An Architecture for Automatic and Adaptive Defense

Ram Dantu†, *Member, IEEE,* and João W. Cangussu, *Member, IEEE*

Abstract

Network attacks have become so fast that human mitigation does not cope with security requirements. In addition, attacks are done in a smarter way mutating itself to prevent detection. Therefore, defense mechanisms must be automatic to comply with attack speed and adaptive to comply with their mutation. An architecture to support this trend in defense mechanisms is proposed here. The architecture is based upon three conceptual pillars. The first is based on the use of a multi-feedback loop control to slow down an attack. The second lies on machine learning concepts to properly distinguish between normal e attack traffic. Finally, social network provides the mechanisms to determine trust and reputation levels of network elements. A case study on the application of the proposed architecture to a worm propagation attack provides the initial evidence of the efficacy and applicability of the approach.

Keywords: automatic adaptive defense, multi-loop feedback control, social network, machine learning, worm propagation.

I. Introduction and Motivation

The Internet is prone to worm propagation, denial of service, virus and spams among other attacks. These are partly due to openness of the Internet and the lack of self defense on every node and application. For example, the spreading rate of worms has recently reached alarming velocity. That is, previous worms

Ram Dantu: Department of Computer Science, University of North Texas, rdantu@unt.edu

João W. Cangussu: Department of Computer Science, University of Texas at Dallas, cangussu@utdallas.edu

†This work was partially supported by the National Science Foundation under grants CNS- 0627754 (Detecting Spam in IP Multimedia Communication Services), CNS-0516807 (Preventing Voice Spamming), CNS-0619871(Development of a Flexible Platform for Experimental Research in Secure IP Multimedia Communication Services) and CNS-0551694 (A Testbed for Research and Development of Secure IP Multimedia Communication Services). Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

had a slow spreading rate allowing timely human reaction. However, this is not the case anymore and worms are spreading at an exponential rate. At these spreading levels, by the time a worm is detected, the majority of the hosts are already infected. Future worms will exploit vulnerabilities in software systems that are not known prior to the attack. Neither the worm nor the vulnerabilities they exploit will be known before the attack and thus we cannot prevent the spread of these viruses by software patches or antiviral signatures [1].

There is a need to control fast spreading viruses automatically since they cannot be curtailed only by human initiated control. Some of the automatic approaches like quarantining the systems and shutting down the systems reduce the performance of the network. False positives are one more area of concern [1]. Williamson describes a novel approach to this problem. This situation can be improved considerably by using benign responses, those that slow but do not stop the virus. The main idea is to delay the virus and earn time for human mediated responses [1]. Feedback control strategy is desirable in such systems because well-established techniques exist to handle and control these systems. This technique is based on the fact that an infected machine tries to make connections at a faster rate than a machine that is not infected. The idea is to implement a filter, which restricts the rate at which a computer makes connection to other machines. The delay introduced by such an approach for normal traffic is very low (0.5-1 Hz). This rate can severely restrict the spread of high-speed worm spreading at rates of at least 200 Hz [1].

In addition to containing the worm propagation, distributed denial of service (DDoS) is another attack that requires fast intervention and automated response. Since attacks can be very severe and fast, a single network element may not be able to prevent the attack. Some of the requirements for an automated response are described next:

- Multiple levels of defense: in order to contain the attack to small number of nodes, we need sufficient communication between the different network elements. Infected elements should share the attack-specific information with their neighboring networking as well as with the perimeter controller.
- Cooperation among multiple elements: a new protocol need to be defined for communicating with the neighboring devices. In addition, a new message format needs to be defined. IETF has defined an intrusion exchange format but this does not cover all the threat-specific information.
- Solution that can handle multiple threats: from our observations, several kinds of worms, and DDoS attacks have similar symptoms. Typically these are memory depletion, CPU exhaustion, and high

volume of connections. It is desirable for any prevention/detection technique to block multiple threats.

- Automated patching for vulnerabilities: currently patching (after vulnerabilities are announced) is done manually. Hence it takes several hours and probably days. So it is practically impossible to contain an attack within few minutes. On the other hand, if we can automatically update signatures [2], [3], [4], firewall rules, configuration parameters, and patches, majority of the nodes can be saved from intrusions.

Here we describe a novel architecture for network level security based on feedback control theory. We described this architecture in three levels. At network level, we propose a multi-level feedback controller. Next, at the nodal level, we use the proposed methods for interworking of control system theory, machine learning and social-technical procedures. Tuning the input parameter in a controller is a challenging task. We have used the machine learning techniques for automatically setting the parameters for minimum number of false alarms. Next task is how to trust the neighbor for sharing the security related information. We have based these decisions on the trust and reputation of the sessions being generated. The heart of the controller is based on a two-queue system. These are safe and suspect queue. The suspected connections (marked by the machine intelligence) are delayed for processing and thus slowing down the attack. In the following sections we describe the details of the architecture, and controller.

II. Proposed Security Architecture

A typical communication system is based on final or leaf nodes and internal nodes. Host computers in a network and phones in a telephone system are examples of leaf nodes. Internal nodes could be routers and firewalls in a network and switches and stations in telephone systems. Based on this general structure a defense architecture construct upon feedback control concepts is proposed. The architecture relies on the deployment of controllers at all levels and the sharing of information among these controllers. Also, the defense mechanism can be applied to different types of services such as VoIP (Voice over IP), multimedia, video, and data communication.

