A Distributed Firewall and Active Response Architecture Providing Preemptive Protection

J. Lane Thames
Georgia Institute of Technology
210 Technology Circle
Savannah, GA 31407
001-912-966-7922
lane.thames@gatech.edu

Randal Abler
Georgia Institute of Technology
210 Technology Circle
Savannah, GA 31407
001-912-966-7922
randal.abler@.gatech.edu

David Keeling
Georgia Institute of Technology
210 Technology Circle
Savannah, GA 31407
001-912-966-7922
dkeeling@gatech.edu

ABSTRACT

Firewalls provide very good network security features. However, classical perimeter firewall deployments suffer from limitations due to complex network topologies and the inability to completely trust insiders of the network. Distributed firewalls are designed for alleviating these limitations. Intrusion detection is a mature technology and is very powerful when coupled with active response, which is the act of responding to intrusions without the need of human advisory. This paper describes an architecture that implements a distributed firewall with distributed active response. A fundamental result of the architecture is that it can provide proactive and preemptive security for hosts that deploy the system. Using the open-source software framework, the software implementing this proposed system will be provided to the research community so that the architecture can be extended by other researchers and so that newcomers to network security can start investigating security concepts quickly.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks-*Security and Protection*

General Terms

Security

Keywords

Distributed Firewalls, Intrusion Detection Systems, Active Response

1. INTRODUCTION

The firewall is an extremely effective device that is used to protect computers and networks from attack. It is a device that allows or denies network connections to or from an entity based on the entity's security policy. The security policy is a set of statements that define legal network operations for the entity. A firewall can be either host-based or network-based. A host-based firewall is normally implemented in software that runs on the end host. The network-based firewall is a dedicated device placed in the network's ingress and egress locations, and this type is also known as the perimeter firewall. The classical assumptions of designing perimeter firewall systems constitute the following two principles:

- The network topology is well defined such that the administrator is aware of and controls all ingress and egress points within the network. Firewalls are placed in-line at all defined ingress/egress points, and these firewalls create perimeters such that we can define inside and outside perimeters and associated trust specifications.
- All users within the inside perimeter are assumed to be trusted.

These two principles can longer be used when designing secure networks. First, network topologies of today are very complex. Ingress/egress points can be established without the knowledge or control of the network administrator because of the wide availability of broadband access. It is trivial for end users within the inside perimeter of a network to create personal wireless local area networks or to establish Internet connections using DSL, cable, dial-up modem, or broadband over cellular technologies. Because of these situations, it is impossible for the network administrator to completely define all ingress/egress locations. Second, it is very naïve to assume complete insider trust, and one should not assume complete trust to all inside users. Host-based firewalls have become ubiquitous, partially due to the limitations of the classical firewall deployment as listed above. With a host-based firewall, there is no need to consider perimeters and topologies because the perimeter now exists at the host's network interface. Further, the host does not have to assume trust to any other machine within the local network.

A shortcoming of firewalls is that they are quasi-static devices. The security policy enforced by the firewall remains constant unless there is an explicit need to change the policy. Generic firewalls cannot adapt to real-time threats and attacks unless the administrator takes appropriate measures and applies new policies that target the attacks. Intrusion detection is technology that can monitor hosts and/or networks and provide administrators with alerts when attacks have been detected. The administrator can act on the alerts by configuring policies within the firewall system that counter the effects of an attack. However, the time needed for a human administrator to create and configure policies related to attack alerts is too large for modern day attacks. The solution needed is an automated, active response technology that applies dynamic security policies during the early stages of an attack. This paper discusses a distributed firewall and active response architecture whose goal is to bridge the gap between firewalls, intrusion detection, and active response. The paper is organized as follows: Section 2 will discuss background and related work. Section 3 will give an architectural description of the proposed distributed firewall and active response system. Section 4 discusses limitations of the architecture, and section 5 describes solutions for the limitations. Finally, conclusions are given in section 6.

