Endpoint-driven Intrusion Detection and Containment of Fast Spreading Worms in Enterprise Networks

Frank Akujobi, Ioannis Lambadaris †
Department of Systems and Computer Engineering
Carleton University, Ottawa, ON, Canada
fakujobi, ioannis@sce.carleton.ca

Evangelos Kranakis †
Department of Computer Science
Carleton University, Ottawa, ON, Canada
kranakis@scs.carleton.ca

***Abstract*— Fast spreading network worms have become one of the most service-impacting threats in enterprise and ISP networks. We identify core requirements for effective detection and containment of such worms and propose a technique that uses a combination of distributed anomaly-based host intrusion detection and statistical analysis of network heuristics to detect malicious worm activity. Our proposal employs a collaborative network-centric worm containment approach. We experiment on a live test-bed with fast spreading worms and evaluate the effectiveness of our method in detecting and containing such worms. We also evaluate the system's performance when malicious worm traffic blends with benign network scanning traffic.**

Keywords – False negatives, False positives, Anomaly-based host intrusion detection, Fast spreading worms.

I. Introduction and Motivation

Techniques for detecting malicious intrusions can be divided into two broad groups: Signature-based techniques and anomaly-based techniques. *Signature-based* techniques in general require a managed database of unique malicious code signatures and are only capable of detecting malicious packets with known signatures [1]. Zero-day attacks due to malicious traffic with unknown signatures cannot be detected by such techniques. Similarly, intrusion containment systems that depend on signature-based detection schemes are largely ineffective in containing zero-day attacks. Besides, documented well-known techniques for evading signature detection exist. For instance, while some exploits incorporate special encryption and tunneling protocols to hide the signature of the code [2-4], others employ dynamic polymorphic mutations [6]. *Anomaly-based* intrusion detection systems (AIDS) can be classified as network AIDS or host AIDS depending on where the anomalous behavior is to be detected. Anomaly-based *network* intrusion detection techniques infer malicious activity in a network by detecting anomalous network traffic patterns [7-13]. While these techniques can effectively detect fast spreading worms under favorable conditions, they cannot reliably detect worm propagation that do not cause anomalous traffic patterns [6, 14]. For instance, flash worms do not exhibit anomalous number of connection failures and connection attempts since they have pre-determined lists of vulnerable targets. Such worms will therefore evade existing scanning worm detection techniques that depend solely on number of connection failures and connection attempts for detection [14]. Nonetheless, recent papers [15, 16] [2, 17] and vendor implementations [18] have recorded success in using anomaly-based *host* intrusion detection (AHID) techniques for effective detection of unauthorized intrusions. AHID are capable of leveraging large amounts of detailed context about applications and system behavior to effectively detect anomalous host behaviors [4]. Authors in [17] presented an AHIDS model that detects attacks on a host machine running Windows by checking for anomalous attempts to access the Windows registry. Further, it has been suggested that since end hosts running vulnerable software are the targets of malicious code attacks, they ought to be the point of detection [16]. The technique adopted in [16] shows that with properly instrumented detection software, host-based intrusion detection is effective and capable of eliminating false positives. Their containment approach however relies on broadcast notification messages from infected hosts to other hosts in the network while host-based filters are used to block the intrusion. It is our opinion that while host-based detection holds lots of promise, host-based filtering is not scalable and cannot address worm containment in large enterprise networks. For instance, in ISP networks, network administrators do not have any direct control over individual client machine configuration and therefore cannot depend on them for worm containment. Another problem with host-based containment techniques is that a successful compromise of a participating host could subvert the containment technique [20, 21].

We point out that there are still gaps between existing worm detection and containment techniques and a feasible real world approach to effectively combat fast spreading zero-day enterprise worms. In this paper, we propose a technique that uses *host-based* intrusion detection to

†This research was partially funded by grants from Natural Sciences and Engineering Research Council (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS).

achieve intrusion containment in the *network*. As explained above, there is ample evidence of the effectiveness of host-based anomaly detection techniques in detecting host intrusions and anomalous behaviour with minimal false alarms [16][17]. Our technique is unique because it leverages the benefits of host-based detection to achieve distributed collaborative network intrusion containment of fast spreading worms and malicious code with zero false positives and false negatives.

Contributions of this work are:

- We propose requirements for effective intrusion detection and collaborative containment.
- We present our proposed novel technique and algorithm for detection and containment of fast spreading worms.
- We demonstrate the effectiveness of our technique on a live testbed and analyse its performance.

In the next section we identify core requirements that an effective detection/containment technique should meet.

A. Requirements for effective intrusion detection and containment

1) Verifiable detection of malicious intrusions: An effective detection technique should include detection of actual malicious intrusion attempts or observation of verifiable intrusion as a basis for confirming the existence of malicious activity. Dependence on network traffic heuristics alone for detection either on a host network interface or on network routers does not offer guarantees on the rate of false positives and false negatives [16][29]. The technique proposed in this paper uses an anomaly-based host intrusion detection mechanism that alerts only when an unauthorized intrusion occurs. Such instrumented endpoint intrusion detection software are known to minimize false alarms [16].