A secured network consists of firewalls, sensors, analyzers, honey pots, and various scanners and probes. These components are either separate elements or collocated with hosts, servers, routers and gateways. In this architecture, a (centralized or distributed) controller is responsible for collection and monitoring of all the events in the network. It is a distributed controller and it is knowledgeable about the network

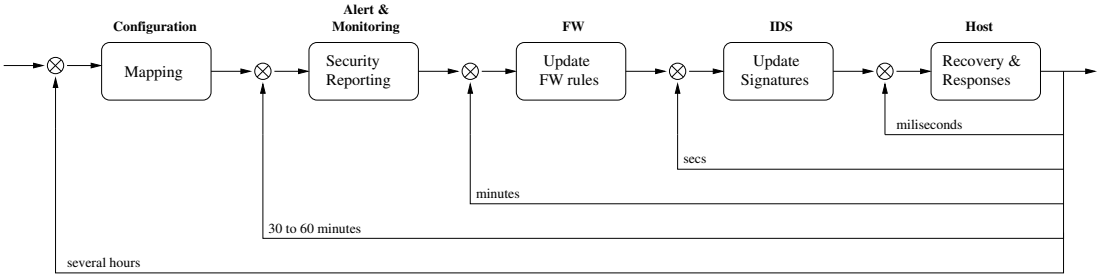

Fig. 1. Controller architecture for end-to-end security engineering. This diagram describes a logical view of a multi-loop feedback controller.The controlling functions are distributed and spread across several network elements. Detection of abnormal behavior in a host can take several hundred milliseconds where as it may take several hours for a human operator to react and reconfigure the network. Similarly a firewall at the perimeter can take several minutes for automatic detection of suspected hosts and adaptation of firewall rules.

topology, firewall configurations, security policies, intrusion detections and individual events in the network elements. This controller is a logical function and can be deployed anywhere in the network. As shown in Figure 1, the controller communicates with other controllers located in different network elements. These network elements are responsible for detection and collection of the events in the node and communicate to the controller. Subsequently, controller will run through the algorithms, rules, policies and mathematical formulas (transfer functions) for next course of action. These actions are communicated to the clients. As described in Figure 1, the architecture evolves from a concept of closed multi-loop control. Changes regarding the security behavior are captured and mixed with the incoming network signals. This piece of information is used to formulate the next course of action. The final result is outcome from multiple loops and integration of multiple actions. The response times within each loop (we call them defender-loops) varies from few milliseconds to several tens of minutes. For example, nodal events like buffer overflows, performance degradation can be detected in matter of milliseconds. On the other hand, it may take several seconds to detect failed logins, changes to system privileges and improper file access.

The proposed architecture is based first on slowing down an attack so that a pattern can be identified. Different types of attacks have different side effects at different levels and sharing information helps on identifying malicious behavior. In addition, attacks occurs with such speed that human detection is almost impossible. Techniques such as machine learning can be used to identify pattern behaviors and detect the attack. However, by the time a proper identification is made, almost all nodes have already been affected by the attack. The use of control theory concepts and machine learning techniques can be combined to provide an automatic and adaptive efficient solution. The use of a controller cannot contain an attack by

itself but it has been shown that it can slow it down considerably as seen in Section II-C. Figure 2 describes an architecture for automatic defense, building blocks, and their interaction in securing the network. The machine learning system (MLS) learns the legitimacy of the incoming connection based on the number of input and output connections, number of infected hosts, connections dropped, etc. [5] In addition this system receives information about the trust and reputation of the incoming connections. This allows machine learning techniques enough information to properly identify the attack and take measurements (update the signature of filters in the system [2], [3], [4], update set points and other parameters of the controller, etc.). Moreover proper identification decreases the chances of false positives and/or negatives. However, such solution is not enough in the sense that communicating elements may be infected and the information shared (used by the machine learning techniques) may not be reliable. Social networks can then be used to determine trust and reputation levels for the elements. Based on these levels, the type and amount of information to be shared can be specified as well as the use of information received by other elements. Based on this information, MLS classifies the information as legitimate and new connections. In particular the MLS considers the past experience (history) with source and calculates the probability of incoming connection as legitimate or unknown. Social networks system (SNS) receives this classification and computes the trust and reputation of incoming sessions based on the social closeness of the source. For example, the social closeness depends if the elements are considered friends, coworkers or family.

Initial experiments have shown that the proposed architecture is general enough to be applied to any topology, though the topology may affect the communication overhead. The controllers to be deployed are software based and therefore are not affected by the topology. In addition the proposed architecture can be used to defend against different threats such as distributed denial of service (DDoS) and worm propagation. The controllers in this case can use different signatures to identify different threats at different levels. For example, in the case of a spreading worm, the change in velocity in the number of connection requests can be used to identify a possible infection at host level. At this point, the controller at the host is activated. The host then communicate this to the firewall which can use a different signature, such as change in the acceleration in the number of requests, to activate its own controller. A DDoS attack can use different signatures (such as CPU exhaustion and memory exhaustion) and different controllers to contain the attack. Therefore, the proposed architecture is general with respect to services, topologies, threats, and signatures.

