Managing and Securing the Global Multicast Infrastructure

Prashant Rajvaidya, Krishna N. Ramachandran and Kevin C. Almeroth
Department of Computer Science
University of California–Santa Barbara
Santa Barbara, CA 93106-5110
E-mail:{prash, krishna, almeroth}@cs.ucsb.edu

Abstract

A lack of mechanisms to monitor and manage multicast networks has adversely affected progress in several areas critical for successful deployment. One such area involves discovering and solving multicast security vulnerabilities. Although a number *of vulnerabilities exist, the most troubling are a set of easily exploited Denial-of-Service (DoS) attacks. The main reason for this concern is that the one-to-many nature of multicast can significantly magnify the effects of these attacks. Among the possible multicast DoS attacks, those that target the the Multicast Source Discovery Protocol (MSDP) can be most damaging. MSDP vulnerabilities are unusually easy to exploit and can lead to infrastructure-wide damage. In this paper, our goal is to develop a security framework that protects against DoS attacks through detection and then "deflection". In developing our framework, we first examine the vulnerability of multicast protocols, to DoS attacks. We use data collected with our global monitoring infrastructure,* Mantra*, to analyze the nature and effects of attacks that have already occurred. We then create additional, more virulent strains. Finally, we propose a family of solutions to detect and deflect the effects of each attack. Our techniques are evaluated by simulating their effectiveness against both real and simulated workloads.*

Keywords*: Multicast Security, Network Security, DoS Attacks, MSDP, Multicast Monitoring, Network Management*

1 Introduction

Assuring the reliable and efficient operation of the multicast infrastructure requires mechanisms to manage and monitor a complex system of protocols at an inter-domain level. However, there is only limited work in developing network management and monitoring systems that are suitable for this purpose[1]. Most of the existing systems are either not up to date with current developments and/or have limited application scope. The utility of current systems is especially limited both in monitoring security vulnerabilities as well as in providing information that can be used to deter attacks in real time.

Securing the infrastructure for a new service like multicast should be an integral component of any effective management solution. Among the few multicast management solutions that do exist[1], almost none incorporates mechanisms to provide security. As a result, many multicast security vulnerabilities have been neither discovered nor rectified[2]. Furthermore, although a growing number of network administrators are deploying multicast[3], without effective monitoring and management mechanisms, operational experience in how to detect and control attacks is often lacking.

A lack of effective mechanisms for monitoring and managing network security is more damaging for emerging Internet technologies like multicast. Newer technologies are typically more vulnerable to security breaches because their protocols are in the early stage of development and are neither well-designed to handle attacks nor tested in real attack situations. In the specific case of multicast, security vulnerabilities and damage from attacks can be significantly greater than for other evolving services. This is because multicast networks operate on top of the existing Internet infrastructure and, as a result, inherit most of the security problems of the current Internet. Moreover, as the multicast service model is fundamentally different from the unicast model, multicast networks introduce several additional vulnerabilities.

Among known multicast security problems, vulnerability of the infrastructure to Denial-of-Service (DoS) attacks is of prime concern and the main focus of this paper. There are two main reasons for this concern. First, the broadcast nature of multicast magnifies the effects of DoS attacks by a magnitude equal to the number of receivers. Incidences have already occurred in recent years where simple attacks were magnified to such a degree that their effect was infrastructure-wide and crippling. Second, since much of the multicast infrastructure uses links and devices that provide both unicast and multicast connectivity, a DoS attack that exploits the vulnerabilities of multicast can potentially hamper the routine operation of other services in the Internet.

In this paper, we investigate some of the security vulnerabilities of the existing multicast infrastructure. We focus specifically on the vulnerabilities of the Multicast Source Discovery Protocol (MSDP)[4]. MSDP is a good place to start because: (1) it has already been attacked, (2) it is key to the operation of the current infrastructure, and (3) it provides insight into how other evolving Internet services can be protected. Our study of MSDP DoS attacks first analyzes how the different characteristics of the protocol can be exploited for attacks. This is followed by an investigation into the effects of two different MSDP DoS attacks that severely hampered multicast operation, the first in January 2001, and the second in January 2003. These investigations are based on analysis of data collected from several important locations in the network through our global monitoring infrastructure, Mantra[5, 6, 7]. Using this analysis, we evaluate the differences between the two attacks and identify the basic characteristics of each of the attacks. We, then, devise more virulent attacks that can accomplish the same damage but are harder to detect.

Our work to develop solutions against MSDP-based DoS attacks focuses on guarding against the types of attacks that have already occurred as well as against more virulent strains that could exist. We describe several mechanisms for *detection* and *deflection*; i.e. countering the effects of various strains. Our solutions use straightforward anomaly detection schemes that are less complex than the schemes commonly used for attack detection in conventional (unicast) networks[8]. The simplicity of our solutions keeps their resource requirements low, which makes our solutions viable for operation even within core network routers. Nevertheless, our solutions can learn from MSDP usage trends and automatically adapt to changing traffic patterns. In the latter half of this paper, we evaluate the effectiveness of some of these solutions by first modeling workloads for attack traffic and then simulating the solution response.