2) Collaborative network-based containment: Containment can be executed either at the host level or on the network. A containment technique that ensures fast propagation of the containment action is required. Ideally, such a technique should propagate the containment action at a rate comparable or superior to the rate of propagation of fast spreading worms. While network-based techniques protect a fraction of the target population when executed, host-based techniques protect only a single host at a time. Clearly, for fast spreading containment, network-based techniques should be deployed. Further, a collaborative containment architecture should be used to suppress large scale worm attacks [20-22]. Authors in those papers experimented with the concept of collaborative cellular containment where the network is broken into logical cells, each with in-built containment capabilities. Their results show that such a distributed and cooperative approach can greatly improve containment systems. Similarly, the *pushback* technique [28] employs a collaborative packet dropping technique to suppress distributed denial of service (DDoS) attacks. Our technique executes network-based containment at the first upstream router (i.e the gateway router) of the subnet or cell or VLAN under attack. The gateway router of the cell then sends signals to peer routers to execute similar containment procedures thus achieving enterprise-wide collaborative containment.

3) Automated rapid response: Response to worm invasion must be automated and fast to be effective. An effective response should involve minimal overhead and processing time between intrusion detection and containment execution. Our technique uses a host-based detection technique and an optimized correlation algorithm to automatically and rapidly implement a network filter against intrusion traffic.

B. Merits of our approach

The technique proposed in this paper combines the significant benefits of host-based anomaly detection with statistical correlation of network heuristics to determine an intrusion traffic flow. It employs a distributed network-centric containment approach to rapidly contain the intrusion traffic. The first merit of our technique stems from its ability to accurately detect unauthorized intrusions due to both known and unknown malicious code which tamper with vulnerable endpoints. The technique is non-responsive to benign traffic hits such as reconnaissance network scans irrespective of the traffic rate of the scans. Intrusion detection alerts are generated only when verifiable malicious intrusion traffic is detected. Occurrence of false alarms is therefore eliminated since the technique responds only to verifiable intrusions and not to benign intrusion attempts which occur quite frequently on enterprise networks and on the Internet. The second merit is that the containment action is network-centric. This makes the technique feasible in real world scenarios where containment actions are typically carried out at layer-3 boundaries thus protecting a fraction of the vulnerable population with a single containment action. Also, our reactive containment technique generates a router filter against ingress malicious intrusion traffic that match the detected attacker's profile. The profile is a 3-tuple consisting of the attacker's source IP address, the target port, and the transport layer protocol used. This reactive blocking ensures that access to the vulnerable service by only the attacking IP address is blocked. Some recent worms [33,34] exploit legitimate ports such as port 80, 135 that are required for enterprise services. Our reactive blocking ensures that only the detected attacker's IP address is filtered from accessing such services. Other legitimate users of the services continue to have access to the services. Third, the proposed detection, correlation and containment protocols run continuously even after a router filter has been generated against an attacker.

Multiple attacks from several attackers can therefore be detected and contained automatically. Such attack models are typical with botnets [35]. Recent work in [36] suggests that a similar dynamic quarantine defense technique is capable of slowing down propagation of network worms. After a successful automated defense against multiple attackers, IT security personnel can review the established router filters and embark on a remediation procedure based on pre-defined enterprise IT security policies. Our proposed system can also be configured to automatically remove the router filters after a defined period following the worm defense. The fourth merit is that the technique lends itself to distributed and collaborative containment. This merit is also enjoyed by the *pushback* technique [28,38] where containment action is initiated from within a logical zone in a network and spreads out into the larger enterprise network. The pushback mechanism detects a DoS attack by monitoring packet drops on router links and determining the aggregate [1] that occurs most frequently amongst the dropped packet. It assumes that such an aggregate is responsible for the high congestion. However, a practical drawback of the pushback detection mechanism is that most modern routers and switches have capabilitites for multi-gigabit links in core and distribution networks and gigabit links to servers and desktops. For such high capacity links, significant packet drops will be observed only when catastrophic DoS attacks occur and delay-sensitive applications would suffer greatly long before the pushback mechanism detects packet drops. Also, there is a strong chance that a huge number of packets not belonging to a DoS attacking traffic can be dropped in a congestion situation. This is especially true in QoS-enabled networks where packets are preferentially dropped to offer better QoS to other delay sensitive applications. The pushback mechanism will result in severe collateral damage in such circumstances. There is also the possibility that some fast spreading worms may not cause significant packet drops on routers and would therefore evade the pushback mechanism. Our proposed technique does not suffer from these drawbacks since it detects malicious activity by detecting verifiable malicious intrusions. Finally, due to the responsiveness of our technique which ensures rapid containment of detected intrusion traffic, it is capable of significantly reducing the spread of fast propagating worms before core and distribution gigabit links in enterprise and ISP networks become congested. Our technique can therefore be effective in combatting DoS attacks due to malicious fast spreading worms.

[1] defined as a subset of the traffic with an identifiable property e.g. same destination IP address.

C. Outline of the paper

Section II of this paper describes details of the proposed detection technique and algorithms used for iterative analysis of network heuristics and intelligent containment of intrusion traffic. Section III contains experimental analysis of the technique while section IV presents our conclusion and highlights ideas and possible extensions for future work.

II. Automated intrusion detection and collaborative containment

The proposed technique is divided into three sub-tasks: i) Detection of intrusion ii) statistical determination of intrusion traffic and iii) containment of intrusion and its propagation.