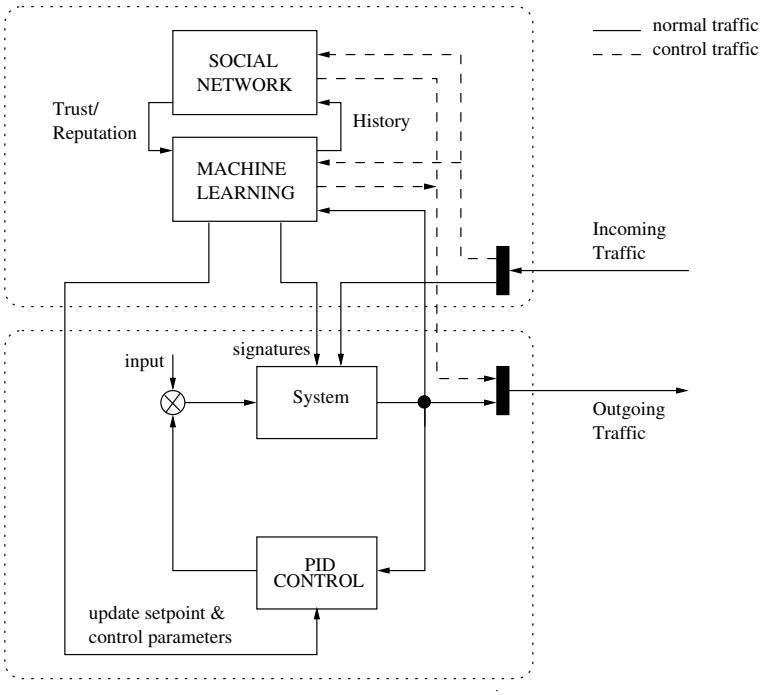

Fig. 2. General control structure to be used in the end-to-end security engineering architecture in Figure 1. For adapting to the dynamic traffic, automatic signatures are created and fed to the automatic controller. In addition, machine intelligence can be adopt based on the events happening in other networks elements. The communication between the network elements can be carried through the exchange of control messages using a low-overhead protocol.

A. Three Dimensional Defense

As stated in Section II, the proposed architecture is constructed upon concepts from very distinct areas: control theory, machine learning, and social behavior. Below we describe the role of each of these disciplines.

Control Theory: There has been over 40 years of research in control theory and its application ranges from obvious areas such as electrical and mechanical engineering to less common areas such as biology and software engineering. Recently, researchers have also start using control theory for quality of service and for network security purposes [6], [7], [8]. In order to provide automatic defense, a mechanism to regulate properties of the network must be in place. The mechanism must be able to drive the property of interest to a desired level by regulating network related inputs (number of packets per time unit, time out, etc.). Also, in some situations a steady state must be achieved. Finally, such mechanism must be easy to deploy and should not demand large computational resources. Feedback control theory provides such mechanism. For example, a PID (Proportional, Integral, and Derivative) controller can be used to

regulate suspicious traffic and slow down a spreading worm. A PID controller can be easily implemented and deployed with negligible resource usage.

Machine Learning: Algorithms based on machine learning techniques can be mainly used for detection purposes [5]. Under our perspective, detection has three dimensions. The first regards the process of identifying if/when an element of the network is under attack. The second dimension deals with the problem to distinguish between normal and malicious traffic while the third dimension tries to identify the source of the attack (for example, an executable inside an element or an external element). A large body of work already exists on these areas [5]. For example, a classification method based on learning decision trees and rules has been used to reduce false alarms, achieving accuracies ranging from 91% to 98% [9]. Their classifier algorithm complements the training examples with domain-knowledge rules provided by some expert or by an early iteration of the algorithm. Lane proposes a semi-supervised model that has achieved results ranging from 93% to 97% [10]. His approache is based on the relaxation of key assumptions for intrusion detection and the use of partially observalbe Mrakov decision process. Another approach uses protocol behavior for detection reaching between 82% and 100% accuracy depending on the windows size (amount of data) used [11]. Since many attacks are related to the improper use of a protocol, the classification is done when the protocol use deviates from the intended use. Moore and Zuev [12] make use of supervised machine learning to classift network traffic. They use a Naive Bayesian approached improved by the use of a fat correlation-based filter. Their results using the improved approach range from 84% to 96% accuracy in the classification. Alternatively, machine learning techniques have been used to classify incoming calls on voice over IP. The approach proposed by Kolan and Dantu [13] is based on a feedback structure using bayesian learning to compute trust and reputation of an incomming call and ultimately determining if the call is a spam or not. They have achieved results of 97.6% accuray with 0.4% of false postives and 2% of false negatives. A list of these classification schemes as well as their accuracy is presented in Table I.

Social Behavior: Learning the behavior of the participating entities would let us make many intelligent decisions regarding how and what information to share. The behavior of the participating entities can be learned during the course of a period of time. The behavior can be estimated by their past history. This process of observing an element's behavior over a period of time constitutes the "trust" of the element. This available trust information helps in classifying the participating entities in cluster of distinct levels

Learning Technique	Accuracy
Learning decision Trees [9]	91% to 98%
Semi-supervised model [10]	93% to 97%
Protocol behavior [11]	82% to 100%
Supervised Mahine-learning [12]	84% to 96%
Trust and Reputation Bayesian Analysis [13]	97.6%

TABLE I

MACHINE LEARNING TECHNIQUES APPLIED TO TRAFFIC CLASSIFICATION AND THE RESPECTIVE ACCURACY OF THE CLASSIFICATION SCHEME

of trustiness. Human feedback (implicit or explicit) can be part of it [14].

Social networks can be used to represent elements relationships that can be derived along the paths of the network [14]. These relationships are transitive and transparent. If domain A trusts domain B and domain B trusts domain C, then with a high degree of confidence, an argument can be made that Domain A can trust domain C. These social networks can be used to infer the associated relations between the elements of the network. The social network represents the associated and trusted neighbors from which the user is willing to receive/share information. A graph can be generated based on neighboring elements and can be used to derive the reputation of an element. Reputation is derived from trusted peers while the trust is calculated based on the past history. The peer proxies would derive reputation by their trusted peer proxies, and this would continue until the last proxy in the via list (for example, relays in the email header) or the proxy that is reachable from source by one hop is reached. Based on the reputation inference from the peer proxies of the source and the entities in-between, the reputation can be inferred.