2. BACKGROUND AND RELATED WORK

Previous discussions of distributed firewall architectures can be classified as two main types: 1) Architectures that employ centralized policy management with end-point enforcement and 2) Defense in depth architectures. A well known work discussing distributed firewalls was given by Bellovin [2], and the implementation details of [2] were described in [3]. Bellovin argues in [2] that conventional firewall designs, specifically the deployment of perimeter firewalls, suffer from the following issues: network topologies can not be well defined, insiders cannot be trusted, certain protocols such as the File Transfer Protocol (FTP) are not easily supported, firewalls cannot inspect encrypted payloads, and perimeter firewall deployments can cause bottlenecks and network performance issues. In Bellovin's proposed architecture, the security policy for an organization is centrally defined and managed, but enforcement takes place at the end-points, i.e. hosts, within the organization's network. Bellovin's solution can be implemented as a hybrid system where classical perimeter firewalls are used in conjunction with the distributed firewalls. Bellovin's distributed firewall requires three components: A policy language, a system management interface, and Internet Protocol Security (IPSec). The network administrator employs the management interface to instantiate the official security policy of the organization, which is defined using the syntax of the policy language. Most firewalls use the Internet Protocol (IP) address as a host identifier. However, Bellovin suggests using the cryptographic signature available with IPSec as the host identifier because they are independent of network topology and are not easily spoofed. Once the policy has been created in the management interface, a special compiler translates the security policies into a format that conforms to the native firewall syntax and ships the translated policies to the end-points.

The architectures described in [4] and [5] are defense in depth systems. In [5], the authors present a firewall network system specifically targeted for reducing the effects of computer worm propagation. Their network system divides an organization's network into many isolated subnetworks. End-points within the isolated subnetworks are classified by the administrator as either clients or servers. All firewalls within the organization are configured to have the same set of policies (firewall rules), and the policies are defined such that all network service requests sent to internal end-points will be blocked (denied) if the request is for a host that has not been defined as a server or if the request if for a service not being offered by the corresponding server. In [4], the authors suggest a cascade of firewalls within an organization's network. The idea is that end-points that require more protection are located on network paths such that communication flows to the end-point must traverse multiple firewalls, and the multiple firewalls in the path have increasing security policies providing added degrees of protection.

Multitudes of work have been published in the area of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). A survey and taxonomy of IDS technology is given in [1]. IDS is technology that can monitor hosts and/or networks and provide administrators with alerts when attacks have been detected. IPS is technology designed for two purposes: 1) To actively monitor network devices for known security vulnerabilities (also referred to as penetration testing), and 2) To automatically respond to real time attacks by way of issuing ***Active Response***. Active response is the act of automatically applying special security mechanisms without the need of human advisory when network intrusions and anomalies have been observed by a detection mechanism. However, most solutions available that provide intrusion detection and active response are expensive and only available in the commercial market. These solutions are proprietary, which can hinder academic research in this area. Further, most available solutions are targeted towards perimeter deployments, i.e. a perimeter firewall that contains IDS and IPS technology that incorporates active response. As previously noted, perimeter solutions do not provide the security needed in today's complex network topologies.

The solution proposed in this paper is an architecture that provides distributed firewall capabilities with active response. The contributions of the proposed architecture are the following:

- The software implementing the architecture will be provided to the research community using the open source development framework.
- The architecture was designed to avoid extreme complexity such that it is easy to use and understand for newcomers of network security research.
- The design is open-ended at both the top and bottom of the architecture such that it can be easily extended to incorporate advanced research and development in firewall and intrusion detection technology.
- The architecture bridges the gap between distributed firewalls, intrusion detection technology, and active response.