A. Detection of Intrusion

We propose a distributed intrusion detection strategy which uses our instrumented anomaly host-based *detector agent (DA)* software running on designated strategically located *detector endpoints (DEs)* within an enterprise network for host-based detection. In large enterprise networks, a number of independently functional DEs can be located within distributed subnets or cells to detect and respond to intrusion attempts on the subnet. This distributed detection enjoys the benefit of operating close to the point of infection or intrusion. It is our assumption that such an approach would be more responsive to a worm invasion compared to other approaches that attempt to detect intrusions at a single ingress point on the network. The DA running on a DE performs continuous real-time recording of the *profile* of all network traffic destined to the DE and originated from outside its subnet. We define a *profile* as a 3-tuple consisting of *srcIP*, *dstport*, *proto*. *srcIP* is the source IP address in the IP header of packets captured by the DE, *dstport* is the target port and *proto* is the transport layer protocol used. This *profile* format was chosen because it contains sufficient information to implement a traffic flow-specific block on most real world enterprise edge networking devices. The DE records are reinitialized regularly so that only recent intrusion activities are maintained in the records while ensuring that the size of the record remains under 5KB. When a verifiable unauthorized intrusion is detected by a DA it responds by notifying the upstream gateway router (GR) of the intrusion and triggering a transfer of its record to the router over an encrypted channel. The size of the DE record has a proportional relationship with the DA-to-GR record transfer time which directly impacts the containment interval of our technique. It is therefore important to maintain the size of the DE record under 5KB by regularly initializing the record. With a DE record size of under 5KB, encouraging experimental results were

obtained. Their dedicated functions in a subnet or cell are to monitor, record, detect and respond to verifiable intrusion attempts. They do not respond to benign traffic hits such as network scans. In our experiments, we used port 9090 UDP to emulate a vulnerable service running on the DEs and malicious intrusion detection was verified only when an attacker establishes a connection to the vulnerable service. DEs do not initiate communication with any host outside their cell. Traffic hits on the DEs from outside their cell must therefore be as a result of benign network scans or verifiable malicious intrusions. A minimum of two DEs are required in each logical cell to facilitate our correlation algorithm on the GR. The DEs within a cell also maintain state with each other to ensure their GR does not receive multiple intrusion notifications for a single attack instance.

B. Iterative statistical determination of intrusion traffic

A process running on the upstream gateway router (GR) monitors the transfer of records from DEs within a target subnet. When the GR receives the first record from a DE in the subnet it immediately establishes other secure sessions with the remaining DE(s) in that subnet and retrieves their records. A *correlation* algorithm triggered on the gateway router performs an iterative statistical analysis on all the records to determine the *profile* of the most likely intrusion traffic. We use a number of states to describe the correlation algorithm.

1) Modeling: The algorithm starts by modeling each DE record as an event Y_i while the profiles in each record are modeled as the outcomes of the event. The outcomes are modeled as a discrete random variable X with a set of probabilities $P_X(x_i) = P[X = x_i]$. Using this model, $P_X(x_1)$ represents the probability that a particular profile x_1 occurs in the record. The event Y for a particular record can be represented as:

$$Y = \{x_i : x_i \text{ is a } \textit{profile} \text{ in the record}\} \tag{1}$$

2) Normalization: In this state the algorithm uses Z-score normalization to standardize the set of probabilities on a common scale. Z-score for $p_X(x_i)$ is defined as:

$$Z(x_i) = \frac{p_X(x_i) - \overline{p_X(x_i)}}{\text{stdev}(p_X(x_i))} \tag{2}$$

where $p_X(x_i)$ is the input variable, and the normalized scores have a mean of zero and a standard deviation of one.

3) Comparison: In this state the algorithm first compares the outcomes in Y_i and checks for *common outcomes* across all Y_i events that occur in i DEs. A common outcome is an outcome that exists on all Y_i events. It generates event C as a subset of Y such that the outcomes of event C consist of the common outcomes across all Y_i events. The event C can be represented as:

$$C = \{x_i : x_i \text{ is a common outcome}\} \tag{3}$$

$$= Y_1 \cap Y_2 \cap Y_3 .... \cap Y_i \tag{4}$$

If (*number of outcomes of event* $C = 1$)
{
then outcome is associated with intrusion traffic profile; transition to the *Trigger* state;
}
If (*number of outcomes of event* $C = 0$)
{
then worm activity does not exist; transition to the *Abort* state;
}
If (*number of outcomes of event* $C > 1$)
{
then transition to the *Selection and Iteration* state;
}

4) Selection and Iteration: In this state the algorithm generates event H such that the outcomes in H are a selection of the outcomes in C with probabilities that have a Z-score greater than a chosen threshold Φ where $0 \leq \Phi \leq \infty$. The outcomes of event H are considered suspicious due to the degree of statistical deviation of the frequency of their occurrences from the mean and are therefore selected for further analysis. Φ is a configurable parameter on the GR. The event H can be represented as:

$$H = \{x : \text{Z-score of } p_X(x) > \Phi\} \tag{5}$$

The algorithm iteratively determines the intrusion traffic using the following *if loop*:
If (*number of elements in* $H = 0$ *and* $\Phi \neq 0$)
{
repeat *Selection and Iteration* state with lower Φ value ($\Phi_{n+1} = \Phi_n - \delta$[2]);
} else if (*number of elements in* $H > 1$)
{
repeat *Selection and Iteration* state with greater Φ value ($\Phi_{n+1} = \Phi_n + \delta$)
} else if (*number of elements in* $H = 0$ *and* $\Phi = 0$)
{
worm activity does not exist; transition to the *Abort* state
} else if (*number of elements in* $H = 1$)
{
element is associated with intrusion traffic profile;

[2] δ is the iteration interval and is an input parameter of the correlation algorithm

transition to *Trigger* state.
}

5) Abort: In the *Abort* state the algorithm aborts and waits to be triggered again.

6) Trigger: In the *Trigger* state the algorithm collects the intrusion traffic profile associated with the detected common outcome and triggers automatic containment of the intrusion traffic (see section C).

