RC24121 (W0611-182) November 28, 2006
Computer Science

IBM Research Report

Proactive Security for Mobile Messaging Networks

Abhijit Bose
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Kang G. Shin
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Proactive Security For Mobile Messaging Networks

Abhijit Bose*
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
bosea@us.ibm.com

Kang G. Shin
Department of Electrical Engineering and
Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
kgshin@eecs.umich.edu

ABSTRACT

The interoperability of IM (Instant Messaging) and SMS (Short Messaging Service) networks allows users to seamlessly use a variety of computing devices from desktops to cellular phones and mobile handhelds. However, this increasing convergence has also attracted the attention of malicious software writers. In the past few years, the number of malicious codes that target messaging networks, primarily IM and SMS, has been increasing exponentially. Large message volume and number of users in these networks renders manual mitigation of malicious software nearly impossible. This paper proposes *automated* and *proactive security* models to protect messaging networks from mobile worms and viruses. First, we present an algorithm for automated identification of the most vulnerable clients in the presence of a malicious attack, based on interactions among the clients. The simplicity of our approach enables easy integration in most client-server messaging systems. Next, we describe a proactive containment framework that applies two commonly-used mechanisms—rate-limiting and quarantine—to the dynamically-generated list of vulnerable clients in a messaging network whenever a worm or virus attack is suspected. Finally, we evaluate the effectiveness of proactive security in a cellular network using data from a large real-life SMS customer network, and compare it against other existing approaches. Most messaging networks can implement our proposed framework without any major modification of their existing infrastructure.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

General Terms

ALGORITHMS, SECURITY

Keywords

Proactive security, Instant Messaging (IM), SMS/MMS, mobile viruses, worms, containment

1. INTRODUCTION

The exponential growth of messaging in both home and enterprise environments has made it a potent vector for the spread of malicious code [1]. Social engineering techniques are very effective in spreading malware in these networks since infected messages appear to come from addresses in personal contact lists, address and phone books. The problem is compounded further by the increasing convergence of various messaging platforms. For example, users can now send IM messages from mobile phones, and SMS messages to mobile phones via SMS gateways on the Internet. Given the extremely large number of messages in public IM and SMS networks,[1] the potential for damage from rapidly propagating malicious software is very high in messaging networks. This has not escaped the attention of malicious code writers. According to [1], self-propagating worms represented 91% of malicious code in large public IM networks in the second half of 2005—a number that has been steadily rising. Similarly, there are now a growing number of malicious codes written for mobile handsets that exploit SMS/MMS to proliferate [2]. It is clear that if a response can be taken in the early stages of an epidemic in these networks, the spread can be limited to a small number of clients. Therefore, developing *proactive security* frameworks in mobile messaging networks is an important area of research. However, most mobile network operators and messaging providers have not implemented proactive security for the following reasons.

A key aspect of proactive security is to take steps *before* a client is compromised or at the earliest indication of a virus or worm activity in the network. Therefore, finding vulnerable clients to a given malicious software is a key first step to any proactive security strategy. Note that this step must be entirely automated or the window of opportunity will be lost. Given the large number and distributed nature of messaging networks, it is not possible to place monitors everywhere in such networks. However, the messaging server—the Short Messaging Service Center (SMSC) in case of SMS/MMS messages, and the IM server—provides a natural way to identify such clients as we explain later.

One may argue that the time window between detection and proactive containment can be very small and no proactive action can stop a fast-spreading malicious code. For example, it is theoretically possible to have "Flash Worms" [3, 4] that can infect most of the vulnerable hosts of an enterprise within seconds. While such attacks are possible, there has been a noticeable decrease in malicious

*This work was performed while the author was at the University of Michigan, Ann Arbor, Michigan, and was supported in part by NSF Grant No. CN 0523932.

[1] More than 1000 billion SMS messages were sent in 2005, and according to [1], the three largest IM providers—AOL, MSN, and Yahoo!—each accounted for over 1 billion IM messages sent per day.

agents that spread very fast via random scanning and simply clog corporate networks. However, there has been a steady increase in stealthy Trojans, and malicious agents that install adware and spam relays, exploit enterprise applications such as database servers, and host malicious websites. For example, Win32.Opanki.d [5] arrives as a link via the AOL IM network and when executed, it opens a backdoor via an IRC channel. For these emerging threats, discovering *group associations* with an already-infected or suspected client in near real time will lead to better proactive containment and it is an important focus of our work.

Finally, any proactive response must address the potential loss of service and delays in the messaging network due to preemptive shutdown or policing of clients. Since it is common for anomaly detection systems to generate many false positives, a straightforward quarantine of clients based on alerts may result in unacceptable levels of message loss and delay. Therefore, one must design proactive strategies that increase the level of countermeasure with increasing alert correlation.

Most of the published studies on modeling and containment of malicious software have focused on scanning- and email-worms due to their prevalence and several successful large-scale attacks on the Internet. On the contrary, there appears to be very little published work on proactive security of messaging networks. This is the primary motivation of our work. In this study, we would like to achieve three primary goals: (i) automated compilation of the list of messaging clients that are vulnerable to a spreading virus or worm attack, (ii) development of a group-behavior-based proactive response framework using client interactions in a messaging network, and (iii) compare the effectiveness of proactive response with traditional reactive mechanisms (e.g., anti-virus tools). The starting point of our study is observed interactions among clients comprising the "service-behavior" topology of the messaging network. The containment itself is implemented in the form of *client rate-limiting* [6] (also known as "throttling") and *client quarantine*. However, instead of a straightforward application of these mechanisms, we build a behavioral alert-based system that progressively mounts a stronger response with increasing alerts, and backs off when alert levels decrease with time.

The framework is implemented typically at the messaging service center (i.e., SMSC in case of SMS/MMS and IM servers) where logs of client communication are available. These logs can be analyzed to generate a service-behavior graph for the messaging network. It is then further processed to generate behavior clusters, i.e., groups of clients whose behavior patterns are similar with respect to a set of metrics: *interaction frequency, attachment and message size distributions, number of messages, number of outgoing connections to other clients* and *list of traced contacts*. When the number of alerts in a particular behavior cluster reaches a threshold, the messages belonging to that behavior cluster are first rate-limited to slow down a potential malicious worm or virus. The effect of repeated similar alerts and false positives is kept below a threshold during this initial containment step. When the alerts reach a second threshold, the containment algorithm applies proactive quarantine, i.e., it blocks messages from suspicious clients of these behavior clusters. This step essentially enables the behavior clusters to enter into a group defense mode against the spreading malware. This combined approach of rate-limiting and quarantine with increasing response to alerts in the network provides a graceful service degradation, yet a very powerful defense as our evaluation will show.

This paper makes three primary contributions. First, it presents a method for automated identification of vulnerable clients in a messaging network. Second, it provides a practical solution for improving security in these networks based on an adaptive group-behavior-based proactive approach. Third, it demonstrates that proactive security can offer an order-of-magnitude improvement in containing malicious software in messaging networks, over existing "detect-and-block" approaches.