The rest of the paper is organized as follows. In Section 2 we give a brief overview of the vulnerabilities of the multicast infrastructure and work in securing it. Section 3 investigates possible DoS attacks that exploit the vulnerabilities of MSDP. In Section 4 we describe our mechanisms for detecting and deflecting these attacks. In Section 5 we describe the setup that we have used for evaluating these mechanisms including workloads and our simulator. Evaluation results are presented in Section 6 and the paper is concluded in Section 7.

2 Inter-Domain Multicast and its Vulnerability to Attack

Since its creation, the multicast infrastructure has undergone several fundamental changes: from a dense mode overlay network–called the MBone–to native sparse mode deployment, and finally to the latest efforts in the direction of Source Specific Multicast (SSM)[9, 10]. During the early years, the focus was on protocol design issues like the efficient creation of distribution trees, congestion control, reliability, and robustness. More recently, efforts are concentrated on developing new group paradigms, debugging existing protocols, and ensuring widespread deployment. As a result of these efforts, multicast protocols have matured, widespread acceptance has grown, and deployment has increased[3]. However, amidst these developments, work in securing the multicast infrastructure has been largely overlooked.

In the current infrastructure, most of the multicast protocols have at least some security vulnerabilities and security problems exist in every aspect of the topology, from end-hosts to core routers. In this section, we list some of these security problems, discuss how various vulnerabilities can be exploited, and describe the effects of some of the attacks on the operation of multicast. We begin by presenting an overview of the operation of the current infrastructure.

2.1 Operation of Current Multicast Infrastructure

Multicast is a network model for one-to-many and many-to-many delivery of high-bandwidth streams in a scalable and bandwidth-efficient manner. This model uses the notion of a logical group of participants, called a *multicast group*, that are interested in content being sent to them. Each such group is identified using a Class-D IP address that is usually chosen at the sender host. The sender simply sends data packets to a group and the network forwards them to the participants of that group by replicating these packets at appropriate nodes in the topology.

Most of the current infrastructure only supports a service model called Any Source Multicast (ASM). ASM primarily relies on four protocols[11]: the Internet Group Management Protocol (IGMP)[12], the Protocol Independent Multicast (PIM) protocol[13], the Multicast Border Gateway Protocol (MBGP)[14], and the Multicast Source Discovery Protocol (MSDP)[4]. MBGP acts as an inter-domain route exchange protocol that allows the propagation of reachability and path information across domains. PIM uses this route information to create and manage distribution trees through which data is transferred from source(s) of a group to its participants. Hosts initiate group join and leave operations using IGMP. Finally, MSDP acts as a source announcement protocol and

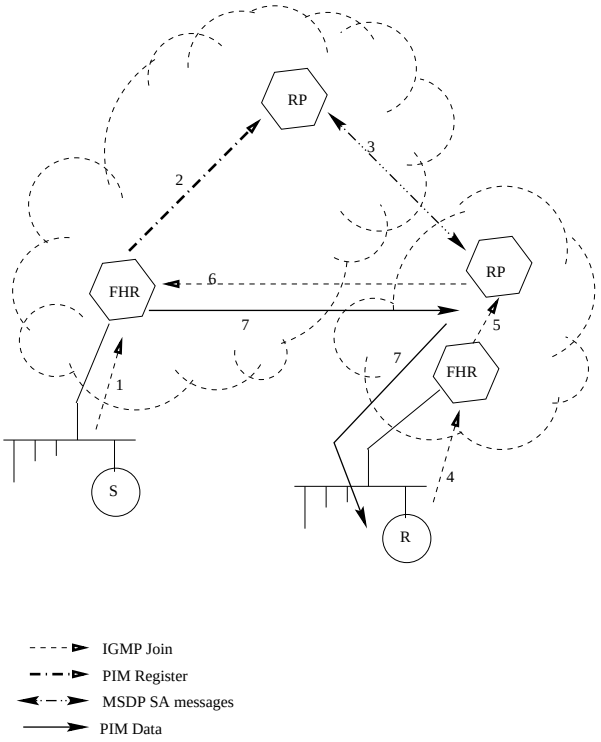

Figure 1: Multicast data delivery in ASM.

is responsible for propagating information about active sources across the entire infrastructure.