It is our assumption in the *Comparison* state that during a worm invasion there is a high probability that DEs located in a cell under attack will experience early intrusion attempts and therefore will record *profiles* of the intrusion traffic. Previous work reveal that several recent scanning worms such as Code RED II [23-25] and Nimda [26, 27] preferentially target other hosts from IP address ranges closer to the vulnerable target host (i.e. in the same /24 or /16 network). Intrusion attempts as a result of flash worm activity will also be captured by the DEs if the DEs run the same vulnerable software as hosts within the subnet they are located and therefore cannot be differentiated from the vulnerable hosts. Our assumption will fail only if the attacker has prior knowledge of the DEs and instruments a worm that selectively avoids malicious intrusion attempts on the DEs. We also assume that while non-existence of common outcomes across the Y_i events suggests non-occurrence of worm activity, the concurrent occurrence of a common outcome and a detected verifable malicious intrusion attempt is a strong indication of worm activity. The selection and iteration process is used to progressively and statistically analyze the records from the DEs until the profile that corresponds to the common outcome with the greatest relative frequency of occurrences is determined if one exists. The correlation algorithm assumes that such a profile must be associated with the malicious intrusion traffic and triggers a containment action.

The detection and correlation algorithms are capable of deciphering fast spreading malicious worm traffic from benign peer-to-peer traffic and other forms of legal traffic. The correlation algorithm iteratively determines the traffic profile which registers the greatest numbers of simultaneous hits on all DEs in a target cell. First, such a traffic profile must be captured on all DEs (i.e a common outcome). Among several common outcomes, the traffic profile corresponding to the common outcome with the greatest relative frequency of occurrence is then determined during the Selection and Iteration phase of the correlation algorithm. Legal peer-to-peer traffic and regular web traffic would not register simultaneous hits on pre-defined open vulnerable ports running on all the DEs let alone register a relatively large number of simultaneous hits on all the DEs. Only a fast spreading illegal traffic or worm would be capable of registering such huge number of simultaneous hits on all the DEs. Recall that DEs do not initiate communication with any host outside their cell. Traffic hits on the DEs from outside their cell must therefore be as a result of benign network scans or verifiable malicious intrusions.

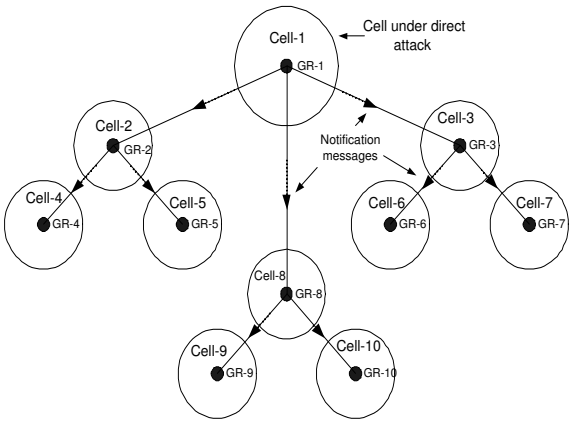

Fig. 1. Distributed collaborative containment strategy

C. Containment of intrusion

Our technique executes containment of malicious intrusion traffic by automatically applying router filters on the GR to block the profile associated with the inbound malicious intrusion traffic as determined by our correlation algorithm. The filter is applied against the 3-tuple consisting of the IP address of the attacker, the target vulnerable port and the transport protocol used. This filter ensures that inbound access to the vulnerable target service by the attacking IP address is blocked. If the target port is required for legitimate services e.g port 443, 8080, other legitimate users of the service can continue to enjoy the service. While authors in [19] affirm that content filtering containment techniques are generally more effective than address blocking techniques, simulation results in [19] show that if certain conditions are met such as if reaction time can be minimized, address blocking techniques can be more effective. Besides, content filtering techniques also suffer from all the previously discussed inherent drawbacks of signature-based worm detection strategies. Our approach uses a rapid reactive blocking technique that meets the timing conditions for effective worm containment [19]. We propose a distributed collaborative containment strategy in which the GR communicates its recent worm containment action along with the 3-tuple profile of the blocked malicious intrusion traffic to peer gateway routers (see Fig. 1). In Fig. 1, cell-1 represents the subnet under direct attack and GR-1 is the gateway router for cell-1. After executing a reactive block against the attacker's profile, GR-1 notifies its peer GRs (GR-2, GR-3 and GR-8). A *reactive blocking* protocol running on the peer gateway routers

determines how the routers handle the notification. The *reactive blocking* protocol was designed to immediately block the malicious intrusion traffic after receiving the notification and then monitor the suspected vulnerable target ports for existence of worm activity. If the protocol determines that worm activity is not prevalent it disables the block thereby preventing a potential denial of service. We use a number of states to describe the *reactive blocking* protocol.

1) Reactive blocking: The protocol starts by creating a router filter on a peer GR that blocks the profile associated with the malicious intrusion traffic. The filter blocks ingress traffic that match the profile from entering all cells or subnets existing on the peer GR.

2) Monitoring: In this state, with the filter still applied, the peer GR carries out real-time monitoring and recording of the number of hits on the filter for a y-second interval to confirm actual existence of worm activity. After y seconds the algorithm computes a *probing rate* where:

$$\textit{Probing rate} = \frac{\textit{number of hits on filter}}{y} \tag{6}$$

It then carries out the following *if loop* with chosen parameter w.

If (*probing rate* $> w$ *probes/sec.*)
{
then worm attack is prevalent; transition to the *Notification* state;
}
If (*probing rate* $\leq w$ *probes/sec.*)
{
then worm attack is not prevalent; transition to *Unblocking* state.
}

3) Notification: In this state the peer GR notifies other peer GRs of the suspected attacker's profile. This also triggers the reactive blocking protocol on the peer GRs (see Fig. 1).