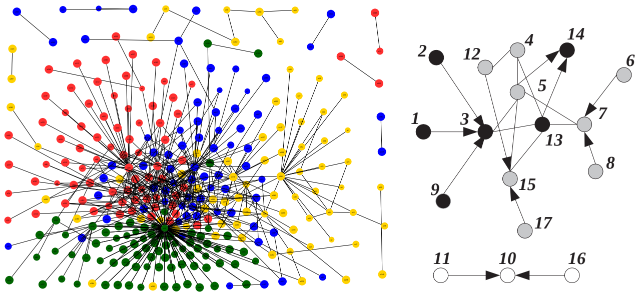
The rest of this paper is organized as follows. Section 2 presents motivations behind the behavior-clustering approach. Section 3 describes how behavior clustering can find vulnerable clients for proactive response. Section 4 describes the proactive rate-limiting and quarantine algorithms, and their group-behavior-based implementation. Section 5 evaluates proactive security in a messaging network using data from a large real-life SMS customer network. Section 6 reviews recent literature on malware targeting messaging networks, and malware containment. We describe future work and make concluding remarks in Section 7.

2. MOTIVATION: FINDING VULNERABLE CLIENTS

The most common form of proactive defense is *getting there first*, for example, to patch a client to protect against an existing vulnerability, or to remove capabilities from the client, making it more secure. However, a central problem of proactive defense is to decide which clients are the most vulnerable when a malicious activity is identified in the network, be it an intrusion, a virus or worm. This is fundamentally different from reactive or "detect-and-block" defense which is activated only when a client is in the process of being compromised. Generating a list of vulnerable clients on-demand in near real time is, therefore, a fundamental problem to study in proactive defense. The more accurate the list of vulnerable clients, the faster the attack can be suppressed with less interruption to the users. Our motivation in studying group-behavior of clients is to generate this list by analyzing Charging Data Records (CDRs) [7] and message headers that are logged at the centralized store-and-forward messaging servers.

Before discussing our approach, we first need a brief discussion of the SMS messaging system. When a mobile user sends a message from a handset (i.e., Mobile Originated or MO) or a web-based gateway to another phone, the message is received by the Base Station System (BSS) of the service provider. The BSS then forwards the message to the Mobile Switching Center (MSC). Upon receiving a MO message, the MSC sends the end-user information to the Visitor Location Register (VLR) of the cell and checks the message for any violation. It then forwards the message to the provider's SMSC. The SMSC stores the messages in a queue, records the transaction in the network billing system and sends a confirmation back to the MSC. The status of the message is changed from MO to Mobile Terminated (MT) at this point. Through a series of steps, the message is then forwarded by SMSC to the receiving user's MSC. The MSC receives the subscriber information from the VLR and finally forwards the message to the receiving handset. The store-and-forward nature of SMS/MMS networks makes it possible to collect client interaction patterns from the time-stamped logs. A similar observation can be made for IM servers as well, although the procedure to store and forward messages is much simpler. Most IM messages between users are mediated by the IM server. In some networks, file-transfer request and response messages are relayed through the server, but the actual file data are transferred between the entities directly. This makes the task of collecting information about client interactions very easy by simply monitoring the connection logs at the server.

Next, we provide the motivations for development of a group-behavior-based proactive defense strategy. Traditional end-point solutions (e.g., switching off service ports on individual clients or at firewalls) can detect and protect against only specific types of attacks. A more effective defense can be built by studying how clients interact with each other in the network from periodic inspection of the server logs. If clients can be grouped together based on their *common* behavior, it may be possible to contain a broad range of attacks that manifest in specific behavior anomalies. The building block of our approach is finding clusters of such common behavior called "behavior clusters." Once a virus or worm activity is detected at a client, members of its behavior cluster can be put on the list of vulnerable machines since they may be the most likely and immediate target of the malicious activity. This is often the case for topological worms that spread via IM, email, SMS and P2P file-sharing.



(a) **(b)**

Figure 1: (a) Clustering of common behavior, (b) Microscopic view

We motivate the usefulness of behavior clusters with a simple example. We collected messages from a small departmental network of 200 unique hosts, and constructed a service-behavior topology based on open client and server port bindings among the hosts (Figure 1(a)). Figure 1(b) shows the service-behavior graph for a small subset of nodes at a given time—the actual number of nodes (504, including nodes external to the network) and edges (230) are too large to display in both figures. We show a 4-color clustering of this subset of nodes. The arrows in Figure 1(b) indicate the directionality of messages as inferred from the traces. The nodes without any arrows denote bidirectional messages. To generate the clusters, we considered a simple metric of number of neighbors, and minimized the overlap among the four clusters. Note that nodes 10, 11 and 16 form a disjoint group from the rest of the nodes and have no interactions with the rest of the network, other than their internal dependencies. These nodes can be grouped into a single behavior cluster (denoted in white circles). Upon detection of a malicious software on any one of these three hosts, one can proactively quarantine the other two nodes without affecting messaging in the other clusters. On the other hand, if one quarantines the cluster with nodes (denoted in black circles) completely from the rest of the network, the total cost of quarantine will be the sum of messages exchanged along overlapping edges $e_{3,5}, e_{4,13}, e_{7,13}, e_{13,15}$, and $e_{5,14}$. Note that the three clusters form a logical partition of the network. Each cluster provides a list of vulnerable clients any time a client inside the cluster raises an intrusion or malware alert. We give a more formal treatment of the behavior clustering and partitioning problem in Section 3.2.

3. FINDING VULNERABILITY BY ASSOCIATION

We mentioned in Section 2 that the first step in applying any proactive mechanism is to find a set of vulnerable clients in near real time, i.e., as soon as an attack is detected. In this section, we propose an approach called *behavior clustering* to generate this list. The underlying principle of behavior clustering is to find vulnerability by association. It assumes that the vulnerability index of a client is increased sharply if it has come in *contact* with an infected client in recent past. By *contact*, we mean messages exchanged between the two clients, e.g., a text or multimedia message. Whether a client is infected by the way of such communication with an infected client depends on whether it shares the same vulnerability. Therefore, our goal is to develop an automated procedure to cluster clients into behavior groups based on their messaging patterns and application/protocol stacks installed on them. The rest of this section describes the steps necessary to generate these clusters.

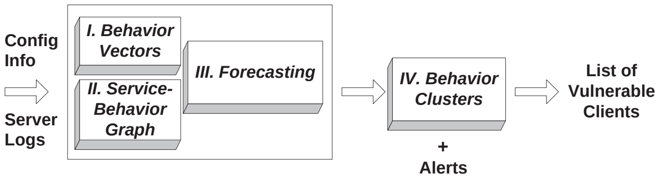


Figure 2: Generating behavior clusters from message logs

Figure 2 shows the three steps necessary for automated behavior clustering: (i) calculation of *behavior vectors* and service-behavior graph, (ii) short-term forecasting of behavior vectors, and (iii) generation of behavior clusters by partitioning the service-behavior graph. These steps are repeated periodically depending on how often the behavior vectors change among clients, and the outcome is a set of closely-related behavior clusters for the network that can be used to find vulnerable clients upon detection of an attack.

