exempt table. The exempt table comprises a list of IP addresses and ports that are exempt from the algorithm. If the header information results in a miss, the algorithm determines whether there is a recent DNS query by a client host for the destination IP address prior to sending the datagram by checking the resolution table.
If there is a miss, the algorithm records the destination port in the no-resolution table, increments its counter and then determines excess using the threshold with value $V$. For server hosts, the algorithm checks the presence of the destination port of outbound TCP SYN and UDP datagrams in the inbound table. If there is a hit, a counter for such entry is incremented for TCP datagrams. An additional verification of the destination IP address is made to determine a reply UDP datagram, and if the destination IP address does not match the IP address recorded for such entry in the inbound table, its counter is incremented and then excess in threshold is also determined. Upon a host exceeding the set threshold, the algorithm invokes the containment system and then checks the presence of the suspect port in the inbound table. If there is a hit, an additional countermeasure is applied at the network layer using an access control list (ACL) to block all inbound datagrams destined for the suspect port in the network segment. A time-to-live (TTL) is provided for entries in all the caches. The default TTL value for DNS (86400 seconds) is applied to the resolution cache and 60 seconds is applied to the no-resolution and inbound caches. Furthermore, the algorithm decrements the counters in the no-resolution and inbound tables by half after the expiration of a timing window of $T$, and then checks all caches to determine and remove entries with expired TTL values.

4. Evaluation Procedure

To evaluate the proposed mechanism, a software prototype was developed and tested using worm propagation experiments in a controlled environment. The SAFE mechanism was tested along with two previously reported worm detection techniques namely DSC [2] and DNS-based detection schemes [5, 6]. The schemes were also implemented in software based on the description provided by their authors. The DNS-based scheme was termed DNS-RL.

The testing environment used for the evaluation process is a virtualised testbed described in [10]. The testbed contains four virtualised enterprise networks comprising a number of virtual network cells. The testbed has a scale of 1200 virtual machines, supports the use of worm daemons and has utilities for replaying network traces as background traffic.

To generate background traffic during the worm propagation experiments, the evaluation used the CDX 2014 dataset [11]. The CDX 2014 dataset meets the requirements of the evaluation because it is an attack free trace that contain payload information for the variety of protocols needed. Additionally, the traces include a wide range of collected traffic from 20 network hosts.
MUHAMMAD AMINU AHMAD

The evaluation process used two contemporary pseudo-worms that were developed based on the Microsoft RDP (CVE-20120002) vulnerability of 2012 and the Shell-Shock vulnerability of 2014 [1]. Ahmad and Woodhead [1] reported the likely susceptible population values and potential datagram sizes of the Microsoft RDP and Shell Shock vulnerabilities as circa 16.5M and 3800 bytes and 42.5k and 2000 bytes respectively.

Additionally, the bandwidth available for an infected host and the worm datagram size determine how fast a worm can send datagrams. The average Internet connection speed was estimated to be within the range 10 Mbps to 1000 Mbps [12]. Although it is impossible for a host to achieve the maximum speed of a network card, the vast majority of Internet connected hosts are capable of transmitting data at 60 Mbps to 120 Mbps [13]. Thus based on the assumption that the Internet connected hosts exhibit an average data transmission rate of 90 Mbps, the scan rate $S$, required for a single worm instance to transmit a datagram of size $M$ (in bytes), over a C megabits Internet connection per second can be determined using $S = \frac{C}{M-B}$ [1] Therefore, the likely scan rates for the RDP and Shell Shock pseudo-worms are

$$\left(\frac{90,000,000}{3800-8}\right) = 2960 \text{ and } \left(\frac{90,000,000}{2000-8}\right) = 5652$$

datagrams per second respectively. The scan rates of the pseudo-worms were scaled down by a factor of 24 and 45 for the RDP and Shell Shock pseudo-worms respectively to avoid overloading server resources. The resulting scan rates employed in the experiments are 125 “infectious” datagrams per second for RDP and Shell-Shock. Furthermore, the results of the experiments were scaled up by a factor of 24 and 45 for the RDP and Shell-Shock pseudo-worms respectively.