The algorithm's accuracy and false alarms are described in [14] through an example on how to detect unwanted emails. Dantu and Kolan [14] devised a technique for classifying the emails as "spam", "phishing," and legitimate. They classify the incoming emails by performing the following analysis on the header; DNS-header analysis, SMTP path-and-relay analysis, social-network analysis, time series analysis, and principal-component analysis [15]. Finally, based on the results from these analyzes, they successfully isolated unsolicited and phishing emails from legitimate emails. Furthermore, they categorized spammers and phishers into serial spammers/phishers, recent spammers/phishers, prospective spammers/phishers, and suspects. Next, they classified legitimate emails and trusted domains into socially close (family

members, friends), socially distinct (strangers, etc), and Opt-outs (tele-marketing). Finally, they analyzed the technique's performance for false positives and false negatives using ROC curves. Overall, our classifier resulted in a precision of 98.65% which is well comparable to the precisions achieved by current filters as can be seen in Table II. The details of the intellegent classifier are described in [15].

Analyses	False Positives	False Negatives	Precision Phishing Hits
DNS header analysis	260	00	85%
DNS header Analysis + social network analysis	13	05	92.63%
DNS header Analysis + social network Analysis + time series analysis	06	01	98.4%

TABLE II

SUMMARY OF RESULTS (DNS-HEADER ANALYSIS, SOCIAL NETWORK ANALYSIS AND TIME SERIES ANALYSIS) ANALYSES PERFORMED ON AN E-MAIL CORPUS CONSISTING OF 13,843 EMAILS COLLECTED OVER 2.5 YEARS.

B. Experimental setup

In order to evaluate the potential of the proposed approach we have used a virtual network testing environmnet, named Goliath [16], to conduct the experiments. Goliath allows for the creation of multiple "virtual" nodes in a single machine and any overlay netwrok topology can be chosen. Goliath has been built on top of a real-life netwrok and, therefore, incorporates all TCP properties. For the experiments conducted here a total of 42 machines have been used with 12 virtual nodes in almost all machines totalling 500 virtual nodes. Also, since Goliath is a platform free environment, a mixture of Windows and Linux machines have been used. The results for this initial experiment are presented next in Section II-C.

C. Preliminary Results

Preliminary results of applying a PID controller to slow down a spreading worm are presented next. The results are a clear indication that the solo use of a controller can slow down the worm but cannot contain it. The integration with other techniques form a more complete solution for the problem as seen in Figure 3. The controller is based on a queue system of connection requests as described next.

It is assumed that, as the number of requests increases, a portion of them will be sent to a delay queue to be served later. The remaining ones are sent to a safe queue to be served immediately. The overall

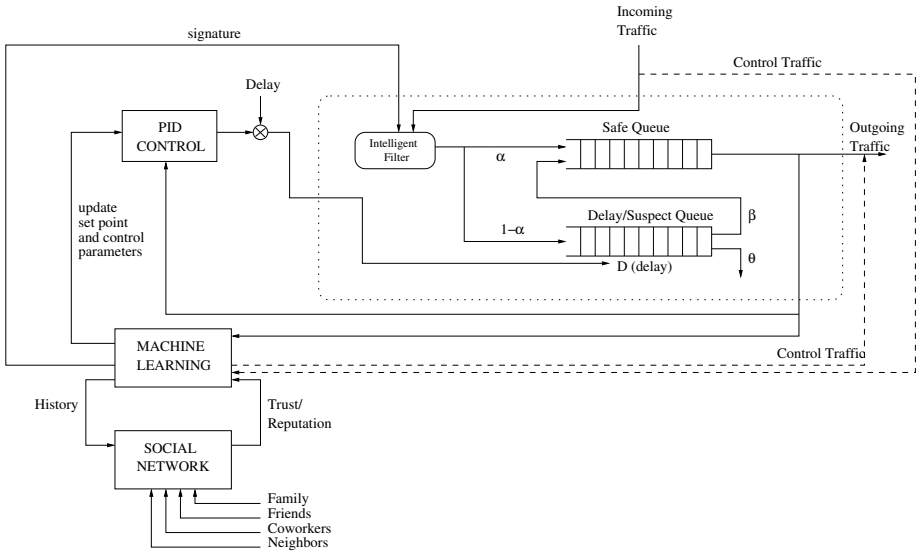

Fig. 3. Queue system used in conjunction with the PID controller, machine learning, and social networks. This is a detailed view of controlling propagation of a worm. In this example, suspected traffic is delayed and merged with the normal traffic at a later time. This delay will either drop or delay new connections. The amount of delay is automatically adjusted based on the intensity of the detected attack. Traffic is classified as safe or suspected based on the past experience/transactions with the source. This classification is carried out by the use machine learning algorithms.

structure of this queue system is depicted in Figure 3. Parameters related to the size of the delay queue and the number of dropped (time-out) connections are used to control the total number of connections resulting in a slow down of a spreading worm [7]. Sapphire worm spreading is taken as an example to show the applicability of our approach. The number of requests generated by the worm increases according to an s-shape format [17]. The goal of the example is to slow down the spreading velocity of a worm by controlling the total number of connections detected by a firewall.

It is assumed here that all requests in the safe queue, at one point in time, are served. Therefore, at each time a server is allocated to handle the requests at the safe queue, it serves all the new incoming requests plus all the ones transferred from the delayed queue at the previous time period. Hereafter we refer to the proposed approach by `ADS` (Automated Defense System). The input for the `ADS` is the number of requests. In a normal traffic condition this number tends to vary over a constant value. However, under an attack condition the number of requests will assume a different behavior. At the host level a step input can be used to characterize the attack. Once the host is infected, the worm will try to make a certain number of connections per time unit in addition to the normal requests. The step function however cannot characterize an attack at the firewall level. For a large network, an S-shaped function is a better alternative

to represent such behavior.