3. DISTRIBUTED FIREWALL AND ACTIVE RESPONSE ARCHITECTURE

The architecture was designed based on the concept of hosts within a trusted domain of administration that can detect anomalous behavior and create blocking policies against the anomalous host. These blocking policies are shared with the neighborhood members of the domain of administration. Once a policy is created by a host or received from a neighbor, the policy is translated into a firewall rule that denies access to or from the anomalous host. The fundamental design principle of this architecture is the following: *Once a source IP address has been classified with an anomaly detection mechanism as being untrustworthy, deny all access to or from the anomalous host for all members of the trusted domain of administration*. We define the anomaly detection mechanism as any tool that can be used to detect abnormal computer or network activity. Being classified as untrustworthy implies that the source of an attack has malicious intentions and can issue other harmful exploits against the hosts within the Trusted Domain of Administration (TDA). The TDA is defined as the set $\sum = \{S_1, S_2, \dots, S_M\}$ of hosts under the control of a single administrative authority. Figure 1 will be used to illustrate a discussion of the architecute. From Figure 1, the TDA is composed of $\sum = \{S_1, S_2, S_3, S_4, S_5\}$, where each S_i is some type of host possibly offering network services to the Internet. Further, we assume that each server is running a host based firewall, for example the UNIX based iptables [6]. As shown in Figure 1, the TDA can span across multiple Local Area Networks (LAN). The only requirement is that the hosts are administered by a single administrator or administrative domain, i.e. a group of administrators under a common authority. There is a purpose for the definition of the TDA. Trust is a critical aspect of this architecture. If trust is neglected, there is a possibility that false policies can be injected into the system such that denial of service could be issued on random hosts. Therefore, a well defined TDA is required. Using the definition of the TDA as given above, trust can be achieved. Since a server must "trust" its administrator (by default), and each server in the TDA trusts the same administrator, then each server in the TDA can trust one another as a consequence of transitive trust.

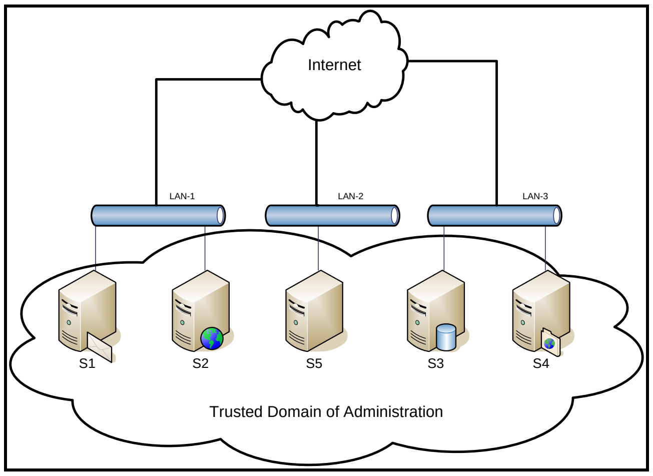

Figure 1. A trusted domain of administration

A simple description of this distributed firewall and active response architecture follows: Each S_i has a detection process that is monitoring for attacks. Once an attack is detected, the offending IP source address is added to a local database and a filtering rule is created within its local host-based firewall that denies all access from the offending IP address to the host. Then, the host sends this information to each of its neighbors within the TDA so that they can also add a deny rule for the offending address. This process is an act of distributed active response that implements preemptive protection. Consider the case where a malicious node issues an SSH dictionary attack against S_1. Once S_1 has detected the attack, a blocking rule will be created for the offending IP address, and after distribution the other neighbors will have a blocking rule too. At some time in the near future, the same malicious node issues a sequence of attacks against S_5. But, because this host has previously added blocking rules based on the distributed security policy received from S_1, it will not be affected by the attacks. Hence, S_5 has been preemptively protected. Figure 2 shows a block diagram of the distributed firewall and active response architecture proposed in this paper. The architecture is composed of 4 primary management modules: *Firewall Management, Policy Management, Distribution Management,* and *Autonomous Detection System (ADS) Management*. Further, the architecture contains 2 specification interfaces: *Firewall Rule Specification Interface (FRSI)* and *Policy Description Specification Interface (PDSI)*.