Ahmad and Woodhead [1] reported the number of susceptible hosts per million Internet hosts for RDP and Shell-Shock pseudoworms as 4454 and 12 respectively. Thus, due to the scale of the testbed used, which has a maximum number of 1200 hosts, four class B size ($2^{16}$) networks were used for RDP and five class A size ($2^{24}$) networks were used for Shell Shock. Thus, the resulting values were

$$\left(\frac{2^{16}}{4} + 4 \times \left(\frac{4454}{1,000,000}\right)\right) = 1168 \text{ and } \left(\frac{2^{24}}{5} + 5 \times \left(\frac{12}{1,000,000}\right)\right)$$

susceptible hosts respectively, within the relevant network address spaces.

1. **Experimentation Setup**

The evaluation experiments were conducted using the software prototypes of SAFE, DSC and DNS-RL. During the evaluation, a prototype of a detection scheme was positioned on the gateways of each network, and for SAFE, the containment system was positioned on the switches as depicted in Figure 1.
The RDP and Shell Shock worm propagation experiments were conducted using random and then hit-list scanning behaviours for each detection scheme. The random scanning technique probes IPv4 addresses within the routable address space. The hit-list scanning technique infects a list of pre-compiled vulnerable hosts and then each infected host uses random scanning. For each pseudo-worm experiment, a number of hosts (1168 for RDP and 1007 for Shell Shock) were configured with the correct daemon to make them vulnerable to worm attack datagrams while other hosts were configured to replay the CDX traces as background traffic. The worm attack and traffic replay events were executed concurrently in each experiment. The experiments were conducted without any countermeasures in place, then repeated with the countermeasures and the CDX dataset as background traffic using threshold values of 100 through 400 anomalous datagrams sent by a host in a timing window of 10 seconds. The worm infection event was initiated by sending a UDP datagram to one of the vulnerable hosts.
1.1 RDP Pseudo-worm
The RDP pseudo-worm experiment was conducted using 1160 client hosts and 8 server hosts. The pseudo-worm daemon was configured to listen on UDP port 3389 and then transmit UDP datagrams to port 3389 at a scan rate of 125 “infectious” datagrams per second, once “infected”. Five RDP pseudo-worm experiments were conducted using one initially infected host. The average result of the five experiments are presented in Figure 2. The RDP-based worm experiment was repeated with a hit-list of 10 and 20 hosts. Figures 3 and 4 show the results of RDP pseudoworm propagation using hit-lists of 10 and 20 hosts respectively.

1.2 ShellShock Pseudo-worm
The ShellShock pseudo-worm experiment was conducted using 996 client hosts and 10 server hosts. The pseudo-worm daemon was configured to listen on UDP port 8080 and then transmit UDP datagrams to port 8080 at a scan rate of 125 “infectious” datagrams per second, once “infected”. Five ShellShock pseudoworm experiments were conducted using one initially infected host. The average result of the five experiments are presented in Figure 5. As with RDP, the ShellShock worm experiment was repeated with a hit-list of 10 and 20 hosts. Figures 6 and 7 show the results of RDP pseudo-worm propagation using hit-lists of 10 and 20 hosts respectively.
6. **Discussion**

This section discusses the infection behaviours of the candidate pseudo-worms using random and hit-list scanning and the false positives observed during the experiments.

6.1 **Random Scanning Infection**

The results of random infection behaviours for the RDP and ShellShock pseudo-worms using a threshold value of 100 are presented in Figures 2 and 5. When no countermeasure solution was in place, the RDP pseudo-worm infected 95% (1110) of the hosts in eight seconds as shown in Figure 2. Additionally, the ShellShock pseudo-worm infected 95% (956) of its susceptible hosts in 145 seconds as shown in Figure 5. When the detection schemes were applied, the infections were delayed and suppressed by DSC and DNS-RL and blocked completely by SAFE. With DSC and DNS-RL, the RDP pseudo-worm infection was the infection was delayed by 12 seconds and suppressed to 44% (510) and 50% (580) respectively. The worm infections were detected by the DSC and DNS-RL schemes and the counter measure solution was applied, but the initially infected host continued sending infectious datagrams, which infected a large number of hosts. However, with SAFE, the initially infected host, for each pseudo-worm experiment, was detected and then blocked from sending out datagrams at the data-link layer, which stopped the infection completely for each of the two worm outbreak scenarios.