3.1 Step I: Behavior Vectors

We define a "behavior vector" as a collection of features about any client in the messaging network. The behavior vector, denoted as $\theta_u(t)$ at any client u at time t, is calculated from two sources: version information ('physical' feature) and messaging logs ('temporal' features). Most malware spread by taking advantage of known exploits in software and protocol stacks. Therefore, an accurate snapshot of how clients are configured across a network is very useful to determine which clients are vulnerable to a spreading malicious software. Enterprise networks typically install configuration management databases (CMDBs) [8] that contain details of the applications (email, P2P, IM, SMS) and software stack (OS, network) on each host. Queries to CMDBs can therefore yield the physical feature of the behavior vector at a host. We collectively denote the physical feature space as ϕ_c. This feature space can be partitioned to find clusters of similar configurations. Then, whenever a virus or worm is discovered targeting a specific application client or software exploit, one can readily find the most vulnerable clusters where a proactive action is needed. In public IM or SMS networks, it is not possible to access client OS and application stack information. However, most messaging clients transmit client version information and a few additional details about the client environment (e.g., Windows or UNIX) during the connection setup. This information can be extracted from the server logs where access to enterprise-level CMDBs is not possible.

The second component of a behavior vector is calculated by analyzing messages exchanged among the clients, and therefore, it is a temporal feature. The generic parameters that we have implemented are: CDF (cumulative density function) of neighbor interactions (n_m) ("how often a client exchanges messages with another client"), number of outgoing connections to unique user IDs (n_g) ("importance of a client"), and mean and maximum of message inter-arrival times (t_{mean}, t_{maxm}).

In summary, the vulnerability index of a client in the messaging network depends on its physical and temporal features, or, in short, its *behavior vector*. The components of the behavior vector at a client u at time t are given as:
$\theta_u(t) = [\{\phi_c\}, \{n_m, n_g, t_{mean}, t_{maxm}\}]$.
This vector is updated whenever their values change based on filters placed on the server.

3.2 Step II: Service-Behavior Graphs

While behavior vectors represent client-level observations as logged by the server, they do not describe interactions among the clients. This is captured by creating a service-behavior graph for the network. We represent the service interactions with a *directed graph*, $G(V_d, E_d)$, in which V_d is the set of vertices (i.e., unique participants or client IP addresses) in the network and E_d is the set of edges. $G(V_d, E_d)$ is generated by applying the following simple rules.

R1. A pair of vertices $(u, v) \in V_d$ are assigned a *directed edge* $e_{uv} \in E_d$ if and only if there exists a non-zero contribution to their respective behavior vectors via e_{uv}.

R2. IP addresses that are external to the networks are labeled with an additional flag. The edges belonging to these hosts represent "outside" connections to the network and therefore, should be quarantined during an attack. Examples are http links embedded in messages.

3.3 Step III. Short-term Forecasting of Behavior Vectors

Since behavior vectors of clients change frequently in most messaging networks, logs collected at different time intervals may indicate different behavioral patterns. Therefore, the service-behavior graph, generated at fixed time intervals, may differ from the actual behavioral patterns of the network when the list of vulnerable machines need to be generated. Therefore, a proactive action based on a straightforward application of behavior vectors computed in the last analysis period may not be the most effective. Since behavior vectors have a strong temporal component, we apply a short-term forecasting algorithm to the parameters such that a prediction can be made from the observed values in recent past. This is currently achieved by applying the standard exponential smoothing procedure [9].

3.4 Step IV. Behavior Clustering

The final step is to group the vertices in $G(V_d, E_d)$ into a number of clusters based on their behavior vectors, where clients in the same cluster are similar in terms of their physical and temporal patterns. There are a number of techniques for classification and clustering in the literature. We adopt a hierarchical graph partitioning approach as presented below, although other approaches can also be used.

The partitioning problem can be formulated as a multi-constraint, connected and bounded k-way graph partitioning problem as follows. Given an undirected graph of service interactions, $G(V_d, E_d)$ with scalar edge weights $w_e : E_d \to N$, each vertex $v \in V_d$ having an n-dimensional behavior vector $\theta_v^{(n)}$ of size n ($\sum_{\forall v \in V_d} \theta_v^{(i)} = 1.0$ for $i = 1, 2, \cdots, n$), and an integer $b \in \{2, 3, \cdots, \|V_d\|\}$, partition V_d into k clusters, $V_d^1, V_d^2, \cdots, V_d^k$, such that

- $G_i = (V_d^i, E_d^i)$ induced in G by the i-th cluster is connected;
- $\forall i \in \{1, 2, \cdots, k\}$: $1 \leq \|V_d^i\| \leq b$;

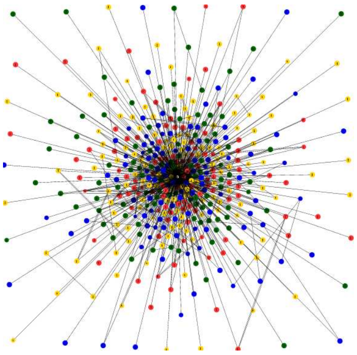


Figure 3: Behavior clustering of an IM network $(k = 4)$

- $\sum_s w_{(e)}(s)$ where $s \in E_d, s \notin E_d^i$, is a minimum $\forall k \in \{2, 3, \cdots, \|V_d\|\}$; and
- the following constraints are satisfied:

$$\forall k : \ell_i \leq \sum_{\forall v \in V_d^k} \theta_v^{(i)} \leq u_i \qquad (1)$$

where $[\ell_i, u_i]$ for $i = 1, 2, \cdots, n$ are n intervals such that $\ell_i < u_i$ and $\ell_i + u_i = 1$.

Note that the number of clusters, k, is not provided as input to the above problem, and therefore, must be evaluated as the number of distinct behavior clusters in the graph. As an example of Steps I-IV, Figure 3 shows the partitioning of an IM network of 450 clients (i.e., unique IP addresses) into four behavior clusters $(k = 4)$ based on traces we collected from a large enterprise network. The IM users of this network used three public-messaging protocols—Yahoo Messenger (YMSG), MSN Messenger (MSNMS) and AIM—to communicate with each other. Therefore, ϕ_c for a host consisted of one or more elements of $\{$YMSG, MSNMS, AIM$\}$, depending on which IM protocols were used from that host. The rest of the behavior vector parameters were calculated directly from the traces. This approach to behavior clustering offers several benefits when a proactive response is taken in the network.

(1) Connectedness among the vertices within a cluster: This property guarantees that any two vertices within a cluster be closer to each other in terms of their features and connectivity than vertices in another cluster. This is important for localizing messages within a cluster while other clusters are proactively contained, so that direct peer-to-peer file transfers between clients are always available.

(2) Minimization of service-edge costs: The cost of the cut (called "edge-cut") determines the quality of the clusters, and is, therefore, the primary partitioning objective. There are many possible choices for the partitioning objective function. For the containment problem, we minimize the sum of the weights of the edges that span multiple clusters. The goal is to minimize the number of messages exchanged between different clusters. Then, any proactive quarantine or rate-limiting of a cluster will cause minimal message interruption to other clusters.

(3) Satisfaction of vertex constraints within the clusters: The k-way partitioning algorithm takes into account the relative weights of the vertices as well as those of the corresponding edges. The constraints as shown in Eq. (1) can be used to balance the partitions in terms of the vertex constraints, e.g., for including the clients' geographic domains.

An important deployment question is how often the service-behavior graph should be updated. We can apply the *triggered updates* concept implemented in many intrusion detection systems, e.g., GrIDS [10].

Using triggered updates, the service-behavior graph is updated whenever (i) new vertices and edges are added (or subtracted) to (or from) the last computed graph, and (ii) the parameters of the behavior vectors change by a certain threshold over previous values. This is part of our ongoing work in which we are studying logs collected from a real-world messaging server to understand the temporal aspects of service-behavior graphs.