Figure 1 presents a simple illustration of the steps involved in the process of multicast data delivery in ASM. When a source (S) starts transmitting data to a group (G), the First Hop Router (FHR) for the source generates a PIM register message corresponding to the source-group pair, usually referred to as (S,G). This register message is sent to the PIM Rendezvous Point (RP) that also acts as an MSDP router for the domain. From here, through the exchange of MSDP Source Active (SA) messages, information about the source is propagated to the rest of the infrastructure. When a host (R) wishes to receive the content being transmitted to a particular group, it sends an IGMP join request to its own FHR. The FHR, in turn, sends a PIM join message to the RP in its domain. Finally, the PIM distribution tree is extended to the receiver so that is starts receiving data from the source.

One of the biggest drawbacks of the current deployment is that it requires networks to have knowledge about, and maintain state for, all multicast sources that are actively sending traffic to any group. This knowledge is critical, because in order to support many-to-many data delivery, networks must assume the responsibility of knowing the IP addresses of a group's source(s). This responsibility is satisfied via the functionality provided by MSDP. MSDP propagates and maintains information about *all* active sources within the infrastructure. Not only does this result in added overhead for network devices, but, as we will soon describe, is one of the main multicast security vulnerabilities.

2.2 Security Vulnerabilities in Multicast

Multicast has not been designed with much thought to security. Just like unicast, multicast allows some security mechanisms to be built on top of existing protocols. However, progress in developing and deploying such mechanisms has been quite slow. Existing multicast security problems pertain to three areas: (1) securing data from eavesdroppers; (2) providing admission control; and (3) securing protocol control traffic. These are described in further detail below.

Securing data from eavesdroppers: One of the elementary security problems that arises in multicast networks pertains to securing data from eavesdroppers. Although this problem is the same for unicast networks as well, it becomes harder to solve for multicast because conventional solutions like IPsec and other encryption mechanisms cannot be easily applied to multicast data delivery. This is the case because solutions such as IPsec rely on key management and distribution related tasks. These tasks are relatively easy to perform for end-to-end data delivery in unicast. However, in multicast, these tasks suffer scalability challenges due to large groups and dynamic group membership. Work is ongoing in this area [15, 16].

Providing admission control: Admission control to multicast groups would allow a group manager, possibly the primary source, to decide which hosts could participate in a group. However, no such mechanism exists and, as a result, several types of attacks can be launched simply because end hosts are able to join groups arbitrarily. For example, a host can start sending bogus high bandwidth data to a popular multicast group as a form of denial-of-service. Other kinds of attacks are possible when groups implement additional services on top of data delivery like congestion control and reliability. A single group member, incorrectly claiming high loss or congestion, can cause a source to significantly reduce its transmission rate.

Securing protocol control traffic: All of the multicast protocols used in the Internet today have security vulnerabilities. These vulnerabilities stem from the weaknesses of the control and reporting mechanisms that the protocols use. Malicious control traffic can be injected into the network to cause applications at the participant hosts to malfunction, disrupt data delivery for the entire group, or to flood the network with erroneous control traffic. IGMP's security problems exist because the query messages and membership reports that it uses can be spoofed to create inaccurate state about group membership. For example, forged leave messages can cause the generation of numerous bogus query messages from the router. Another example is that forged join messages or membership reports can cause a router to create and maintain unnecessary state. For PIM, forged control messages can be used to alter the structure of multicast distribution trees and also to hamper the data distribution across them. Vulnerabilities in MBGP are mainly due to the susceptibility of the underlying BGP protocol[17]. Finally, vulnerabilities in MSDP exist because control traffic–Source Active (SA) messages–can easily be spoofed

and sent throughout the infrastructure. Therefore, a single multicast host can cause a large number of bogus SAs to be generated and sent to every MSDP router in the infrastructure. Although, this is a very basic DoS attack, it can have widespread negative repercussions.

3 Vulnerabilities of MSDP to DoS Attacks

Identifying the vulnerabilities of MSDP is a critical need in securing the multicast infrastructure. MSDP vulnerabilities stand out because they are very easy to exploit and because the scope of their effects is typically infrastructure-wide. With very little effort, MSDP peers and, even worse, the whole multicast topology can be incapacitated with excess load.

In this section, we first describe the general operation of an MSDP-based DoS attack. Then, we describe inadvertent attacks that were caused by two Internet worms: (1) the *Ramen Worm*; and (2) the *Sapphire Worm*. Both of these attacks, although they *did not intend to affect multicast*, triggered a large SA storm in the infrastructure. We describe the operation of these worms, analyze the effects that they had on the MSDP infrastructure and discuss their differences. Finally, we generalize these differences to present characteristics of attacks and, thus, identify the factors that can be varied to generate different "strains" of attacks. An understanding of possible attacks through the study of these worms and the subsequent generalizations sets the stage for the following section where we develop possible detection and deflection solutions.