4) Unblocking: In this state the filter is removed to prevent a denial of service on legitimate traffic.

Both y and w are configurable parameters of the protocol. Fast spreading worms are known to exhibit probing rates in the order of tens of thousands probes/second. It is our assumption that in an event of a fast spreading worm invasion the reactive blocking protocol on peer GRs will reach *Notification* state if y and w parameters are properly chosen. In that scenario, all enterprise gateway routers eventually get to be aware of the attacker's profile and establish filters against ingress traffic that match the attacker's profile thus achieving enterprise-wide fast and automated containment.

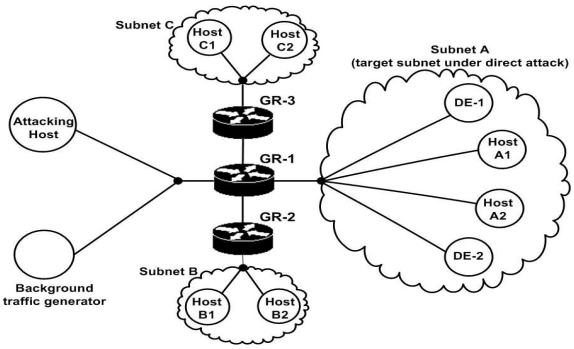

Fig. 2. Network layout of experimental testbed

III. Experimental design and analysis

To evaluate the functionality and performance of our proposed technique, we emulated self propagating worm invasion attacks using a modified blaster worm code which attempts to randomly establish socket connections on UDP port 9090 with a pre-generated list of vulnerable hosts (Host-A1, Host-A2, DE-1, DE-2) in the target subnet A (see Fig. 2). Vulnerable hosts were emulated using endpoints running a hypothetically vulnerable UDP service on port 9090. They are considered infected when they accept connections from our instrumented worm code and a file named *intrusion* is copied into their /root/ directory. UDP was used as the worm transport layer protocol to avoid TIME-WAIT problems associated with overloading servers with TCP connection as well as to emulate *TCP-unfriendly* worm traffic. Port 9090 was chosen arbitrarily. We measured average infection rates of upto 70 hosts per second (h/s) using a single instance of our instrumented worm. In comparison, the code red worm [25] infected 359000 hosts in less than 14 hours, equivalent to an average infection rate in the order of 7.1h/s. The Witty worm [30] infected 110 hosts in the first 10 seconds, equivalent to an average infection rate of about 11h/s while the Slammer worm [31] infected more than 75,000 hosts within 10 minutes, equivalent to an average infection rate of over 125h/s. When hosts within the target subnet are infected by the attacking host, they in turn propagate the worm by also attempting to establish connections with a pre-generated list of hosts outside the target subnet. In our experiment we chose to limit worm propagation from infected hosts in subnet A to destinations outside of subnet A in order to eliminate the possibility of having the DEs erroneously respond to attacks originating from within their cell.

In the experiments we varied the infection rate of our emulated worm between 1h/s and 35h/s and observed the performance of our proposed detection and containment technique. Fig. 2 shows the layout of our live experimental test-bed. We used only two detector endpoints (DE-1 and DE-2) and two vulnerable hosts (Host-A1 and Host-A2) in the target net-

work (subnet A) to simplify demonstration of the technique and result analysis. GR-1 is the gateway router for subnet A while GR-2 and GR-3 are the gateway routers for subnet B and subnet C respectively. All gateway routers ran our correlation algorithm and reactive blocking protocol. After GR-1 executes a containment block, it notifies GR-2 and GR-3 and they too execute similar blocks following the reactive blocking protocol. The gateway routers, detector endpoints and vulnerable hosts were implemented on Linux systems running kernel 2.4 with 733Mhz Intel processors and 256MB RAM. The GR-1 provided network time protocol (NTP) services for time synchronization and all link capacities were set to 100Mbps full duplex.

A. Experiment 1: Worm attack without background traffic

In this experiment the attacking host was used to launch direct attacks on the four hosts in subnet A (Host-A1, Host-A2, DE-1, DE-2) selecting each one randomly. Our *detector agents (DAs)* ran on DE-1 and DE-2 and re-initialized DE records in 20-second intervals. After infection, Host-A1 and Host-A2 attempt to propagate the worm code by continuously attempting socket connections on UDP port 9090 with Host-B1 and Host-B2 in subnet B and Host-C1 and Host-C2 in subnet C. This emulates worm propagation. The objectives in this experiment were to: i) Evaluate the responsiveness of our proposed technique in protecting a target network from unauthorized worm intrusions. ii) Evaluate effectiveness of the technique in containing spread of network worms, hence protecting other vulnerable hosts and services that reside outside the initial target network. The following system performance metrics were measured:

1) Containment interval T: computed as the time interval between detecting the worm intrusion traffic on a DE and the establishment of a containment block against the attacking host on the gateway router. This interval comprises the time necessary for propagating the intrusion information to the upstream GR, the time required for the GR to retrieve all the DE records from all the DEs in the target cell, and the time required to run our correlation algorithm and reactive blocking protocol on the GR.

2) Rate of false negatives: computed as the fraction of total number of experimental runs performed that did not result in a containment action even in the presence of malicious worm activity.