A protocol is required to define the overall working of the system in an organized manner. In case of a worm attack it is necessary that the protocol achieves the desired function, and that the defense mechanism is able to contain the spread of the worm. The assumption here is that there is an infected host in the network. The firewall interacts with both infected and uninfected hosts. Normally, a mixture of normal and attack traffic is sent by the infected host. When an uninfected host receives an infected packet and an attack is detected, it sends an infected signal to the firewall and starts infecting other hosts. This occurs when any host on the network is infected in this manner. At each point, the firewall checks for a threshold number of infected hosts. Once the threshold for infected hosts is reached, the firewall sends a kill process or reduce setpoint signal to all the infected hosts and a reduce PID setpoint signal to all the uninfected hosts. The firewall also checks for the containment threshold, which is reached when there are no infection messages in the network for a certain period of time. Once the containment threshold is reached the firewall again sends a reset signal to all uninfected hosts. This signal resets the setpoint to initial values.

The objective of our approach is to delay the worm propagation and restrict the spreading to a small number of hosts. The performance of ADS depends on a number of variable parameters and the outcomes may vary based on the values of these parameters. For example, the outcome may vary depending on the portion of connection requests forwarded to suspect queue compared to the safe queue. In this context, we assumed two modes of queuing; one intelligently (based on machine learning techniques) and another one randomly. Similarly, the firewall has the option of issuing different commands to the hosts for stopping the spread of worm. For example, it can ask the hosts to kill a suspected process or drop all the requests in the suspect queue (again, machine learning can be used to identify malicious executable code [5]). We divided the experiments into two parts: i) control using inner loop (host only - Figure 4(a)); and ii) control using the inner and outer feedback loops (host and firewall - Figure 4(b)). As can be seen in Figure 4(a) the deployment of the controller at the host level can slow down the attack but cannot contain it. Also, the higher the efficiency of the machine learning algorithm used in the identification of normal and malicious packets, the longer it takes to infect all hosts. Figure 4(b) shows the results of applying the controller both at host and firewall level. Again, the solo use of the controller does not contain the attack but when combined with machine learning algorithms to identify malicious executable code, ADS is able to contain

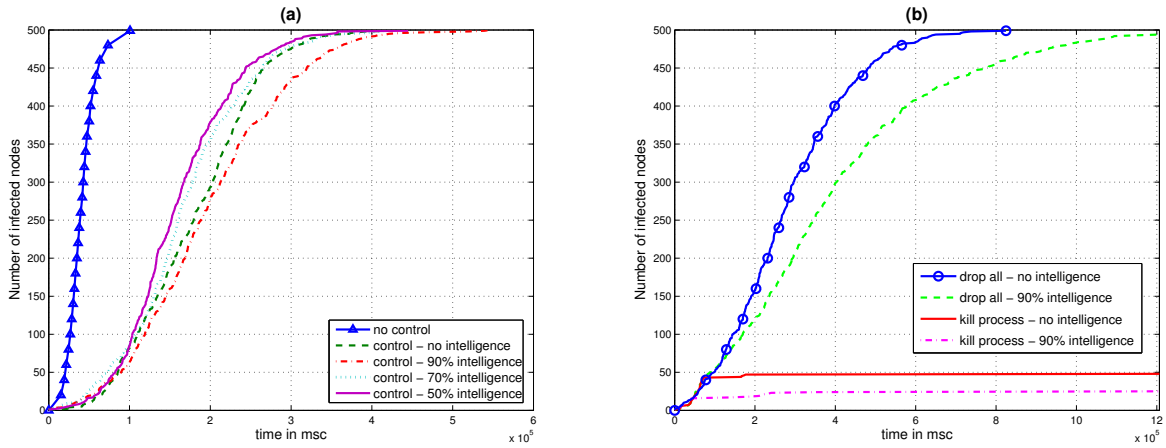

Fig. 4. Results from the use of a PID controller at: (a) host level only and (b) host and firewall level.

the attack to less than 5% of infected nodes.

The results in this section show that ADS is very effective against fast scanning worms. The results from the experiments show that by using just the inner loop (controller deployed only at the host level), we can delay the spread of worm by more than five minutes. When we use the outer loop (controller deployed at firewall) in conjunction with the first one, we notice a five-fold increase in delay in the spread of the worm. The outer loop can operate in two modes: the drop-mode (drop all connections in the delayed queue) and the kill-mode (kill all processes generating suspect traffic). In drop-mode connections to be sent to the delayed queue can be identified without intelligence (selected randomly) or with intelligence (using a machine learning algorithm to identify attack patterns). We achieve an eight-fold increase in the delay when we use the outer loop in the drop-mode and a twelve-fold increase in the delay when we add intelligence in the drop-mode. We achieved better results when we used the outer loop in the kill mode, where we were able to limit the spread of the worm to a few machines in the network. In this mode, similar to the drop-mode, we achieved better results by containing the worm to less than 5% of the machines in the network, when we used intelligence to distinguish between attack traffic and normal traffic. Even without the use of intelligence to classify the traffic, with the use of the outer loop, we were able to contain the worm to 10% of the machines in the network. Use of intelligence to classify traffic, in general, increases normal traffic throughput, while reducing the number of false positives. Thus we see that ADS is able to effectively contain as well as prevent the spread of the worm as described in [7].

It is clear from the experiments above that a complete solution does not lie on the use of a single technique and on its application at a single level. Feedback control can slow down a worm but it cannot

contain the worm. In addition, the controller is more effective when an intelligent algorithm is used to identify malicious packets. In the experiments above, the identification has been done randomly to show its effect. Actual techniques can be based on machine learning concepts or bayesian approaches [5]. Finally, containment cannot be achieved by a single element of the network and cooperation and information sharing are crucial for an automatic defense system. Therefore, trust and reputation of the elements need to be taken into consideration when receiving and sending information and Social Networks can play an important role in defining the level of cooperation.