3.1 Operation of MSDP-Based DoS Attacks

Most MSDP-based DoS attacks aim to incapacitate the network by flooding it with bogus source active (SA) messages. In this respect, a particular vulnerability of MSDP is that any host on an MSDP-enabled network can start an infrastructure-wide attack with minimal effort. This is because of two fundamental characteristics of MSDP: (1) every time a source in a multicast-capable network first sends an IP packet to a new Class D address, an MSDP peer in the domain sends a corresponding SA announcement to all of its peers; and (2) every MSDP peer broadcasts all SAs it receives to every other connected MSDP peer. The first operation allows generation of bogus SAs and, hence, makes initiating an attack easy. The second operation enables these SAs to be propagated across the entire infrastructure. Therefore, a single host can make its local MSDP peer generate a large number of SA announcements, flood the multicast infrastructure with these announcements, and cause every MSDP peer to process these messages.

Figure 2 illustrates an example of this kind of attack. A host can send one packet to each of a large set of Class D addresses in quick succession. Regardless of the size and contents of these packets, the First Hop Router (FHR) for this host will consider them to be valid and forward them to the RP that is the MSDP peer for the domain. This RP will then generate one SA for each unique Class-D address. Thereon, SA forwarding among peers will eventually propagate the SAs throughout the multicast infrastructure.

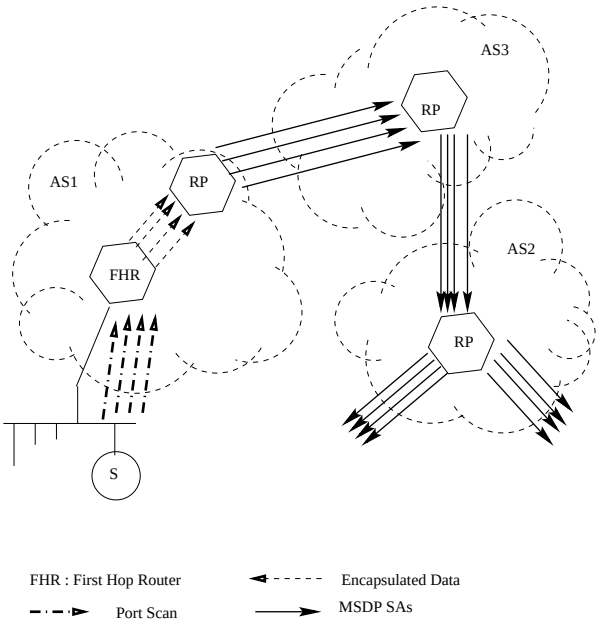

Figure 2: A simple MSDP-based DoS attack.

3.2 Previous MSDP-Based DoS Attacks

On quite a few occasions, DoS attacks have overwhelmed the multicast infrastructure with large numbers of SA messages. These attacks were concentrated in January 2001 and then again in January 2003. On both occasions, Internet worms were responsible for the attacks. While the former was caused by a worm called Ramen, the latter was the result of a worm called Sapphire (also known as Slammer and SQLExp)[18]. Attacks triggered by these worms relied on the basic mechanism that is illustrated in Figure 2. The attack consists of a single host sending data to a large number of class-D addresses and thus flooding networks with bogus SAs. Interestingly, these attacks were not targeting multicast. These worms, intended to just port-scan a random set of IP addresses. While the Ramen worm aimed to discover a particular vulnerability in Linux hosts, scanning by the Sapphire worm aimed to discover a vulnerable version of a popular database engine. Regardless of the original intentions, these worms triggered a flood of bogus SA messages because several of the addresses that they scanned belonged to the multicast (Class-D) address space.

We have been able to gauge the impact of these attacks on the global infrastructure with the help of data collected from multiple routers using our global monitoring infrastructure, Mantra[5, 6, 7]. Furthermore, we have also been able to substantiate our claims that MSDP DoS attacks can rapidly span the entire infrastructure and can have serious negative repercussions. Figure 3 plots the number of SA messages seen in the infrastructure in the one month period starting on January 12, 2001. Similarly, Figure 4 plots the number of SAs observed during a 24 hour period starting at 4:00PM (PST) of January 24, 2003. Both sets of results are based on the aggregate view of MSDP tables collected from multiple routers at 15-minute intervals. Results for the Ramen worm show that there

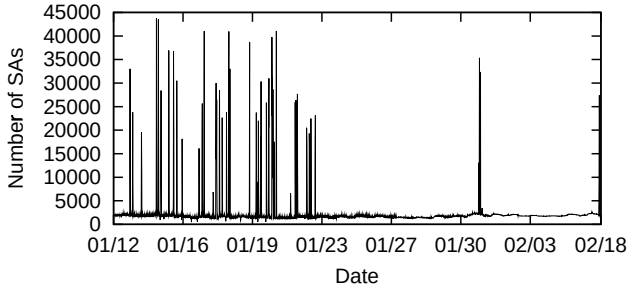

Figure 3: Effects of attack because of Ramen worm (January 2001).