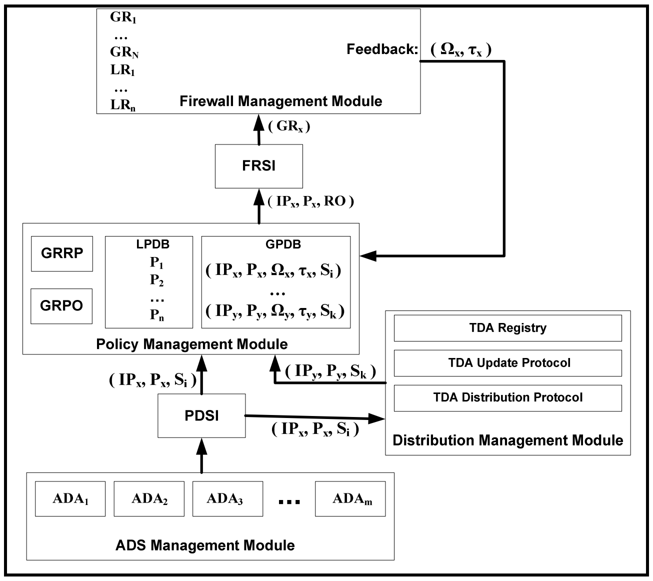

Figure 2. Block diagram of the architecture.

This system is designed to be completely distributed with no centralized control. Each member of the TDA must run a system implementation that conforms to the architecture's design specifications. However, the implementation is system independent and can be developed with whatever technology the developer chooses.

Autonomous Detection System Management Module: The ADS management module is simply the collection of independent Autonomous Detection Agents (ADA) that are active on the local system and the collection of rules that define how the ADA must interact with the policy management module. An ADA is an autonomous system that can detect computer and network abnormalities. The ADA can be as simple as a process monitoring the local host for failed login attempts to a full scale IDS deployment such as the SNORT IDS [7]. The ADS was designed to be highly modular and extensible so that researchers can easily investigate new abnormality (anomaly) detection theories and test the active response of their designs. Further, the modularity allows for many different types of independent, autonomous abnormality detectors to reside on a single host thereby providing a more robust defense in depth strategy for the local host and the overall TDA. Currently, the architecture specifies one rule within the ADS: the ADA must be implemented such that it adheres to the contract offered by the PDSI.

Policy Description Specification Interface: The PDSI is an interface that offers a globally recognized policy description that all members of the TDA interpret with the same meaning, i.e. it is a formal syntax describing a policy (a policy language). The following is the PDSI contract: **(IP_x, P_x, S_i)**. The syntax is a 3-tuple containing the IP address of the attacking host (IP_x), the policy's action to take against the attacking host (P_x), and the host identifier (S_i). The host identifier is some value that uniquely identifies the host that generated the Policy Description Specification (PDS). P_x is a policy action construct. The architecture currently defines only one action, P_x = DENY. Future extensions can be made whereby one might offer rate-limiting definitions. However, P_x shall never be extended to offer an operation that elevates a security privilege, i.e. P_x = ALLOW. The ADA issues the PDS to the policy management module and the distribution management module via the PDSI.

Policy Management Module: The Policy Management Module (PMM) contains 4 components: *Local Policy Database (LPDB), Global Policy Database (GPDB), Global Rule Removal Protocol (GRRP),* and *Global Rule Placement Optimizer (GRPO)*. The LPDB is the collection of security policies that have been defined by the system administrator for the local host. For example, if the local host is a web server, the list of local policies could be stated like the following: {allow from any to localhost service http; deny from any to localhost service any}. The policy states that the local host can receive inbound HTTP connections and all others are to be denied. The GPDB is the collection of 5-tuples having the following form: **(IP_x, P_x, Ω_x, τ_x, S_i)**. This list of variables represents the IP address of the anomalous host, the policy action to apply against the anomalous host, the firewall hit-count (Ω_x), the firewall hit-time (τ_x), and the host identifier. The firewall hit-count and hit-time are feedback variables that are received from the local host's firewall module and are used by the GRRP and GRPO. The architecture defines that the hit-count is initialized to a value of 1 and the hit-time is set to the local host's current system time upon insertion into the GPDB. These values are recalculated over time by the firewall management module. The GRRP and GRPO are responsible for removal of global firewall rules and optimization of their placement within the firewall rule base, respectively. These functions will be described in more detail in section 5. The IP address, policy action, and host identifier within the GPDB are received either by the local PDSI or the distribution management module. The ones received via the distribution management module represent the policies received from one of the neighboring servers within the TDA. Once new policies are received by the PMM, they are inserted into the GPDB. After insertion into the GPDB, the policy is sent to the FRSI for insertion into the firewall rule base.