To improve statistical significance of our results, 100 attack instances were carried out for each infection rate level used in the experiment. The average containment interval and rate of false negatives were recorded. Parameters used by the correlation algorithm were: $\Phi = 0.05$, $\delta = 0.05$. These values determine the number of iterations performed

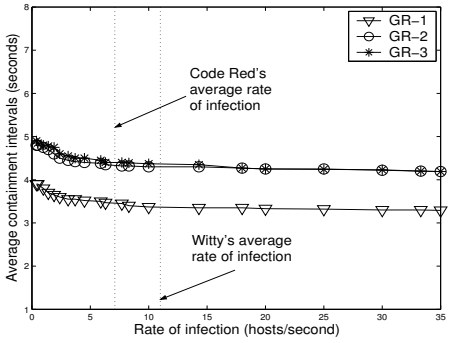

Fig. 3. Average containment intervals

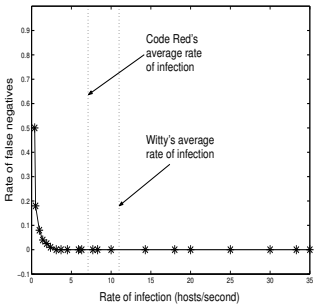

Fig. 4. Rate of false negatives without background scanning traffic

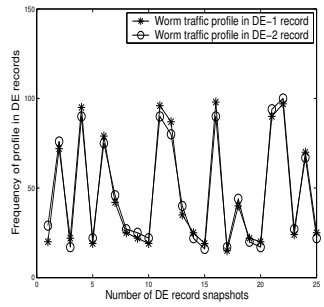

Fig. 5. Recorded *profiles* at 25h/s infection rate

by the correlation algorithm before the intrusion traffic profile is determined. Parameters used for the *reactive blocking protocol* were: $y = 20$seconds, $w = 10$probes/second. Optimizing the choice of parameters is reserved for future work. Fig. 3 to Fig. 7 show the results obtained. Our results in Fig. 3 show that the measured average containment interval on GR-1 remained nearly steady at a mean value of 3.4s with a variance of $0.03s^2$ as we varied the infection rate of our instrumented worm. Similarly GR-2 and GR-3 recorded mean containment interval values of 4.33s and 4.39s respectively with variances of $0.042s^2$ and $0.045s^2$ respectively. The results suggest that containment interval on the gateway routers is largely independent of worm propagation speed or infection rate but is primarily determined by the operations of the detection algorithm and containment technique. The results also suggest that our approach to collaborative containment is capable of containing fast spreading worms within *five seconds* of detection. In comparison, recent simulations by Moore et al suggest that an effective worm containment should require a reaction time of well under sixty seconds [19]. While worm infections in an enterprise network will not be completely prevented by our technique during a worm outbreak, the results suggest our technique is capable of protecting a proportion of vulnerable hosts in any enterprise or global network during a zero-day fast spreading worm invasion. Factors that determine the size of the protected portion of enterprise hosts include worm propagation speed and de-

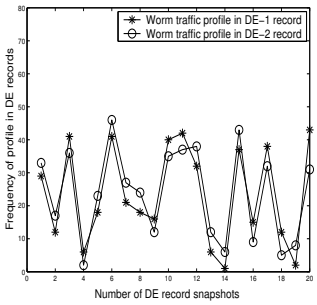

Fig. 6. Recorded *profiles* at 10h/s infection rate

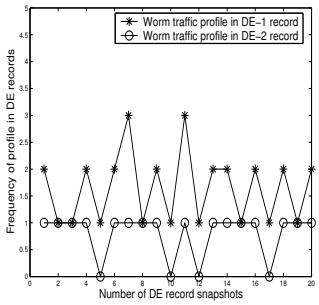

Fig. 7. Recorded *profiles* at 0.5h/s infection rate

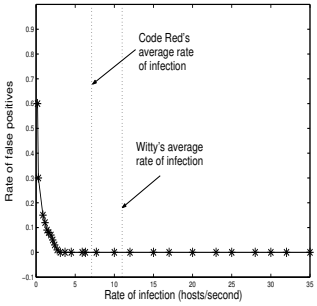

Fig. 8. Rate of false positives with scanning traffic

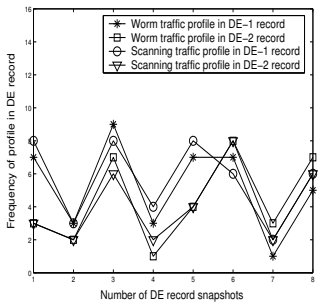

Fig. 9. Recorded *profiles* at 2h/s infection rate with scanning traffic

ployment strategy of the technique. Network placement of GRs and choice of configuration parameters for the both DEs and GRs are crucial to rapid collaborative containment in an enterprise network. Fig. 4 shows that for infection rates over 3h/s our technique exhibited zero false negatives.

However, for infection rates less than 1h/s we observed non-zero rates of false negatives. For our correlation algorithm to determine the offending traffic profile during worm activity, the active DEs in the target subnet must *simultaneously* have record of the worm's traffic profile. Fig. 5, Fig. 6 and Fig. 7 show snapshots of DE records for experiments with infection rates of 25h/s, 10h/s and 0.5h/s respectively. It can be observed in Fig. 5 and Fig. 6 that for 25h/s and 10h/s infection rates both DE-1 and DE-2 simultaneously have entries of the worm traffic profile for all captured snapshots. This results in zero false negatives (see Fig. 4). For 0.5h/s infection rate (Fig. 7), we observe that worm attacks do not spread fast enough to register simultaneous records on both DE-1 and DE-2 in a number of instances. This results in higher rates of false negatives. *The detection and correlation algorithm proposed in this paper is however optimized for detection of fast spreading worms*. We leave detection of stealthy worms for future work.