III. Related Work

Recently automated defense against attacks has been a major topic of discussion in the network security community. In a recent paper, Cisco [18] clearly states the need for self-defending network. It also describes some of the properties of such an adaptive system. The properties are: (i) Remain active all the time; (ii) Perform unobtrusively; (iii) Minimize propagation of attacks; and (iv) Quickly respond to as-yet unknown attacks. Our conjecture is that ADS has all the above properties, thus it is a suitable self defending mechanism. Next we compare ADS with some of the contemporary approaches. To the best our knowledge, only the work of Porras, Levitt, et. all [8] takes a comparable approach for automated defense of worms. However ADS outperforms their approach by containing the attacks to a smaller number of nodes in a shorter period of time.

A_1 - *An OS independent Heuristics-based Worm-containment System. [19]*: This work presents results about controlling the worm using heuristics at the host level. Their system continuously observes outgoing network traffic over a finite-duration traffic window, and using heuristic-rules executing in a secondary environment, detects infections. Their system automatically quarantines the machine to stop further propagation of the worm. Similarly, ADS also has a host level detector, but adds a network level feedback and the decision to quarantine is based on both host and network level, thus resulting in less false positives. Hence using both the host and the network level makes containment more effective. Also, we have a feedback control mechanism which does not use heuristics but uses the current state to provide feedback to the controller.

A_2 - *Fast Containment of Scanning Worms [20]*: This work uses a method where they limit a worm's spread by isolating it to a small subsection of the network. They achieve this by turning the scan suppressor

around and then use their own scan detector algorithm which does not let the worm escape once the worm comes into the internal network. Their work is suitable for deployment in low cost network hardware. We concentrate on stopping of the worm itself and do not care about the way its scanning. The method described in [20] works only at the network level. ADS works at both the network and the host levels.

A_3 - *Throttling Viruses - Restricting propagation to defeat malicious mobile code*: In [21], Matthew et al. show that if a worm is spreading through a network, the connection rate within the network goes up. They also try to control the increase of rate of spreading to hosts. In this work, they try to limit the number of connections made to a new host. They first filter the connections going to new hosts and then limit them using a delay queue, similar to our system. In this work we use the same fact that an uninfected machine will behave differently than an infected machine when worm is spreading through a network. The difference in our work is that we use intelligent queuing (based on trust and reputation described in [14]) and our filter is based on control systems feedback loop which continuously monitors the outgoing connection rate. If the connection rate goes up, it brings the rate down by dropping requests regardless of whether it is going to new or old host.

A_4 - *Feedback Control Applied to Survivability*: In this work Kreidl et al. [22] use a host-based feedback model to keep the host running even when there is a worm in the network. Their architecture consists of a set of sensors, actuators and a controller. The purpose of the sensors is to detect any anomaly. The actuators' purpose is to keep the anomaly under check. The sensors and the actuators are controlled by the controller. When some anomaly, such as worm attack happens, the sensors give feedback to the controller, which then activates the actuator, based on the feedback. Thus, their method is a type of feedback control defense. Our work also uses the feedback model, but ADS is based on control system theory. The sensor is based on the connection rate; they use a commercial IDS system for sensing. Both ADS and their model kill processes in an effort to stop the spread of the worm. ADS does it considering the current state of the network, while they consider the state of the host.

A_5 - *Modeling a Computer Worm Defense System [23]*: This work uses two steps to propose a defense against worms. First there is a friend model where hosts exist in the form of a set of friends. When one of the hosts gets infected, it finds out the PT (Perceived Threat) from that attack and tells its friends, alerting them to look for a similar attack. The second step is a tree model architecture. Here the authors represent the network elements in a tree model where all internal nodes are firewalls and all the leaves are

vulnerable nodes. It is assumed that each node has an equal number of children. The tree has a threshold defined for each level. Once the threshold for that node is reached, i.e. the number of children is greater than the threshold value; the node first enables its firewall rules and then informs its parent node about the presence of some anomaly. Thus the information about the anomaly propagates up the tree, resulting in multilevel security. In ADS firewall (or any perimeter controller) controls the action of each host by sending appropriate signals to it. The hosts do not share anything between them which considerably reduces the overhead. We use a feedback control at host and firewall to control the rate and we are not concerned with the prediction of threats but their identification.

Features↓ Approaches →	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	ADS
Cooperative					√		√		√
Centralized Control	√	√	√	√	√	√	√	√	
Distributed Control									√
Host based only	√			√				√	
Network based only		√	√				√		
Host and Network based					√				√
Single loop control	√	√	√	√	√	√	√	√	
Multi loop control									√
Single variable control	√	√	√	√	√	√	√	√	
Multi variable control									√

TABLE III

COMPARISON BETWEEN ADS AND 8 DIFFERENT APPROACHES. FEATURES IN BOLD REPRESENT DESIRED ONES.

A_6 - *Mitigating Distributed denial of Service Attacks using a Proportional-Integral-Derivative Controller [24]*: This work presents some similarities with the ADS approach as they also use a PID controller to control the incoming connection rate. They create a model which stops any denial of service attempt on the network. Their model is not a host-based model, rather their model sits on outer routers of a protected network. The difference is that we observe the outgoing rate for each host, and we are not concerned with the incoming rate. ADS is based on a well formulated state space model discussed in Section II-C and also has an outer loop controller at the firewall. The differences between their approach and ADS are listed in Table III.