The overall complexity of the partitioning phase is $O(\|E_d\|)$, and therefore, is determined by the size of messaging network, G. In reality, this step is extremely fast. For example, the time required to generate behavior clusters for a service-behavior graph with $V_d = 9269$ hosts and $E_d = 9836$ edges ranges from 0.04 second (2 clusters) to 0.16 second (32 clusters) on a dual-CPU (1.5GHz) AMD Opteron 240 platform.

4. PROACTIVE CONTAINMENT METHODS

In this section, we explain the basic rate-limiting and quarantine mechanisms that serve as the building blocks of our proactive response framework. While scan detection-based methods [11, 12] protect an enterprise from incoming infections, rate-liming and quarantine seek to contain outbound infected messages. These methods can be applied on both individual as well as a group of clients. When these are applied on a group of clients as in the case of proactive defense, the first step is to obtain a list of vulnerable clients most relevant to the generated alerts. We assume that this list can be obtained on-demand via the behavior clustering algorithm described in Section 3.

4.1 Rate-limiting

The rate-limiting (also known as “virus throttling”[6, 13, 14]) is a general class of response techniques that seek to limit the spread of a worm or virus once it is detected on a host. For example, it has been applied to contain IM worms in [13]. It is based on the observation that normal or acceptable behavior of many Internet protocols such as TCP/IP, email and IM differs significantly from the corresponding worm-like behavior. Most users of email, SMS and IM interact with a slowly-varying subset of other users as compared to malicious codes that attempt to send messages to all contacts in a victim’s address book or buddy list. The original virus throttling algorithm proposed by Williamson [6] limits the rate of outgoing connections to new machines that a host is able to make in a given time interval. Figure 4 explains the basic rate-limiting mechanism. A *working set* of specified length ($n = 4$ in Figure 4) is maintained for each user that keeps track of n recent addresses that the user has interacted with. When the user attempts to send a message to a new contact, the recipient’s address (“h” in Figure 4) is compared with those in the working set. If the address is in the working set, the message is allowed to pass through. Otherwise, the message is placed on a *delay queue* for sending at a later time. At periodic intervals, the delay queue messages are processed as follows: the destination address of the message at the head of the queue is added to the working set replacing the oldest address in the working set (using a least-recently used or LRU algorithm). Then, all messages in the delay queue destined for the newly-admitted address are removed from the queue and sent to the recipient address. When the length of the delay queue exceeds a pre-determined threshold, all new contact attempts from the client can be blocked, e.g., by reducing the size of the working set to zero, and the user may be asked to validate the messages in the queue. The rate-limiting mechanism is implemented at the server since it initiates or processes all requests made by the clients. Further, when implemented at the server, users are not able to modify the rate-limiting configuration parameters. Therefore, rate-limiting can be implemented easily for email, IM, SMS and centralized P2P file-sharing (e.g., Napster). The most important advantage of rate-limiting is its ability to enforce containment in a gentle manner, as opposed to quarantine which results in complete shut-down of the client. Therefore, rate-limiting mechanisms are generally preferred by enterprise networks over quarantine. We should note that several variants of rate-limiting have been proposed to date. Recently, Wong *et al.* [14] have presented an excellent empirical study of these schemes as well as a new DNS-based rate-limiting algorithm for general worm containment.

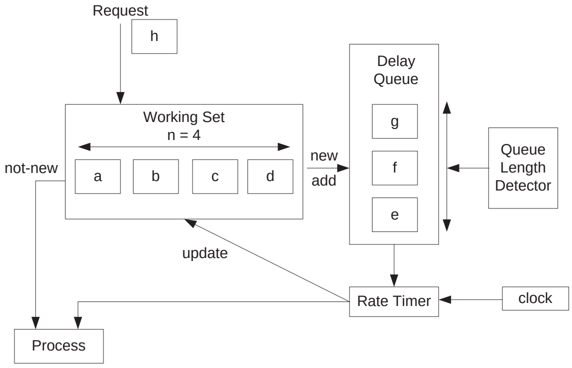


Figure 4: Virus throttling algorithm by Williamson [11]

For our implementation of proactive rate-limiting, we chose to implement the Williamson’s throttling algorithm as the basic per-client rate-limiting mechanism, with an important difference. We implement rate-limiting only for messaging services, and not the other ports on the host. This prevents the excessive delays and blockage of all legitimate applications on the host in case of an infection, as reported in [14]. Note that the messaging worm propagation doesn’t result in large volumes of failed connections or data in the network. But, the rate at which an infected client sends messages to other clients in the network may deviate significantly from its normal sending rate. This can be detected efficiently by the Williamson’s virus throttling algorithm although the algorithm is prone to high false positive rates when a client has been infected. We deal with this problem by applying a group-behavior-based approach that effectively reduces the false positive rates.

4.2 Quarantine

In contrast with rate-limiting, quarantine-based systems prevent a suspicious or infected client from sending or receiving messages. This can be implemented at the messaging server so that any connection attempt by the user on an infected client is refused. Recent industry initiatives such as Network Admission Control (NAC) [15] and Network VirusWall [16] are intended to enforce established security policies to endpoint devices as they enter a protected network. The Cisco NAC allows non-compliant devices to be denied access and placed in a quarantined local network, or given restricted access to resources. However, such systems are in very early stages of development for SMS/MMS networks.

In our implementation of quarantine, we simply reduce the size of the working set to zero in the rate-limiting module on a client and let the delay queue grow without triggering any new malicious software alert. This is enforced *after* an alert has already been issued from the client and a malicious activity has been detected. This effectively quarantines the client from sending any more messages.

Next, we propose a proactive group behavior containment (PGBC) algorithm that combines the basic rate-limiting and quarantine mechanisms described above with behavior clusters to develop a proac-

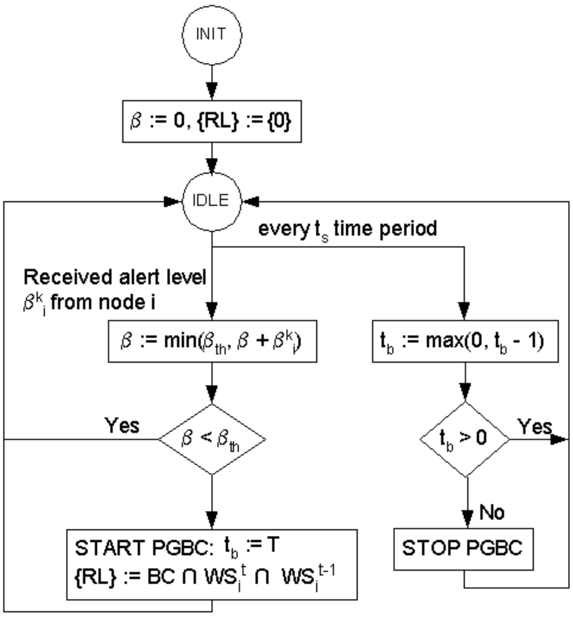

INIT
β := 0, {RL} := {0}
IDLE
every t_s time period
Received alert level β^k_i from node i
β := min(β_{th}, β + β^k_i)
t_b := max(0, t_b - 1)
Yes
$\beta < \beta_{th}$
$t_b > 0$
Yes
No
START PGBC: t_b := T
{RL} := BC ∩ WS_i^t ∩ WS_i^{t-1}
STOP PGBC

Figure 5: Proactive rate-limiting and quarantine for a behavior cluster

tive response scheme at the messaging server.