B. Experiment 2: Worm attack with background network scanning traffic

In this experiment, the same experimental conditions used in experiment 1 were maintained, with the addition of a continuous background scanning traffic. The network scanning traffic stream was generated using a UDP socket application attempting connections to port 1234 on the four hosts in subnet A (Host-A1, Host-A2, DE-1, DE-2). See Fig. 2. The traffic is benign since it does not attack the vulnerable service on port 9090 UDP. Typically, such network scans are stealthy and are used by attackers for reconnaissance to enable them locate potential targets before launching an actual attack [37]. The packet transmission rate of the scans was therefore kept constant at *two packets per second*. The objective of the experiment was to assess the impact of such network scans on the accuracy of our detection and correlation techniques. We measure the *rate of false positives* exhibited by our technique when worm traffic blends with the network scans. Rate of false positives was computed as the fraction of total number of experimental runs that triggered a containment action against a traffic profile that did not belong to the attacking worm.

Results in Fig. 8 show that for infection rates greater than 3h/s, our algorithms were accurate in determining the attacker's profile exhibiting 0% rate of false positives. Using detection of verifiable malicious intrusions as trigger for our algorithm plays a vital role in ensuring zero false positives alerts for fast spreading worms. For stealthy worms with infection rates less than 2h/s, the results show non-zero rates of false positives. In this region, worm traffic is indistinguishable from the scanning traffic, and exhibits in a number of instances, lower frequency of occurrence in DE records compared to the scanning traffic (Fig. 9). It therefore evades our correlation algorithm even though verifiable intrusions due to the worm traffic were detected by the DEs. Essentially, when worm traffic blends with network scans, our technique will record non-zero false positives *only* if probing rate of network scans constantly exceeds the probing rate of fast spreading worm traffic on the target cell. Since this is rarely the case for real-world fast spreading worms our proposed technique can be considered accurate in detecting fast spreading worms. However, *in the absence of worm traffic, benign network scans would be ignored by the DEs since the scans do not attempt to exploit the vulnerable service running on the DEs and therefore cannot be verified.*

IV. Relevant Discussion

Concerns can be raised about the performance of our technique during DoS attacks and whether or not sending intrusion notification messages across the network would exacerbate already congestion links in such a circumstance. Previous work by Moore et al [19] suggests that a minimum reaction time threshold of 60 seconds is required for any worm detection and reaction mechanism to be effective. In [19], reaction time of a containment system was defined to include

the time necessary for detection of malicious activity, propagation of the information to all hosts participating in the system, and the time required to activate any containment strategy once this information has been received. Other related work show that it took several hours for Code Red I v2 to spread among over 350,000 Internet hosts [23]. Slammer worm [31] infected more than 90 percent of vulnerable Internet hosts, equivalent to more than 75,000 hosts within 10 minutes while the Witty worm [30] infected almost all of its 12,000 victims in 45 minutes. The Witty worm infected 110 hosts in the first 10 seconds. However, our experiments show that our proposed technique is capable of automatically containing fast spreading worms within 5 seconds of detection on any of the DEs in a target subnet. The interval between release of the worm in the wild to detection of the worm on a DE running our algorithm depends on the propagation speed of the worm. This interval will be very short, in the order of few seconds for fast spreading worms. Our technique, if deployed on all edge networks on the Internet is therefore capable of significantly reducing the spread of these fast propagating worms before core and distribution gigabit links in enterprise or ISP networks become congested. Besides, since only small-sized notification packets (under 100 bytes) are transmitted between peer GRs running our reactive blocking protocol, GR to GR traffic is unlikely to congest multiple gigabit links typically used for core and distribution connections. In switched edge networks, the DEs as well as other hosts within a target network would be typically connected through switch ports which constitute seperate individual collision domains. This also minimises the possibility of having congested links between the DEs and the gateway router upon detection of an attack. In addition, the observed size of the DE records transfered between each DE and a GR is relatively small, in the order of 5KB and is therefore unlikely to congest 100MB links typically used to connect hosts to switched networks. On the contrary, our proposed technique can be effective in combatting DoS attacks due to malicious fast spreading worms. Using our technique, such DoS attacks can be contained within few seconds of detection.

V. Conclusion and future work

In this paper we outlined crucial requirements for effective detection and containment of fast spreading malicious worms and proposed a distributed collaborative containment technique which relies on distributed host-based anomaly detector endpoints for notification. We identified gaps between existing proposals for detection and containment of such worms and our proposed solution. Experimenting on a live test-bed, our results revealed that the technique is capable of containing fast spreading zero-day worms within *five seconds* of detecting the attack with zero false positives and false negatives. The results also show that in the presence of benign network scans, fast spreading malicious worms can be accurately detected.

For future work, there are a number of possible directions. First, we intend to carry out further analysis of our distributed host-based intrusion detection and network centric containment technique for combatting fast spreading worms. This involves development of analytical models to quantitatively evaluate the effectiveness of the technique in slowing down worm propagation as well as protecting a vulnerable population during a worm invasion. Second, we would like to extend our host-based intrusion detection and network-centric collaborative containment technique to DDoS attacks that may not necessarily be caused by infectious worms. DDoS attacks are of various kinds and it is unlikely that one defense mechanism can defend against all DDoS attack types. We intend to identify DDoS threat models that our technique protects against and evaluate its performance under such attack scenarios. Third, cells within large networks vary in size and while our algorithm requires a minimum of two detector endpoints in a cell to trigger our correlation algorithm we have not provided guidelines for choosing an optimum number of detector endpoints for a given cell size if one exists. This also requires some investigation.

References

[1] K. Wang and S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In Proceedings of the 7th International Symposium on Recent Advanced in Intrusion Detection (RAID), pages 201-222, September 2004.