Firewall Rule Specification Interface: The FRSI offers a contract between the Firewall Management Module (FMM) and the PMM, and the PMM must be implemented such that it conforms to the contract. The FRSI's contract has the following form: **(IP_x, P_x, RO)**. The 3-tuple contract contains the IP address of the anomalous host, the policy action to apply against the anomalous host, and the Rule Operation (RO). Three types of rule operations have been defined: {RO = Insert, RO = Delete, RO = Modify}. These define the firewall operations of inserting a new rule, deleting an existing rule, or modifying an existing rule. The FRSI is responsible for dynamically mapping the data contained within a received contract into a firewall rule based on the firewall's native syntax. Using an interface such as this offers extensibility because the architecture will not be constrained to one type of firewall product. The FRSI can be modified by the system implementer to map the data to the syntax of the desired firewall to be used.

Firewall Management Module: The FMM is responsible for dynamically configuring the firewall rule base when rule structures are received from the FRSI. Further, the FMM collects the hit-count and hit-time feedback variables and sends these variables to the PMM. The hit-count represents the number of times that a rule within the set of global rules has been matched (hit) by a particular incoming IP address and the hit-time represents the local system time of the last hit.

Distribution Management Module: The DMM receives the 3-tuple **(IP_x, P_x, S_i)** either from the local host or from one of the local host's neighbors within the TDA. If the tuple is received from the local host, it is immediately distributed to the other members of the TDA. And, if it is received from one of its neighbors, it is immediately sent to the PMM for processing. The DMM contains three components: *TDA Registry, TDA Update Protocol,* and *TDA Distribution Protocol*. The TDA registry stores host identifier information. At minimum, this is a list of all the members of the TDA. The identifier values are not strictly defined by the architecture. However, the suggested values are S_i = { IP_i, $Host\_Name_i$, $Public\_Key_i$ }, which is a set containing a member's IP address, an assigned host name (possibly its DNS name if available), and its public encryption key. Of course, if encryption keys are used, an agreed upon encryption standard must be designated. The TDA update protocol defines policy update procedures. The architecture currently defines one procedure, which is to update the GPDB and firewall rule base immediately upon receiving new policies. However, future implementations can define procedures for periodic update intervals of the GPDB whereby the host can send copies of its local GPDB to other members of the TDA. Then, a TDA member can contain its GPDB and a copy of other members' GPDBs. The copies represent a "global" view of attack activity and can be used by the GRRP and GRPO for making more intelligent removal and optimization decisions. Finally, the TDA distribution protocol defines how the TDA members will distribute information between each other. This is not strictly defined by the architecture, but encrypted communication streams are suggested. For example, one can use the SSH protocol as the delivery mechanism.

4. LIMITATIONS OF THE ARCHITECTURE

There are some limitations of this architecture: The possibility of a large number of firewall rules, Dynamic Host Configuration Protocol (DHCP), and Network Address Translation (NAT).