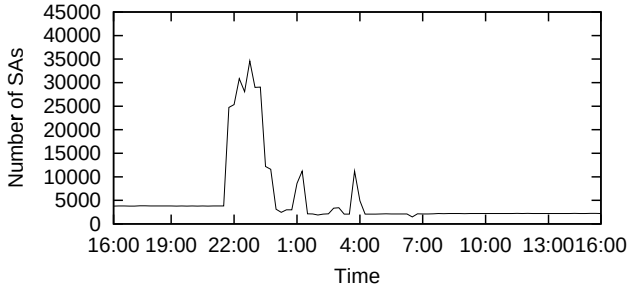

Figure 4: Effects of attack because of Sapphire worm (January 24, 2003).

were 40 distinct attacks over a period of 20 days. During this period, the number of bogus SAs generated because of these attacks ranged from 10,000 to about 45,000. On the other hand, in the case of the Sapphire worm, only four distinct attacks were observed, all within a period of 10 hours. The number of bogus SAs ranged from just about 5,000 for the weakest attack to about 34,500 at the peak of the most severe attack.

The differences between the Ramen and Sapphire worms are not limited to the size of the SA floods that they cause. Detailed analysis of our data sets reveals that the characteristics as well as the effects of these attacks were quite different. First, the Class-D addresses that the two worms scanned were different. SAs generated by Ramen were a contiguous /16 range of addresses, i.e. 2^{16} consecutive IP addresses, starting from a randomly selected IP address. Addresses for SAs generated by the Sapphire worm were chosen randomly using a linear congruent, or power residue, pseudo random number generator algorithm[18]. Another important difference between the two worms was the number of simultaneous locations from which the attacks originated. All 40 instances of the Ramen worm originated from different domains at different times. On the other hand, several instances of the Sapphire worm attacks were launched simultaneously from multiple locations. As a result their effect on the multicast infrastructure was distributed in nature. Finally, the duration of attacks in the infrastructure was different as well. While the effects of each instance of the Ramen worm were observed in the infrastructure for about 30 minutes, for Sapphire, the durations varied from 30 minutes to about 2 hours. Durations were different mainly because some instances of the Sapphire worm were triggered by multiple hosts and attacks from all these sources did not start at the same time. Therefore, propagation of their SA floods started only one after another causing the overall

effect to last longer.

Another interesting observation about these two worms is that even though the attacks are short-lived, the damaging effects typically last much longer. Given the rate at which both worms scanned unique IP addresses, hosts compromised by either of the worms would have generated the entire SA flood in close to a minute. However, Figures 3 and 4 show that SAs from the flood were present in the infrastructure for at least 30 minutes. The reason is that MSDP takes time to propagate SA messages and will continue to do so long after an attack stops. In essence, the MSDP topology is processing and forwarding the backlog of SA messages until all have been handled. Moreover, as we will explain later, once a router receives bogus SAs, it stores them in its local cache prolonging the effects of an attack even further.

3.3 Characteristics of MSDP-based DoS attacks

The attack scenarios described thus far have underlined the fundamental principle behind most MSDP-based DoS attacks, i.e. flood the network with bogus SAs. However, different attack strains are possible. Identifying the characteristics of MSDP-based DoS attacks is therefore important for identifying the damage a particular strain can cause, where in the network its effects will be most severe, and what type of solutions can be devised to detect and deflect it. We have identified three basic attack characteristics: (1) attack source; (2) SA characteristics; and (3) SA generation rate. Details about these characteristics, their importance, and their implications are described below:

Attack Source. Information about the attack source is important not only for identifying malicious host(s), but also for understanding how long the attack might last. This is because the number of external MSDP peers for a network and its proximity to the core of the MSDP flooding topology affect how severe an attack will be and how quickly it will spread. Good connectivity means an attack will spread more quickly and create a larger flood of attack traffic. Therefore, the location of the attack source will decide the time the attack will take to propagate across the infrastructure and how long bogus SAs will last in the infrastructure. Attacks can be centralized or distributed. Distributed attacks are particularly malicious because they are much harder to detect and at the same time can generate many more bogus SAs.

SA Characteristics. SA messages are defined by three fields. These are:

1. *Group Address*: Class-D, multicast group address that the original multicast data packet was sent to.
2. *Source Address*: IP address of the source that sent the packet.
3. *RP address*: IP address of the originating RP that generated the SA message.

Of these three fields, the group address is easiest for an end host to manipulate because it simply involves changing the destination (Class-D) address. The next easiest is the source address as it can be manipulated through

simple IP address spoofing though, SA messages will only be generated for those source addresses that belong to the RP's domain. Finally, because of the MSDP Peer Reverse-Path-Forwarding(RPF) rules [4], the RP address of an SA is not something an end host can easily change. The only way to generate SAs with different RP addresses is through a distributed attack where hosts from multiple domains participate.