4.3 Proactive Group Behavior Containment

Figure 5 presents the steps of the PGBC algorithm as implemented in the server. When an anomaly is detected at a client *i*, e.g., by monitoring its delay queue length (or via a malware detection agent running at the server), the algorithm increments a client alert level (δ_i) by a value δ_i^k that depends on the severity of the alert. The PGBC algorithm suppresses alerts for a period of π_{delay} seconds before allowing a single alert β_i^k for client *i* in the algorithm. The purpose of the hold-off counter, π_{delay}, is similar to the back-off counter described in [17]: it prevents a single client that triggers a stream of alerts from forcing the entire messaging network to enter into a proactive defense mode. When the alert level on a client violates a pre-determined threshold value (i.e., β_i^k is reached), the server activates a rate-limiting for messages sent by the client, i.e., the size of its working set is reduced and outgoing messages from the client are queued at the server. A separate process decrements the alert level at every time step until it reaches zero, at which point, the rate-limiting is stopped for messages sent by the client.

The messaging server generates behavior clusters at periodic intervals from the messages exchanged among the clients, using the clustering algorithm described in Section 3. Whenever an alert level β_i^k is generated for a client, the algorithm updates the total alert level β of the corresponding behavior cluster. When the behavior cluster alert level reaches a threshold value (β_{th}), the server activates a rate-limiting on the most vulnerable clients of the behavior cluster, namely, the nodes that have exchanged messages with the infected client. This list is computed via a set *intersection* of the client in the behavior cluster and the working sets of the infected client at current and previous time steps. This step of the algorithm also enforces a quarantine of all messages from the infected client (i.e., it is no longer rate-limited but is blocked from messaging). A separate process decrements a backoff timer (t_b) from the value of *T* assigned at the beginning of the group defense mode, and transitions the behavior cluster from the proactive defense mode back to the normal mode when either (i) there are no more alert messages or (ii) t_b becomes 0.

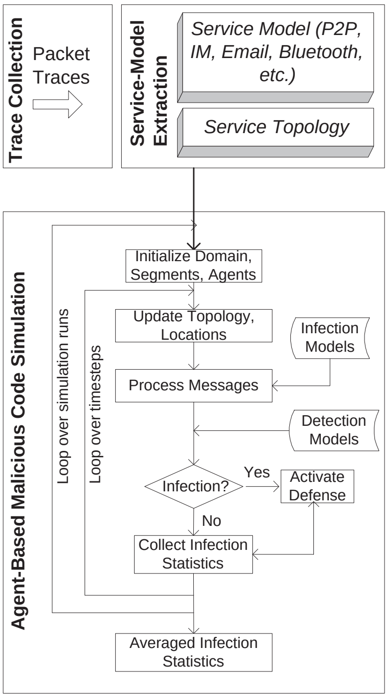

Trace Collection
Packet Traces
Service-Model Extraction
Service Model (P2P, IM, Email, Bluetooth, etc.)
Service Topology
Agent-Based Malicious Code Simulation
Loop over simulation runs
Loop over timesteps
Initialize Domain, Segments, Agents
Update Topology, Locations
Infection Models
Process Messages
Detection Models
Infection?
Yes
Activate Defense
No
Collect Infection Statistics
Averaged Infection Statistics

Figure 6: Flowchart of AMM emulation steps

Note that PGBC gradually slows down outgoing messages from a group of clients and brings them back to the normal mode when no alerts are received for a period of time. This is in contrast with the traditional detect-and-block schemes that cause sudden message loss and delay in the network.

5. EVALUATION OF PROACTIVE DEFENSE IN AN SMS NETWORK

5.1 Agent-Based Malware Modeling

In this section, we briefly discuss an agent-based malware modeling (AMM) framework that we have developed to investigate malicious software propagation in a variety of wired and wireless networks. We refer to [18] for a detailed description of AMM and its various capabilities. In AMM, we model a mobile network as a collection of autonomous decision-making entities called *agents*. The agents represent clients within the network such as PDAs, mobile phones, service centers(e.g., SMS Center) and gateways. In case of agents representing mobile devices, the connectivity changes as users roam about the physical space of the network. The behaviors of the agents are specified by a set of services running on them. For example, an agent may consist of client applications for email and SMS/MMS messaging, whereas the SMSC agent may consist of a store-and-forward server only. Thus, there are two types of topologies in our simulation environment. The *physical*connectivity is determined by the physical network infrastructure, movement of the agents, location of access points and base stations, whereas the *logical* connectivity is determined by the messages exchanged among the agents.

Figure 6 shows a high-level flowchart of AMM. The first step is to prepare the following input parameters: (i) *infection-model parameters* for the target service (SMS for the present study), (ii) *topology* of service interactions among the agents (i.e., SMS messaging patterns among the users), (iii) *location* of base stations, (iv)

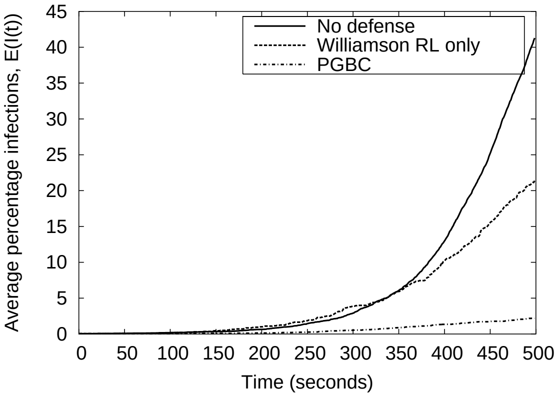


Figure 7: Overall performance of PGBC and WRL

mobility models for agents that are mobile, (v)*infection and replication state machine* (i.e., attack vector) of the malicious software, (vi) *detection model* and (vii) *an attack response model*. We have implemented PGBC, Williamson's rate-limiting (WRL) and reactive (i.e., detect-and-block) responses to compare their effectiveness. At each timestep, the coordinates of mobile agents are updated based on their mobility models resulting in new connectivity graphs. Next, each agent exchanges messages with other agents according to the SMS service model—the probability of any of these messages being infected is calculated from the service-infection model. The time steps are repeated over a user-specified number of trials so that the results can be averaged over these trials. The simulator is general enough to experiment with different algorithms for malware detection and containment.

5.2 SMS Messaging Logs

The topology of a messaging network can be extracted from CDRs collected from a real-world cellular network, and the relevant parameters are then input to AMM. To the best of our knowledge, there does not exist any malware propagation model using SMS/MMSservices in a cellular network. A recent study [19] of SMS usage characterization collected call data records and SS7traces over a three-week period from a large cellular carrier with 10 million mobile users. The data allowed us to reconstruct a realistic SMS messaging network with the following parameters: message sending rates, message receiving rates, cumulative density functions (CDF) of user-to-user message size (B) and message service times at the SMSC, and the SMS service topology. The original data involved a very large number of messages (over 59 million) and users (over 10 million). Therefore, we scaled the data to a small number of users (2000), while still preserving the basic characteristics of the original data sets. Further information on our reduction process and SMS malware propagation can be found in a companion paper [2]. In the present study, we focus on the proposed proactive defense strategy.