As seen from the FMM in Figure 2, there are a total of n + N firewall rules configured within a local host's firewall. The n LPDB policies map to the n local firewall rules (LR), and the N GPDB policies map to the N global rules (GR). Each member of the TDA will contribute some proportion of the total number of policies within the GPDB and the total number of global rules in the firewall rule base. As the number of TDA members increases, the value of N will increase. Normally, n will be a modest value that remains approximately constant for a host based firewall. So, as N increases it becomes the dominating factor of n + N. Each incoming connection to the host must be processed by the firewall. Most firewall implementations will start at the top of the rule list and process down the list until a match is found. Once a match is found, the firewall makes an access control decision based on the matched rule's definition. Meanwhile, other connections into the host must wait in a connection queue until it is their time to be processed by the firewall. The amount of time needed by the firewall to process a connection can be considered as a queuing service time S_T. This service time is probabilistic in nature. However, in certain cases, as the number N of global rules increases, S_T can be approximated as being proportional to N. This implies that the waiting time W_T of a connection within the connection queue will also be proportional to N. Therefore, there is a possible degradation of performance for the host as the number of firewall rules increases. This is most significant when the host is an Internet server. The case where the above assumption holds is for a host that has a small probability that a network connection C is abnormal, i.e. a host that is not attacked very often. Mathematically, this assumption is stated as follows: $\Pr(C = \text{anomalous}) \approx 0$. For this particular case, as N increases, the average service time for a connection being processed can be described by Equation 1.

$$E[S_T] = \varepsilon N + \delta \qquad (1)$$

In Equation 1, δ is the average amount of time needed for a connection to be processed through the localhost's TCP/IP stack and εN is the average amount of time needed for a connection to be processed by the firewall. Given a connection queue of length η, the average waiting time W_T for connections in the queue will be proportional to $\eta E[S_T]$:

$$E[W_T] = \gamma\eta E[S_T] \qquad (2)$$

Figure 3 shows a plot of the connection service time as a function of N for a host running the Redhat Linux operating system and using an iptables firewall.

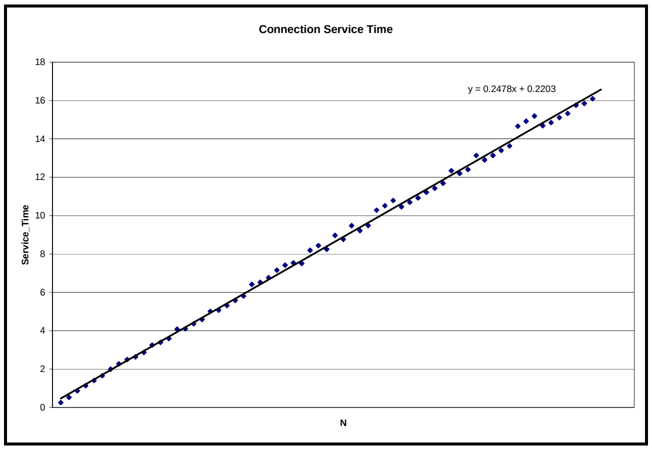

Figure 3. Experimental results for a firewall service time.

The data shown in Figure 3 reveals the service time for a connection through a firewall having a number rules from N = 0 to N = 65535. The best fit line's equation is $S_T = 0.2478*N + 0.2203$, where time is in units of milliseconds. For this host, the time for the connection to be processed by the TCP/IP stack, i.e. N=0, is 0.2203 milliseconds and $\varepsilon = 0.2478$.

DHCP is a protocol that is used to dynamically configure a host's networking parameters upon bootstrap time. When the host boots, it queries a DHCP server requesting its network parameters, one of which is its IP address. A DHCP server assigns a lease time for the IP address that it offers to a client. Once a host has obtained an IP address from the DHCP server, it must renew its IP address when the lease time expires. Upon renewing its IP address, it may or may not receive the same IP address that it received previously. Many Internet Service Providers (ISP) use DHCP for managing their IP address ranges. The possibility that a host will not receive the same IP address when its lease time has expired causes problems with the proposed architecture. Consider the case where a host within some ISP's address range is classified as being anomalous by a TDA member. This offending host will be blocked by the members of the TDA. But, at some time in the future, the host's least time expires and it retrieves a new IP address. The host's original IP address will eventually be assigned to another customer of the ISP. At this point, the TDA will be blocking access to an IP address that now belongs to another host that did not attack the TDA. Hence, the TDA is causing denial of service to the current owner of the IP address.