6.2 **Hit-list Scanning Infection**

Figures 3 and 6 show the results of the worm experiments conducted with a hit-list of 10 hosts. When no countermeasure was in place, the RDP pseudo-worm infected 95% (1110) of the hosts in 6 seconds as shown in Figure 3. The ShellShock pseudoworm attained 95% (956) infection in 55 seconds as shown in Figure 6. The RDP and ShellShock pseudo-worms attained 95% infection in 6 and 150 seconds with DSC and DNS-RL respectively. Furthermore, nine further infections were observed with SAFE...
during the RDP pseudo-worm propagation and no further infections were observed during the propagation of the ShellShock pseudo-worm. Figures 4 and 7 show the results of the worm experiments conducted with a hit-list of 20 hosts. When no countermeasure was in place, the RDP pseudo-worm infected 95% (1004) of the hosts in 5 seconds as shown in Figure 4. The ShellShock pseudo-worm attained 95% infection in 40 seconds as shown in Figure 7. Furthermore, with the DSC and DNS-RL scheme, the RDP pseudo-worm infection attained 95% in 11 and 9 seconds respectively. The ShellShock pseudo-worm attained 95% infection in 90 and 75 seconds with DSC and DNS-RL respectively. For SAFE, 56 further infections were observed during the RDP pseudo-worm propagation and no further infections were observed during the propagation of the ShellShock pseudo-worm.

6.3 Detection Performance
The results for false positive rates are presented in Figures 8 and 9 for the RDP and ShellShock pseudo-worms respectively. The schemes detected all real pseudo-worm datagrams in all the experiments conducted and therefore the true positive (TP) rates are 100%. However, the DSC and DNS-RL incurred higher rate of false positives (FP) than SAFE. SAFE has very low FP rates using 100 and 200 as thresholds and zero FP rates using 300 and 400 as thresholds. Generally, the false positive rate diminishes with rising thresholds values. Furthermore, SAFE raised few false positives with threshold values of 100 and 200, which were caused Net-BIOS service datagrams destined to UDP ports 137 and 138 by hosts in the CDX dataset.
The NetBIOS name service (UDP port 137) manages name registration and resolution for NetBIOS hosts on a local network, while the datagram services (UDP port 138) is used to send message to unique or multiple NetBIOS hosts. A NetBIOS host resolves names using a cache, a name server or an IP subnet broadcast. A NetBIOS name resolution broadcast sent by a host is received by all machines on the same network. Similarly, a NetBIOS host uses datagram services to send message to a group of hosts, which also use subnet broadcast for name resolution. These behaviours are similar to worm scanning because a host is sending datagrams to different IP addresses with the same destination ports without a prior DNS resolution. However, the rate at which the datagrams are transmitted during name resolution and datagram services are not similar to fast scanning worm. Generally, SAFE has demonstrated a better performance in terms of false positives than the DSC and DNSRL schemes across the whole experimental data set.

7. Conclusion
This paper has presented a countermeasure solution against fast network worm malware. An empirical evaluation of a software prototype of the worm countermeasure solution was used to test the scheme using a set of worm experiments. The results showed that the countermeasure solution is sensitive in detecting and isolating an identified worm infection with few or almost no false positives. The results of the comparative analysis with the two previously reported detection schemes showed that the developed countermeasure solution has a better performance.
As for future work, it is desirable to further evaluate the countermeasure mechanism using different sets of background traffic for false positives analysis. It is also desirable to further explore the ways of improving the accuracy of the detection system and speed of containment system. Furthermore, the effect of timing window size will be examined and an improved method of applying threshold values for hosts in a network will be investigated.

8. References