A_7 - *A Hybrid Quarantine Defense [8]*: This work suggests two approaches, rate limiting and a

collaborative action. Porras and Levitt [8] claim that rate limiting will delay the worm attack and that the collaborative approach will not let it reach the saturation level in a network. They suggest that it is best to use a hybrid of these methods. They use MATLAB to validate their methodology. The work done here takes the same two approaches. We have an inner loop and outer loop (representing collaboration) based on control system theory. Also, we verify this by taking a more practical approach, using the testbed developed instead of a simulated MATLAB environment. Our approach contains the infection rate to 0.9% of hosts per minute compared to 5% with their approach. While their approach appears to be based on single loop single variable control ADS use a mutli-loop multi variable control.

A_8 - *Adaptive Defense Against Various Network Attacks [25]*: Similarly to what is proposed here, Zou's work addresses general solutions for various network attacks. They propose a defense mechanism which adapts according to the severity of the attack computed using metrics associated with different attacks. The adaptation is done by minimizing the cost of false positives and false negatives. That is, the cost of accepting attack traffic or dropping normal traffic. Their approach appears to properly identify and predict the severity of the attack. Consequently slowing down/decreasing the effects of the attack is achieved. It is clear that Zou's adaptive approach improves upon fixed-parameter filtering. The approach proposed here is more comprehensive than their solution. As seen in Section II-C, ADS can not only identify and slow down the attacks but also contain them. This is achieved through sharing of information among network elements and also through the use of control theory and machine learning techniques. ADS is a cooperative, distributed, network and host based approach with multi-loop and multi-variable solutions whereas Zou's approach is a centralized non-cooperative host based approach with single-loop solutions.

In general, host-based or network-based defense alone can be both ineffective and inefficient. A combination of both host-level and network-level defenses with intelligent quarantining is a good approach for containing an attack propagation. Also, a centralized control appears to limit the effectiveness of automatic defenses and increase the overhead; a distributed control appears to be a better solution. A cooperative approach is therefore also desired since elements of the distributed control will need to share information. Another important aspect in improving the state of the art in automatic defenses is the use of multi-loop as well as multi-variable control. While this increases the complexity of the controllers it also increases their effectiveness. Table III presents a comparison among the techniques described above and ADS. As can be observed, ADS has all desired components to attain a more generic and efficient solution

in order to achieve automatic defense.

IV. Defending Against Different Attacks

The architecture described in Figure 1 is very general purpose and can be used to defend from several different kinds of attacks. The three dimensional approach described in the Section II-A can be used in prevention and detection of several attacks such as DDoS, worm propagation, and spamming. In this section, we describe the common features in these attacks and how the features of the architecture can provide a comprehensive solution.

DDoS: One of the characteristics of the DDoS attacks is the large volume of bursty traffic to a process or a node. Signatures that can detect the first and second order derivatives of the incoming traffic (e.g., computing velocity and acceleration vectors) can be used to detect such sudden bursts. One issue with this kind of signatures is that it is difficult to differentiate between the bursty and legitimate traffic. Next, this kind of signatures can be detected at the host as well as at the server. In addition, we can see a definite correlation of traffic from the outgoing traffic at certain machines. Hence it is important that several machines to cooperate in detecting the abnormal behavior in a network. For throttling the DDoS attacks, a fast response is required and hence we used a feedback control system. In fact this system can be used for controlling incoming as well as outgoing traffic. In particular, intelligent queuing and control is used to slow down the attacks. Social network analysis can be used for differentiating the friends from strangers.

Worm Spreading: Unknown worms may be difficult to detect by pre-configured signatures. However, it may be possible to detect an anomalous behavior such as creation of large number of unexpected processes and sudden variation of CPU and memory saturation. In some occasions, this can be detected as sudden burst of outgoing connection requests. In addition, these connections can be originated at the host, server, and a perimeter of the enterprise network. It is also possible to observe a correlated behavior of infected hosts and servers. This kind of unexpected behavior can be found at host, server, and the perimeter. Also, machine intelligence can be effective in the automatic generation of the signatures. After receiving the automated signatures, a feedback control system can be effective in slowing down the spreading of the worm. Sometimes, it is possible that machine intelligent fails to detect abnormalities. For example, when we receive attachments to emails, it is important to undertake a social network analysis to differentiate a

worm from legitimate execution of a binary.

Spam Detection: From our observation, when it comes to receiving or rejecting emails people use the social meaning of trust, reputation, and friendship of the sender. Based on the social interactions of an email user, his incoming email traffic can be divided into different categories, such as Phishing emails, Telemarketing emails, Opt-ins emails etc. Opt-in emails are the ones we voluntarily opt to receive the emails from a sender for some information but he might send some unwanted solicitations in addition to wanted information. Also, quite often, spammers send the unsolicited emails to large number of users. Hence, suspected email correlation between different recipients can be a good indicator for filtering the spam. Finally, an explicit as well as implicit feedback from the user can be used for detecting the spam. An explicit feedback is where the recipients label the incoming email as spam. An example of implicit feedback is where the recipient continuously deletes the email without reading the email. An integrated solution containing the user feedback, trust and reputation of the sender can be used for detecting the spam. We used a multi-stage adaptive spam filter based on presence, trust, and reputation for detecting spam in emails calls. The results of using feedback control and social network analysis is described elsewhere [14], [7].

Hence we observe several common functions required for automatically defending attacks. Since attacks are unknown and unpredictable, it is important the architecture adapts to the attack using adaptive feedback.