NAT is a technology implemented in network routing devices that maps internal IP addresses to external IP addresses. NAT can be configured to do one-to-one mapping or many-to-one mapping. With one-to-one mapping, the routing device maps a host's internal IP address to a constant, external IP address. The one-to-one NAT mapping does not impose limitations to this architecture. However, with the many-to-one mapping scheme, the routing device will map many internal IP addresses to a single external IP address. The many-to-one mapping can cause issues with the architecture. Consider the case where some host within a many-to-one NAT based network is detected as being anomalous. When the system applies a blocking rule to the anomalous IP address, the system will be blocking access to all hosts within the NAT based network.

5. SOLUTIONS FOR THE LIMITATIONS

The GRPO and GRRP components of the PMM were designed to solve, to some degree, these limitations. The GRPO component is designed to optimize the placement of the global rules within the firewall rule base for the purpose of reducing the connection waiting time during the event of an attack. This optimization is used to help alleviate the issue of large numbers of global rules. The global rules are of the following form: {deny from IP_x to localhost service any}. All of the global rules must be processed in the firewall first, and then the local policy rules must be processed. This is a consequence of the logical ordering of rules required in an access control list. An example will be given to illustrate. Consider the following firewall rule list:

```
{
        deny from 1.2.3.4 to localhost service any;
        allow from any to localhost service http;
        allow from any to localhost service ssh;
        deny from any to localhost service any;
}
```

This sequence of rules states that IP address 1.2.3.4 should not be allowed to access any services on the localhost. But, all other IP addresses are allowed access to HTTP and SSH. The last rule is the default rule that is used to deny access to any other service on the localhost. If the first rule is placed after the second rule, then 1.2.3.4 would be able to access HTTP. However, that defeats the goals of this architecture, which is to deny access to any service on the localhost once a host has been classified as anomalous. Hence, all of the global rules generated with the ADS or received from another TDA member must be processed before the local policy rules. The idea behind the GRPO is that the anomalous IP addresses are monitored by the FMM, and the "heavy-hitters" (the anomalous IP addresses) that are currently attacking the host have their global rules placed at or near the top of the global rule list. For this to be achieved, the hit-count and hit-time feedback variables are used to make intelligent placement decisions. The FMM sends the hit-count and hit-time to the GPDB once a match is made in the firewall rule base for the anomalous IP addresses. The GRPO component uses the following algorithm to optimally order the rules in the global rule list.

Algorithm GRPO:

```
{
        $X := \{GPDB.(IP_i, \Omega_i, \tau_i)\}$;
        $\Delta\tau_i := \tau_c - \tau_i$;
        $Z := \{X: \Delta\tau_i < T_0\}$;
        $Y := X - Z$;
        $maxsort(Z, \Omega_i), minsort(Y, \Delta\tau_i)$;
        $X \leftarrow Z \| Y$;
        $UpdateRules(X. IP_i)$;
}
```

The set X is extracted from the GPDB and all of the members that satisfy $\Delta\tau < T_0$ are extracted from X and stored in the set Z. The temporal optimization parameter T_0 defines a threshold value for the time range calculated in $\Delta\tau$. $\Delta\tau$ is defined as the time difference between the current system time, τ_c, and the last time, τ_i, that the anomalous host IP_i has attempted a connection. This implies that Z contains the 3-tuples for some set of anomalous IP addresses that have attempted to connect within the last T_0 seconds. Y contains the remainder of 3-tuples for the anomalous IP addresses that have not attempted a connection recently, where recently is defined by T_0. The maxsort() function takes the list of 3-tuples and sorts them using the hit-count variable as the sorting key. This list is sorted from maximum hit count to minimum hit count. The minsort() function takes the list of 3-tuples and sorts them using the hit-time variable as the sorting key. This list is sorted from minimum $\Delta\tau$ to maximum $\Delta\tau$. Then, X is replaced with the concatenation of Z and Y. Finally, the UpdateRules() function updates the logical arrangement of global rules in the firewall based on the updated ordering of IP addresses stored in X. The overall result is that the most recent hitters are sorted in such a way that the ones that are *currently* hitting the most are near the top of the list. This reduces the queue waiting time for normal connections because the anomalous connections of a currently attacking host are processed quicker.