SA Generation Rate. Damage by an MSDP-based DoS attack is directly proportional to the size of the SA flood that it generates. Therefore, the most damaging MSDP DoS attacks are ones where a large of number of SAs are generated and propagated in a very short period of time. However, if attacks cause a rapid surge in the number of SA messages, they are also easier to detect. A more virulent attack could counter simple detection schemes by increasing the number of SAs in the infrastructure at a slower rate. Although this will eventually lead to the same number of bogus SAs being propagated, such an attack would be difficult to detect.

4 Detecting and Deflecting MSDP-Based DoS Attacks

A solution to counter an MSDP-based DoS attack essentially consists of two processes: detection and then deflection. While detection is the process of discovering if an attack is underway and identifying bogus SAs, deflection is the process for annulling the effects of an attacks by keeping bogus SA messages from being processed or propagated. Two metrics are available to evaluate quality of a detection and deflection solution: *latency* and *effectiveness.* Latency is the time that a scheme takes to detect an attack after it has started. Effectiveness of a solution is the effectiveness of the SA characterization scheme, i.e. the ability to differentiate legitimate (good) SAs from bogus (bad) SAs. An effective solution should eliminate all bogus SAs and leave all legitimate SAs unaffected. Developing such effective detection and deflection solutions is the focus of this section. In later sections we evaluate these schemes on the basis of the latency and effectiveness metrics.

Since detecting an attack is the first step in any solution, devising effective detection schemes is critical. In general, detecting an MSDP-based attack involves identifying an anomalous increase in SA traffic by comparing it to certain thresholds. Different types of detection schemes can be devised by varying how these thresholds are chosen. Two basic options are available: *static thresholds* and *adaptive thresholds.* While the former rely on predetermined values, adaptive thresholds rely on values that are automatically adjusted to take recent traffic trends into account. Once an attack is detected, deflecting the attack requires the ability to differentiate good SAs from bad.

In the remainder of this section, we discuss solutions that provide both detection and deflection. For this discussion we classify solutions in two categories: (1) solutions based on the provisions available through existing protocol implementations in routers; and (2) new solutions developed specifically for countering MSDP-based DoS attacks that can be deployed either in an external device or in the router itself.

4.1 Solutions Based on Existing Protocol Capabilities

Existing implementations of MSDP provide configuration options which can be used to alter how MSDP routers accept, store, and propagate SAs. Among these options, three are particularly useful for detecting and deflecting MSDP-based DoS attacks: (1) SA rate limiting, (2) SA cache size limiting, and (3) SA filtering. The operation of these three options are described below.

SA Rate Limiting. Rate limiting is a technique employed by a router to limit the number of SAs it accepts per unit of time on any of its interfaces. Two such limits can be used as solutions against MSDP-based DoS attacks: (1) *register message rate limiting*; and (2) *SA rate limiting. Register message rate limiting* restricts the number of register messages that an RP processes. As register messages originate from the first hop router when a host sends the initial data packet (Figure 1), this limit can be used to reduce the effects of a DoS attack originating from within the RP's domain. *SA rate limiting* allows the number of SAs that an RP receives from its MSDP peers to be limited. Therefore, this limit can be used to reduce the effects of DoS attacks originating in external domains.

SA Cache Size Limiting. MSDP routers cache the most recently received SAs. The goal is to reduce join latency for new receivers by maintaining information about active sources. In the event of a DoS attack, the cache size of an MSDP router can grow tremendously because of the SA flood. This can have an adverse effect on the performance of the router. Not only will memory requirements increase, processing overhead required for cache lookups will increase significantly as well. Furthermore, as bogus SAs might be stored in the cache for a period much longer than the actual attack itself, the effect of the attack on the router's performance can last well beyond the attack. Routers, therefore, allow limits to be set on the size of the SA cache.

SA Filtering. SA filtering is a mechanism that uses Access Control Lists (ACLs) to implement admission control for SAs at an MSDP peer. Filters are typically configured manually and can be used to block SAs on the basis of their source and group address prefixes as well as their origination RP. All messages from a peer can be filtered as well. Although filtering can explicitly control the flow of SAs, it is not a practical solution against DoS attacks. As most DoS attacks do not last very long and the characteristics of the attack vary, manually configuring a filter to counter an attack may not be feasible.

The advantage of solutions based on existing protocol capabilities is that they are simple to use and easy to deploy. However, their effectiveness in countering MSDP DoS attacks is limited. This is primarily because they rely on static thresholds. Their effectiveness is marginalized further because the limits for these thresholds usually need to be configured manually. Configurations are, therefore, tightly bound to the “normal” traffic conditions. If traffic conditions change and the number of legitimate SAs increases because of some legitimate external event, static thresholds can actually do damage to normal, correct operation. Another drawback of these solutions is that

they do not take the characteristics of the SAs into account. Therefore, once an attack starts, good and bad SAs are dropped indiscriminately. So while memory and processing overload are controlled, denial of service is not prevented.