V. Conclusions and Future Work

Applications running on the Internet are prone to various kinds of attacks. Examples of these attacks are: i) DDoS, ii) Worm propagation, and iii) Spamming. These attacks are possible partly because of openness of the Internet and lack of self defense in the network elements. When a node or network is under attack, it is required to guarantee the graceful degradation of the applications instead of sudden death. To accomplish this task, we used an architecture consisting of automatic and adaptive defense strategies. The approach follows the trend on automatic adaptive defense mechanisms that are required to cope with fast and aggressive attacks. To supplement these strategies, we used intelligent algorithms to detect the behavior of the attacker based on the sender's traffic pattern and receiver's interest in receiving the traffic. For example, if there are too many emails/voice-calls from a known receiver within a short period of time, receiver exhibits uninterest in the incoming sessions. Some other symptoms of attacks are:

i) sessions consume large CPU/memory bandwidth and ii) originate large number of sessions. In order to effectively respond to these attacks, we used distributed controllers and a protocol for communication between them. These controllers can be located either in the host, server or a firewall. As part of the controller, we used multiple feedback loops and machine intelligence algorithms for learning the sender and receiver behavior. This kind of learning and control are a closed loop process. The system is normally at the stable state but becomes unstable based on the degree of attacks. Sometimes it is difficult to detect the abnormality based on the machine intelligence only. Hence we considered the social network of the senders and receivers. Overall we observed that cooperation, feedback, self-defense and self healing are the important characteristics of any effective strategy.

REFERENCES

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

[2] Z. Liang and R. Sekar, "Fast and automated generation of attack signatures: a basis for building self-protecting servers," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, (Alexandria, VA, USA), pp. 213–222, ACM-SIGSAC, 2005.

[3] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatic signature generation for polymorphic worms," in *IEEE Security and Privacy Symposium*, May 2005.

[4] K. Wang, G. Cretu, and S. Stolfo., "Anomalous payload-based worm detection and signature generation," in *Proceedings of the Eighth International Symposium on Recent Advances in Intrusion Detection*, September 2005.

[5] M. A. Maloof, *Machine Learning and Data Mining for Computer Security*. Springer Verlag, 2006.

[6] S. Cheetancheri, D. Ma, A. Ting, J. Rowe, T. Heberlien, and K. Levitt, "A framework for worm defense evaluation." NSF Cyber Trust Annual Principal Investigator Meeting, Sept. 25 -27th 2005. Newport Beach California.

[7] R. Dantu, J. W. Cangussu, and A. Yelimeli, "Dynamic control of worm propagation," in *Proceedings. ITCC 2004. International Conference on Information Technology*, vol. 1, pp. 419–423, April 5-7 2004.

[8] P. Porras, L. Briesemeister, K. Skinner, K. Levitt, J. Rowe, and A. Ting, "A hybrid quarentine defense," in *2nd Workshop on Rapid Malcode (WORM)*, ACM SIGSAC, October 2004.

[9] E. E. Bloedorn, L. M. Talbot, and D. D. DeBarr, *Machine Learning and Data Mining for Computer Security*, ch. Data Mining Applied to Intrusion Detection: MITRE Experiences. Springer Verlag, 2006.

[10] T. Lane, *Machine Learning and Data Mining for Computer Security*, ch. A Decision-Theoretic, Semi-Supervisied Model for Intrusion Detection. Springer Verlag, 2006.

[11] J. P. Early and C. E. Brodley, *Machine Learning and Data Mining for Computer Security*, ch. Behavioral Features for Network Anomaly Detection. Springer Verlag, 2006.

[12] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proocedings of ACM SIGMTERIC*, (Alberta, Canada), ACM, June 6–10 2005.

[13] P. KOLAN and R. Dantu, "Socio-technical defense against voice spamming," *ACM Transactions on Autonomous and Adaptive Systems*, March 2007. (Accepted and tentative publication date).

[14] R. Dantu and P. Kolan, "Detecting spam in voip networks," in *USENIX - Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, (Cambridge MA), July 2005.

[15] P. Srikanth, "A multi-variate analyses of smtp paths and relays to restrict spam and phishing attacks in emails," Master's thesis, University of North Texas, 2006.

[16] W. Fagen, J. W. Cangussu, and R. Dantu, "Goliath: A configurable approach for network testing," in *3rd IEEE International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, (Orlando, Florida), IEEE, May 21–23 2007. (Under submission).

[17] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire worm." http://cs.berkely.edu/ nweaver/sapphire.

[18] "Core elements of the cisco self-defending netowrk startegy." www.cisco.com/application/pdf/en/us/guest/netsol/ ns413/c654/ cdc-cont_0900aecd80247914.pdf.

[19] U. Savagaonkar, R. Sahita, G. Nagabhushan, P. Rajagopal, and D. Durham, "An os independent heuristics-based worm-containment system." Intel Corporation, 2005.

[20] N. W. Vern Paxson and S. Staniford, "Very fast containment of scanning worms," in *13th USENIX Security Symposium*, August 9-13 2004.

[21] M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," in *18th Annual Computer Security Applications Conference*, (Las Vegas, Nevada), December 9-13 2002.

[22] O. P. Kreidl and T. M. Frazier, "Feedback control applied to survivability: A host-based autonomic defense system," *IEEE transactions on Reliability*, vol. 53, March 2004.

[23] S. G. Cheetancheri and K. N. Levitt, "Modelling a computer worm defense system," Master's thesis, University of California ,Davis, 2004.

[24] MarcusTylutki and KarlLevitt, "Mitigating distributed denial of service attacks using a proportional-integral-derivative controller," in *6th International Symposium on Recent Advances in Intrusion Detection(RAID)*, (Pittsburgh, PA, USA), September 8-10 2003.

[25] C. C. Zou, W. Gong, and D. Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," in *1st Workshop on Rapid Malcode (WORM)*, (Washington, DC, USA), ACM SIGSAC, October 2003.