The GRRP is designed to remove global rules that have become stale, possibly as a result of DHCP or NAT. Considering this, the architecture needs to remove rules for hosts that have not attacked the host *recently*. Removing stale rules will alleviate the issues with large numbers of rules, DHCP, and NAT. GRRP is designed to remove global rules based on the 2 variables, Ω_x and $\Delta\tau_x = \tau_c - \tau_x$, associated with an IP address, IP_x. If it has been a long time since IP_x has attacked the server, then $\Delta\tau_x >> 0$ and we want to consider removing the global rule for IP_x. However, we want to also consider how often IP_x was attacking the server, and this information is contained in Ω_x. We define the rule removal optimization parameter ω_x as the following:

$$\omega_x = \frac{\Delta\tau_x}{\Omega_x} \tag{3}$$

Equation 3 allows ω_x to be normalized relative to the hit count for a particular attacking host IP_x. By definition, $\Omega_x \geq 1$. Hence, ω_x will be bounded by $\omega_x = \Delta\tau_x$ and $\omega_x = 0$, for all $\Delta\tau_x \geq 0$. We define the rule removal timing threshold T_R as a time interval such that if $\omega_x > T_R$ then the rule should be removed. The following is a formal statement of the algorithm.

Algorithm GRRP:

```
{
        $\forall x \in X$
         If $(\omega_x > T_R)$ Then Remove(GPDB[x], FR[x]);
}
```

In this algorithm, X is the set of all IP addresses associated with the GPDB and the firewall rule (FR) base. The instance x can be used to index into both the GPDB and the FR base. Observing the dynamics of Equation 3, one can see that if host x issues many attacks (implying that $\Omega_x >> 1$) then the rule will remain in the GPDB and FR base for larger amounts of time because $\Delta\tau_x$ will need to grow larger in order to satisfy $\omega_x > T_R$. Hence, the policies and rules will remain active longer for hosts that are more active with their attacks.

6. CONCLUSION

Because of the ubiquity of host-based firewalls, there is a unique opportunity for deploying scalable distributed firewalls and active response systems. This paper described an architecture that can be used to implement such a system. The architecture allows for hosts and servers on the Internet to be proactively and preemptively protected. Further, the software implementing this architecture will be distributed using the open-source software model to network security researchers, practitioners, and security newcomers with hopes of creating new dialogues for future extensions to the architecture and allowing novice security researchers to quickly deploy and experiment with network security concepts.

7. REFERENCES

[1] Axelsson, S., Intrusion Detection Systems: A Survey and Taxonomy, Technical Report, pp. 99-115, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000

[2] Bellovin, S.M., Distributed Firewalls, ;login:, Vol. 24, pp. 37-47, November 1999

[3] Ioannidis, S., Keromytis, A.D., Bellovin, S.M., Smith, J.M., Implementing a Distributed Firewall, In Proceedings of Computer and Communications Security (2000), CCS'00 pp. 190-1999

[4] Smith, R., Chen, Y., Bhattacharya, S., Cascade of Distributed and Cooperating Firewalls in a Secure Data Netwrok, IEEE Transactions on Knowledge and Data Enginnering, Vol. 15, NO. 5, pp. 1307-1315, 2003

[5] Zou, C., Towsley, D., Weibo, G., A Firewall Network System for Worm Defense in Enterprise Networks, Technical Report: TR-04-CSE-01, University of Massachusetts, Amherst, 2004

[6] Iptables Firewall, http://www.netfilter.org/

[7] SNORT IDS, http://www.snort.org/