5.3 Performance of PGBC

In what follows, we evaluate and compare three different defense strategies, Williamson's rate-limiting (WRL), reactive (i.e., detect-and-block) and PGBC, against a malicious code that spreads from one client to another using the address book contacts found on an already-infected client.[2] We used a working set of 5 for WRL and a delay queue threshold of 20 to indicate a malicious code infection. All simulations started with only 1 initial infection (i.e., $I(0) =$ 1) and we averaged the results of 20 runs for each parameter to compute the expected number of infections, $E(I(t))$, clients with WRL, $E(RL(t))$, and clients under quarantine $E(Q(t))$, at each time step t. We report these numbers as a percentage of the total number of clients in the messaging network.

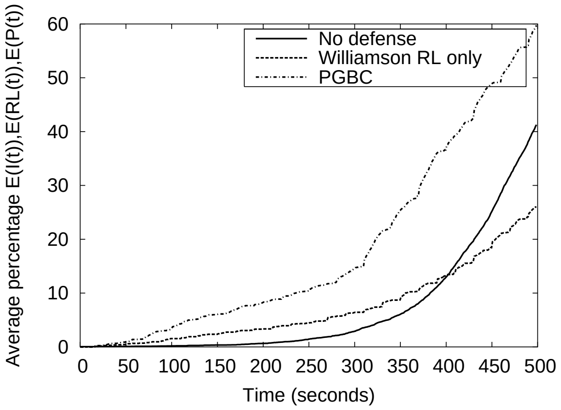


Figure 8: Percentage of clients rate-limited (WRL) and proactively-contained (PGBC)

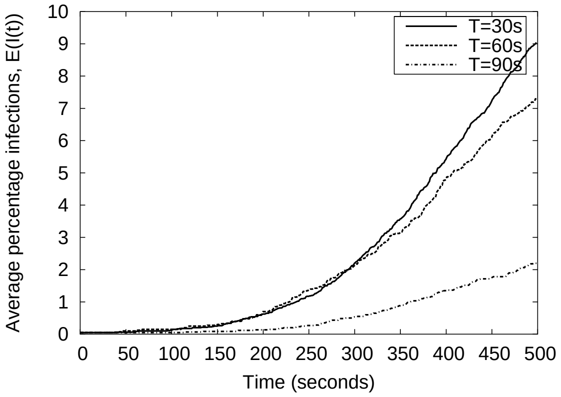


Figure 9: Effect of PGBC backoff timer ($\beta_k = \beta_{th} = 8$, $r = 30s$)

5.3.1 Overall Performance

Figure 7 shows the overall performance of WRL and PGBC against the SMS worm during the first 500 seconds of its propagation. For comparison, we also show the epidemic in a network with no defense, i.e., an entirely unprotected network. Note that in the unprotected network, nearly 40% of the clients are already infected, indicating very fast propagation in a highly-connected topology such as an SMS network. The PGBC algorithm performs an order-of-magnitude ($E(I(t)) = 2\%$) better than WRL ($E(I(t)) = 21\%$). Given this excellent performance, we now explore the effect of various PGBC parameters on its performance. Figure 8 shows the percentage of clients that are rate-limited ($E(RL(t))$) using WRL, proactively-contained ($E(P(t))$) in PGBC and infected ($E(I(t))$) in case of an unprotected network. PGBC results in a larger number of clients participating in proactive group defense, roughly 20% more than WRL allows—this is one of the reasons why PGBC keeps the infection level so small in the network. However, when proactive group defense is applied, the affected messages are delayed by an amount equal to the PGBC backoff timer, T.

5.3.2 Effect of PGBC Backoff Timer, T

The PGBC backoff timer (T) determines how long a behavior

[2] We will denote this as the "SMS worm" for the rest of this section.

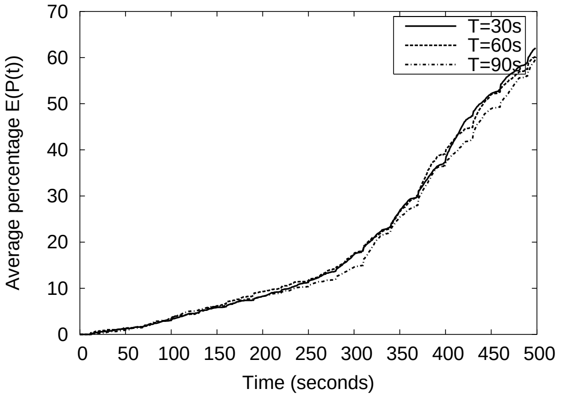


Figure 10: Number of proactive clients for different PGBC backoff timer values ($\beta_k = \beta_{th} = 8$, $r = 30s$)

cluster should remain in the proactive defense mode once it has been activated (i.e., its alert level, β, has reached the threshold β_{th}). Figure 9 shows the percentage infections for different values of $T = 30$, 60 and 90 seconds. To eliminate the effect of other parameters, all the results are for $\beta_k = \beta_{th} = 8$ and a throttle rate (r) for 30 seconds. It is clear from Figure 9 that a longer group defense timer results in a smaller number of infections. However, this has a delaying effect on the messages in the SMS network since clients undergoing proactive rate-limiting will have to wait until the expiration of T seconds before sending messages again. Note that $T = 60$ seconds or a minute results in approximately 7% of infections compared to a longer timer (2%). Since the overall infections are still much lower than that of an unprotected network as well as a network with WRL, we recommend using $T = 60$ seconds—a delay of 1 minute in sending/receiving SMS messages is reasonable during an attack. Figure 10 shows the percentage of proactive client nodes for different values of T. The growth of proactive clients in the group defense mode is nearly identical for all three values, meaning that the decision to choose T should be guided by the maximum level of infections that can be tolerated by the messaging service provider. For example, critical messaging networks may decide to choose a higher value of T to keep $I(t)$ very small.

5.3.3 Effect of Alert Levels, β_k

The alerts (β_k or "L" in Figures 11 and 12) are generated for individual clients when their delay queues reach a threshold indicating the presence of a worm. However, the hold-off counter (π_{delay}) makes sure that repetitive alerts from the same client in consecutive periods are suppressed so that one client node cannot force a behavior cluster to enter the PGBC mode. The choice of β_{th} determines how frequently a behavior cluster enters PGBC given β_k alerts sent from its constituent clients. Figure 11 presents the results for different alert levels β_k equal to $2, 4$ and 8. The rest of the parameters are: threshold alert level $\beta_{th} = 8$, backoff timer $T = 60$ seconds and throttle timer $r = 60$ seconds. We also plot the same results for a different throttle timer of $r = 20$ seconds in Figure 12. The results indicate that client alert levels (β_k's) are sensitive to the throttle timer, especially at low alert levels. Based on our experimental results, we recommend setting $\beta_k = 4$, i.e., two new alerts force the behavior cluster enter into PGBC mode.

5.3.4 Comparison of Reactive vs. PGBC Approaches

Next, we compare the traditional reactive ("detect-and-block") defense adopted by most anti-virus solution providers with a more efficient approach such as PGBC. In reactive defense, a client can be quarantined as soon as its delay queue reaches the threshold, i.e., all incoming/outgoing messages are blocked. The problem with this approach is that by the time the infection is detected, the worm may have already spread to other clients via previously sent messages. Figure 13 compares the reactive approach with two different combinations of PGBC parameters. The reactive approach results in over 12% of the total clients being infected within 500 seconds, whereas either of the two PGBC configurations allows only less than half that amount. This clearly demonstrates the need for a group-behavior-based defense strategy in messaging networks than isolated quarantine of clients.