4.2 Proposed New Solutions

Because existing protocol features are not effective at handling DoS attacks, new mechanisms need to be developed that employ more powerful detection and deflection mechanisms and that are effective even against sophisticated attacks. We propose the following four new solutions in this section: (1) adaptive limits on the SA arrival rate; (2) limiting the SA generation rate of RPs; (3) limiting the SA generation rate of sources; and (4) signature detection.

The solutions that we propose employ simple, yet, powerful detection schemes. As we will show later in this paper, they exhibit low latency and high effectiveness against most types of attacks. Although any security solution is bound to have at least some resource requirements, the simplicity of our schemes reduces their resource requirements and, therefore, makes them more suitable for securing high traffic network locations close to the core of the infrastructure. While low latency and high effectiveness are the two primary goals behind our solutions, our solutions also have some basic learning capabilities and can adapt themselves to trends in SA traffic. To this end, our solutions, except for the signature-based one, rely on defining thresholds that can automatically adjust to changing traffic trends. In the remainder of the section we describe the operation of these solutions followed by a brief discussion of their advantages and their limitations. First, we describe the scheme we use for generating adaptive thresholds.

Adaptive thresholds are generated using dynamic prediction of normal traffic patterns based on double exponential smoothing. Double exponential smoothing predicts threshold values with the help of historical values. A smoothing constant, α, decides weights assigned to the historical values. A second constant, γ helps to accommodate changing trends in the number of SAs. The resultant predictions act as dynamic thresholds such that when the frequency of certain characteristics of the SAs exceed the threshold, an attack is detected. Another factor, δ, allows for variations in the predicted values to protect excess legitimate traffic from being flagged as an attack. The value of δ will also determine how quickly an attack is detected. If the variation is kept small, then an attack will be predicted almost immediately. However, a small δ can result in numerous false positives. If more variation is allowed, the number of false positives will be reduced, however, the time to detect an attack will be longer. We discuss the implications of these choices in δ in greater detail during the evaluation of schemes later in this paper.

Adaptive Limits on SA Arrival Rate. This solution relies on applying an adaptive threshold to the SA arrival rate. This is the simplest of the proposed solutions, and an intuitive improvement on the static rate limiting options available in the routers.

Limiting SA Generation Rate of RPs and Sources. The goal behind applying adaptive thresholds to the SA generation rate is to identify RP(s) and source(s) that may have been compromised. This, in turn, can help in identifying a compromised device and characterizing SAs originating from it as bogus. Identification of a compromised RP relies on predicting the number of SAs expected from each RP. If the number of SAs advertised by an RP at any given time exceeds the predicted value, it is believed to be malicious. Similarly, malicious *sources* can be identified by predicting the expected group membership for every source. Once a compromised RP and/or source has been identified, all of its SAs are characterized as bogus and dropped.

Signature Detection. Previous attacks against multicast can be analyzed and signatures for those attacks can be developed. For example, the signature of an attack could be the distinctive trend of group addresses associated with its SAs. The Ramen worm uses a contiguous range of Class-D addresses. Sapphire attacks use a predetermined random number generation algorithm to chose group addresses. Therefore, if the signature of an attack is known, it can be detected quickly and accurately.

Although the proposed solutions can be very effective in detecting and deflecting attacks, computational resources for some of the schemes are quite high. This is because applying adaptive thresholds on a per-RP or per-source basis will require predicting thresholds for *every* RP and source respectively. While the prediction process itself may have high processing overhead, maintaining state for the predicted values can also increase memory requirements. Because the number of RPs and sources in the infrastructure has been consistently increasing [7], scalability becomes an issue. Another factor that can limit the usability of these solutions is that they can be difficult to deploy. Deploying them on devices other than the router would require installing extra hardware. On the other hand, deploying these solutions as part of the router would eliminate the need for additional devices, but would require implementing additional functionality. In spite of these limitations, it is important to note that these solutions are more effective in detecting and deflecting MSDP-based DoS attacks than those based on the existing protocol capabilities. We have just presented a subjective analysis in support of these solutions. In Section 6, we provide a detailed quantitative analysis.

5 Evaluation Setup

Our main evaluation goal is to assess the effectiveness of various solutions discussed above against different strains of MSDP DoS attacks. To this end, we have devised a three step evaluation process:

1. Generate SA workloads for attacks.
2. Select a set of detection and deflection solutions by choosing from among the schemes discussed above.
3. Construct a simulator to analyze the effectiveness of the schemes selected in Step 2.