[2] M. Locasto, K. Wang, A. Keromytis, and S. Stolfo. Flips: Hybrid adaptive intrusion prevention. In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID), September 2005.

[3] M. Handley, C. Kreibich and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", Proc. USENIX Security Symposium, 2001.

[4] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated Worm Fingerprinting. In Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), December 2004.

[5] S. Rubin, S. Jha, and B. Miller. Automatic generation and analysis of NIDs attacks. In Proceedings of 20th Annual Computer Security Applications Conference, Tucson, AZ, USA, Dec. 2004. IEEE Computer Society.

[6] O. Kolesnikov and W. Lee. Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic. Technical report, Georgia Tech, 2004.

[7] D. Whyte, P.C. Van Oorschot, E. Kranakis. Detecting Intra-Enterprise Scanning Worms Based on Address Resolution. Annual Computer Security Applications Conference (ACSAC'05). Dec. 2005

[8] D. Whyte, E. Kranakis, and P. Van Oorschot. DNS-based Detection of Scanning Worms in an Enterprise Network. In Network and Distributed Systems Symposium (NDSS), 2005.

[9] F. Guo, “Traffic Analysis: from Stateful Firewall to Network Intrusion Detection System, ” RPE Report, January 2004.

[10] P. Barford, J. Kline, D. Plonka, and R. Amos. A signal analysis of network traffic anomalies. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop, Marseilles, France, November 2002.

[11] S. Singh, C. Estan, G. Varghese, and S. Savage. The EarlyBird system for real-time detection of unknown worms. Technical Report CS2003-0761, University of California, San Diego, August 2003.

[12] A. Habib, M. Hefeeda, and B. Bhargava. Detecting service violations and DOS attacks. In Proceedings of Network and Distributed System Security Symposium, San Diego, pages 177 189, February 2003.

[13] M. Mahoney, P. Chan: Detecting novel attacks by identifying anomalous network packet headers. Technical Report CS-2001-2, Florida Institute of Technology (2001).

[14] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The Top Speed of FlashWorms. In 2nd ACM Workshop on Rapid Malcode (WORM), 2004.

[15] H. J. Wang, C. Guo, D. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In ACM SIGCOMM, August 2004.

[16] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham. Vigilante: 12 EuroSys 2006 End-to-end containment of Internet worms. In Proc. of the 20th ACM Symp. on Operating Systems Principles, Brighton, UK, October 2005.

[17] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. J. Stolfo. Detecting malicious software by monitoring anomalous windows registry accesses. In Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID-2002), Zurich, Switzerland, October 2002.

[18] Cisco Systems Inc. Cisco Security Agent ROI: Deploying Intrusion Protection Agents on the Endpoints. 2003

[19] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In Proceedings of IEEE INFOCOM, April 2003.

[20] N. Weaver, S. Staniford, and V. Paxson. Very Fast Containment of Scanning Worms. In 13th Usenix Security Symposium, 2004.

[21] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In Proceedings of the 3rd DARPA Information Survivability Conference and Exposition, April 2003.

[22] M. Cai, K. Hwang, Y. Kwok, S. Song, Y. Chen. Collaborative Internet Worm Containment. IEEE Security and Privacy Magazine, May/June 2005.

[23] D. Moore, C. Shannon, and K. Claffy. Code red: A case study on the spread and victims of an internet worm. In Proceedings of ACM SIGCOMM Internet Measurement Workshop, November 2002.

[24] eEye Digital Security. CodeRedII Worm Analysis. http://www.eeye.com/html/Research/Advisories/AL20010804.html

[25] SecurityFocus. SecurityFocus Code Red II Information Headquarters. http://aris.securityfocus.com/alerts/codered2/.

[26] CERT. CERT Advisory CA-2001-26 Nimda Worm. http://www.cert.org/advisories/CA-2001-26.html

[27] Symantec. W32.nimda.a@mm. http://www.symantec.com/avcenter/

[28] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In Proceedings of Network and Distributed System Security Symposium, San Diego. The Internet Society, February 2002.

[29] S. G. Cheetancheri, J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, E. M. Schooler. A Distributed Host-based Worm Detection System. In Proceedings of ACM SIGCOMM Workshops on Large scale attack defences, September 2006.

[30] C. Shannon, D. Moore. The Spread of the Witty Worm. IEEE Security Privacy, vol. 2, no. 4., Jul./Aug. 2004.

[31] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. IEEE Security Privacy, vol. 1, no. 4, Jul./Aug. 2003.

[32] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The Spread of the Sapphire/Slammer Worm. Tech. rep., CAIDA, Jan 2003.

[33] Microsoft Inc. Virus alert about the Nachi worm. Article ID 826234, Revision 5.6. http://support.microsoft.com/kb/826234, February 5, 2007.

[34] CERT Advisory CA-2001-19. ”Code Red” Worm Exploiting Buffer Overflow In IIS Indexing Service DLL. http://www.cert.org/advisories/CA-2001-19.html, January 17, 2002.

[35] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, R. Lipton. A Taxonomy of Botnets. Cyber Trust 2005 posters. September 2005.

[36] C. C. Zou, W. Gong, and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In Proceedings of ACM CCS Workshop on Rapid Malcode (WORM03) (October 2003).

[37] V. T. Lam, S. Antonatos, P. Akritidis, and K. G. Anagnostakis. Puppetnets: Misusing web browsers as a distributed attack infrastructure. In 13th ACM Conference on Computer and Communications Security. ACM, 2006.

[38] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. ACM Computer Communications Review, Vol. 32, No. 3, July 2002.