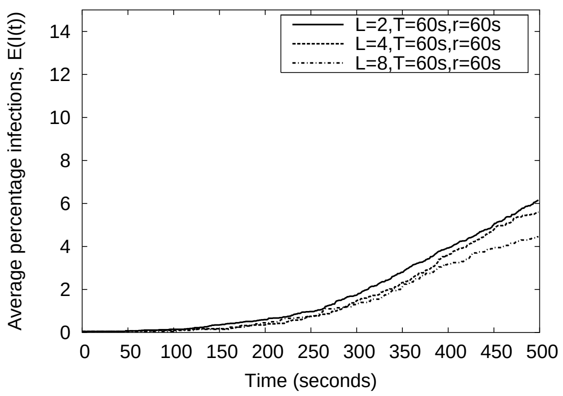


Figure 11: Number of infections for different PGBC alert levels ($T = 60s$,$r = 60s$)

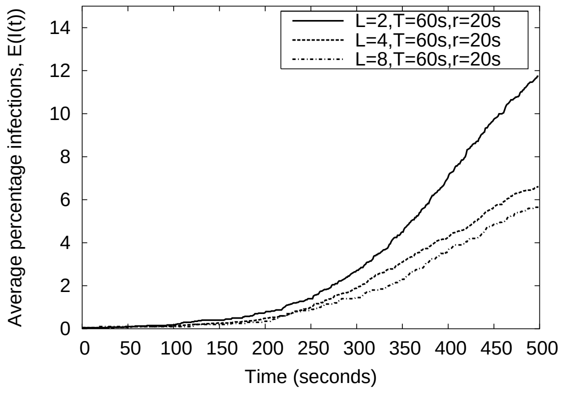


Figure 12: Number of infections for different PGBC alert levels ($T = 60s$,$r = 20s$)

5.3.5 Effect of False Positives

We define the false positive rate as the percentage of clients that were misidentified as malicious by the detection mechanism. Since PGBC applies proactive rate-limiting to some of the clients in the same behavior cluster as the false-positive clients, we calculate two quantities: percentage of false-positive clients (denoted by $E(F(t))$) and percentage of clients that were proactively rate-limited (denoted by $E(P(t))$)due to $E(F(t))$. Figure 14 plots $E(P(t))$ and $E(F(t))$ for different PGBC backoff timer values. From our results, we found that $E(F(t))$ was limited to 6% of the clients, and $E(P(t))$ to 12% of the clients in most cases. This means that approximately 12% of the clients experienced a delay in receiving messages during the outbreak of the malicious code. This is very promising, given that PGBC also limited the spread of the epidemic to only 2–5% of the total number of clients. Figure 15

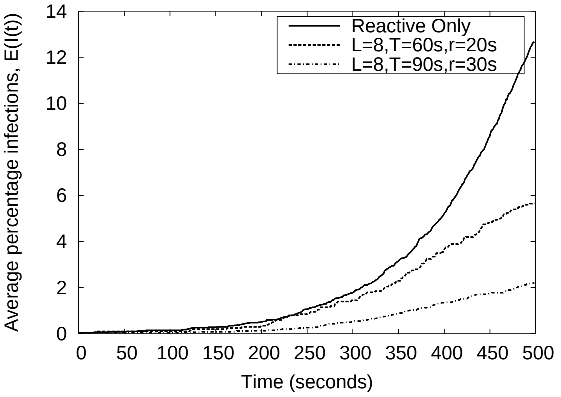


Figure 13: Comparison of reactive and PGBC defense approaches

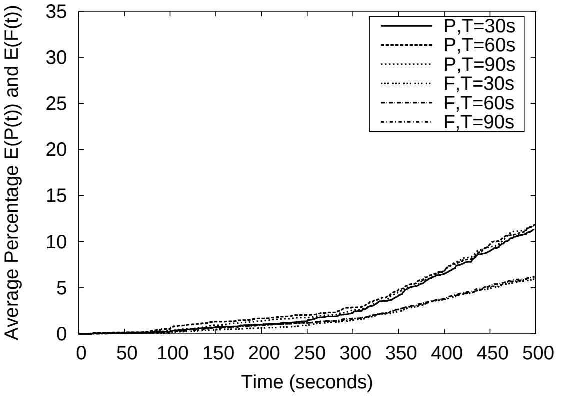


Figure 14: $E(P(t))$ and $E(F(t))$ rates for different PGBC back-off timer values ($\beta_k = \beta_{th} = 8$, $r = 30s$)

plots $E(P(t))$ for different values of PGBC alert levels. The data for various parameters do not show a correlation, confirming that the false-positive rates depend for the most part on the detection mechanism and not on the containment strategy. Overall, PGBC is found to perform very well against the SMS worm. Our results indicate that a key benefit of PGBC is that one can maintain a small throttle timer for per-client rate-limiting with an appropriate choice of PGBC parameters (T, β_k and β_{th}). Therefore, the SMS users do not experience a large delay in receiving messages during an attack. One aspect of PGBC that needs further study is how to choose among the different tunable parameters for a given messaging network.

6. RELATED WORK

The most relevant to the present study are malicious codes that target messaging networks, intrusion detection systems such as GrIDS [10], behavior-based worm detection [20] and Primary Response from Sana Security [21]. Due to space limitation, we refer to [1, 22, 2] for a description of malicious codes targeting IM and SMS/MMS networks. The above references also detail specific vulnerabilities and social engineering techniques these malicious codes typically exploit.

For Spyware and "zero-day" attacks, the conventional signature-based anti-virus tools may result in high false negatives, i.e., attacks that cannot be detected. The behavioral approaches [21, 20] are more suitable to detect these types of attacks due to correlation of behaviors among running processes and application-specific rules. Moreover, upon detection of an attack at a given client, a behavioral host-based intrusion prevention system (HIPS) may be able to determine which services were targeted. Since it is crucial to contain these attacks at the earliest, PGBC can proactively rate-limit clients in the same behavior-cluster identified by the detection system so that these clients can be checked for updated HIPS and anti-virus rules. For both known vulnerabilities and zero-day attacks, the primary role of proactive defense is to slow down the infected messages while the quarantined clients can be patched. The authors of [23] recently presented a "Community of Interest" (COI) approach for discovering host profiles within an enterprise network and then applying rate-limiting to those that show worm-like behavior.

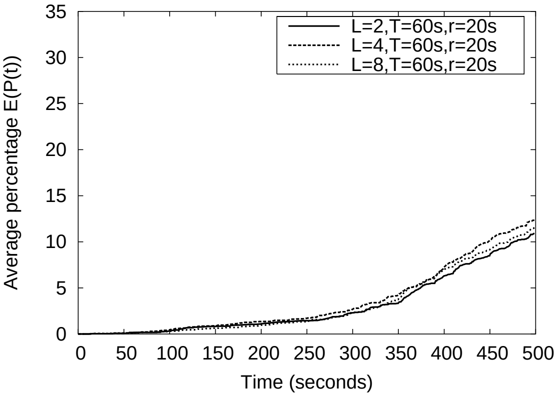


Figure 15: False-positive detection rates for different PGBC alert levels ($T = 60s$,$r = 20s$)

GrIDS [10] is a hierarchical intrusion detection system (IDS) that aggregates host and network information as activity graphs representing the causal structure of network activity. GrIDS organizes the hierarchy in terms of departments within an organization. The edges represent network traffic and attributes between the departments. Each department collects information from its child nodes and passes summary information to its parent. GrIDS uses multiple rule sets to determine how graphs are built and alerts generated. In contrast, the hierarchy in PGBC is based on messaging patterns, not the physical network infrastructure. The rule sets for generating service-behavior graphs are very simple to implement since they can be derived from server logs. Ellis *et al.* [20] presented a novel approach to automatic detection of worms using behavioral signatures. These signatures are generated from temporal and characteristic patterns of worm behaviors in network traffic, e.g., during transfer of infected payloads to other hosts, tree-like propagation and reconnaissance and changing a server into a client. Although not considered, the behavioral signatures can be the basis for quarantining individual clients or groups of clients in their abstract communications network (ACN). The Primary Response from Sana Security is another host-based behavioral approach that monitors running applications and employs multiple behavioral heuristics (e.g., writing to registry, calls to keylogging procedures, process hijacking, etc.) to identify a malicious application. It also correlates actions of multiple running applications to decide whether an application is Spyware. The current detection mechanism in PGBC monitors the length of delay queues to identify worm-like behavior. However, any of the above detection mechanisms can be implemented in PGBC, resulting in more robust and reliable detection of messaging worms.

The "Firewall Network System" described in [24] places firewalls on physical segments of an enterprise network. The firewalls specify access policies to allow only pre-defined service requests. This approach requires accurate specification of all service requests and firewall access rules for the entire enterprise network. The "dynamic quarantine" method in [25] is based on a preemptive quarantine approach that quarantines a host whenever its behavior is considered suspicious by blocking traffic on the suspicious port. The authors of [26] discuss the possibility of deploying automated responses to malicious code, e.g., by proactively mapping the local network traffic components and topology using *nmap*-like tools. Upon receipt of an alert with concrete information about the underlying vulnerability, the corresponding traffic may be blocked before it can reach other parts of the network. The basic premise of our approach is similar but based on a systematic study of messaging patterns among the clients.

7. CONCLUDING REMARKS

We have presented a novel framework called *Proactive Group Behavior Containment* (PGBC) to contain malicious software spreading in messaging networks such as IM and SMS/MMS. The exponential growth of malicious software targeting these networks in recent years requires development of proactive security approaches such as PGBC. Since all the information needed to build PGBC can be obtained from server logs, it can be easily deployed in store-and-forward networks such as SMS/MMS and server-initiated networks such as IM. The primary component of PGBC are service-behavior graphs generated from client messaging patterns and behavior clusters that partition the service-behavior graph into clusters of similar behavior. PGBC uses a combination of message rate-limiting and quarantine with increasing reaction to alerts in the network. In our evaluation results for an SMS network, PGBC is found to be several orders-of-magnitude more effective than traditional defenses such as "detect-and-block"and individual client rate-limiting. From our simulation results, it is evident that proactive defense is key to slowing down malicious codes during the early stages of its spreading. This is critical because there is only a small time window between the time an infection is detected and the time the cumulative infections reach an epidemic threshold. PGBC makes most of this time window by proactively quarantining and rate-limiting vulnerable clients in the network. There are several aspects of PGBC that are part of our ongoing study. The time scale at which a service-behavior graph should be updated for a given messaging network needs further investigation. The scalability of PGBC to very large messaging networks, perhaps with millions of clients, is yet to be investigated. Finally, the effect of false negatives (i.e., infected clients that were not detected by the detection mechanism) on overall infection rates should be studied as well.

8. REFERENCES

[1] Symantec, "Symantec internet security threat report," http://www.symantec.com/enterprise/threatreport/index.jsp, March 2006.

[2] A. Bose and K. G. Shin, "On mobile viruses exploiting messaging and bluetooth services," in *Accepted for 2nd IEEE International Conference on Security and Privacy in Communication Networks (SecureComm)*, August 2006.

[3] S. J. Stolfo, "Worm and attack early warning: piercing stealthy reconnaissance," *IEEE Security & Privacy Magazine*, vol. 02, no. 3, May 2004.

[4] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the internet in your spare time," *11th USENIX Security Symposium*, August 2002.

[5] Viruslist.com, "Instant messaging worm win32.opanki.d," http://www.viruslist.com/en/viruses/encyclopedia?virusid=84950.

[6] M. M. Williamson, "Throttling viruses: restricting propagation to defeat malicious mobile code," in *18th Annual Computer Security Applications Conference*, December 2002, pp. 61–68.

[7] 3GPP, "Ts 32.205 charging data description for the circuit switched (cs) domain," March 2003.

[8] IBM, "Ibm tivoli configuration manager," http://www-306.ibm.com/software/tivoli/products/config-mgr/.

[9] D. Montgomery, L. Johnson, and J. Gardner, *Forecasting and Time Series Analysis, 2nd Edition*, McGraw-Hill, Inc., 1990.

[10] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS – A graph-based intrusion detection system for large networks," in *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[11] N. Weaver, S. Staniford, and V. Paxson, "Very fast containment of scanning worms," in *13th USENIX Security Symposium*, August 2004.

[12] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *In Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[13] M. Williamson, A. Parry, and A. Byde, "Virus throttling for instant messaging," in *Virus Bulletin Conference*, 2004.

[14] C. Wong, S. Bielski, A. Studer, and C. Wang, "Empirical analysis of rate limiting mechanisms," in *Proc. 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2005.

[15] Cisco, "Cisco network admission control," http://www.cisco.com, 2003.

[16] Inc. Trend Micro, "Network viruswall outbreak prevention appliance," http://www.trendmicro.com, 2004.

[17] P. Porras, L. Briesemeister, K. Skinner, K.Levitt, J. Rowe, and Y. A. Ting, "A hybrid quarantine defense," in *ACM workshop on Rapid malcode (WORM)*, October 2004, pp. 73–82.

[18] A. Bose, S. Boehmer, and K. G. Shin, "Malware spreading in mobile enterprise environments," University of Michigan EECS Technical Report, June 2006.

[19] V. Samanta, "A study of mobile messaging services," *UCLA Master's Thesis*, 2005.

[20] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, "A behavioral approach to worm detection," in *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, 2004, pp. 43–53.

[21] S. Hofmeyr and M. Williamson, "Primary response technical white paper," in *Sana Security*, 2005.

[22] M. Mannan and P. C. van Oorschot, "Instant messaging worms, analysis and countermeasures," in *3rd Workshop on Rapid Malcode (WORM)*, 2005.

[23] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, and C. Kalmanek, "Enterprise security: A community of interest based approach," in *In Proc. of NDSS*, 2006.

[24] C. Zou, D. Towsley, and W. Gong, "A firewall network system for worm defense in enterprise networks," in *UMass ECE Technical Report TR-04-CSE-01*, February 2004.

[25] C. Zou, W. Gong, and D. Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," in *ACM Workshop on Rapid Malcode (WORM)*, October 2003.

[26] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "Large scale malicious code: A research agenda," in *DARPA and Silicon Defense Technical Report*, May 2003.