Next, we describe the evaluation setup by elaborating on these three steps. We then analyze simulation results in the next section.

SA Workloads. An SA workload is a chronological list of all the SA messages that an MSDP peer receives within a specific time period. Our goal behind creating these workloads is twofold. First, we want these workloads to be representative of the actual SA traffic that an attack would generate. Second, we want our workloads to represent SA traffic from the point-of-view of a general network location that is at a generic distance from the source of the attack. While the motivation behind the first goal is clear, the second is important because traffic patterns seen by the router in a compromised host's domain is considerably different from anywhere else and is usually easy to characterize as malicious. Therefore, in order to devise a solution that is independent of the attack origination and that is effective at any generic location in the network, workloads should be generic and should take MSDP propagation delays into account.

Workloads used in this paper are based on actual MSDP SA data collected from core multicast routers using our global monitoring infrastructure, Mantra[5, 6, 7]. Mantra collects data at the network layer from multiple multicast routers by capturing their internal memory tables. For MSDP, it relies on state tables that routers use to cache the SAs they receive. Workloads presented in this paper are based on tables collected from one of the more important multicast exchange points on the West Coast of the United States called Federal IntereXchange–West (FIXW). It is important to note that because Mantra collects data from a router's state tables only periodically, the data that is collected is not an exact chronological representation of SA traffic seen by the router. Although SA workloads are not a direct representation of Mantra data, enough information is available in the state tables to overcome any limitations. We use the state tables and our knowledge about MSDP-based attacks to create representative workloads. For example, we account for SA arrival time with the help of lifetime information associated with each SA.

We construct three workloads for use in this paper: (1) a Sapphire worm workload; (2) a hybrid Ramen worm workload with a slow SA generation rate; and (3) a legitimate-surge workload. We decided not to choose the Ramen worm as a workload because the Sapphire worm is more sophisticated. The first two workloads represent different strains of attacks, while the last one represents a legitimate increase in multicast traffic cause by an external event. For each of these workloads, the number of legitimate SAs during normal operation is in the range of 3,400 to 3,900. Attack workloads vary in terms of the rate of increase of SAs as well as in terms of SA characteristics. The characteristics of each workload are described below:

- *Workload 1 - Sapphire Worm*: For this workload, we used our Mantra logs to identify the RPs that generated bogus SAs during the attack period. We then used an interpolation technique to estimate the number of SAs advertised per minute by each RP. The estimated number of SAs for each RP were then made to originate from sources that were not active prior to the attack. We then added the total number of SA messages advertised by the RPs to create the Sapphire worm workload. Figure 5 shows the characteristics of this workload. This workload is 270 minutes long with the attack lasting 170 minutes. The workload contains about 3,600 good SAs that are re-advertised every minute. During the attack, the total number of SAs ranges from 30,000-35,000.
- *Workload 2 - Hybrid Ramen Worm with Slow SA Generation Rate*: The SA characteristics of this workload are the same as that of the Ramen worm, i.e. the addresses for the source and RP remain the same while

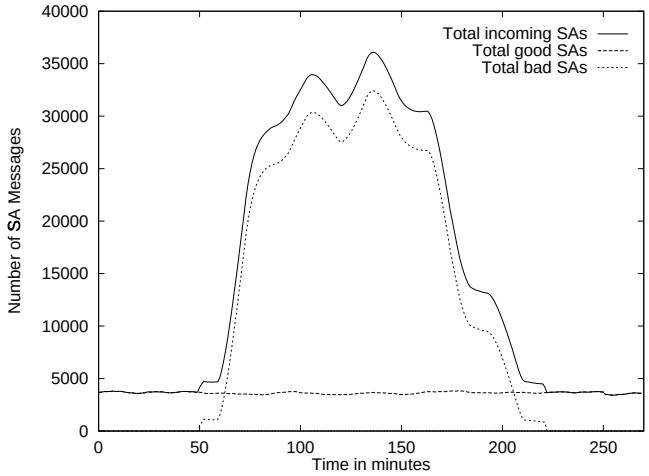

Figure 5: Workload 1 - Sapphire worm.

the group address varies. Figure 6 shows the characteristics of this workload. The workload has about 3,600 good SAs per minute. The peak number of SAs during the attack is only about 7,000. The attack represented by this workload lasts for about 4.5 hours and the rate of bogus SA generation is only 25 SAs per minute. The peak is, therefore, achieved by re-advertising all the bogus SAs from the previous cycle along with new ones. The number of SA messages advertised by this workload is substantially smaller than the SAs advertised by Ramen. This has been done intentionally to introduce stealth into the attack. The number of SA messages is low enough to possibly pass as an increase because of legitimate use. Although the SA messages are low in number, the effect of this attack can still be quite damaging because of the cache buildup of the bogus SAs.

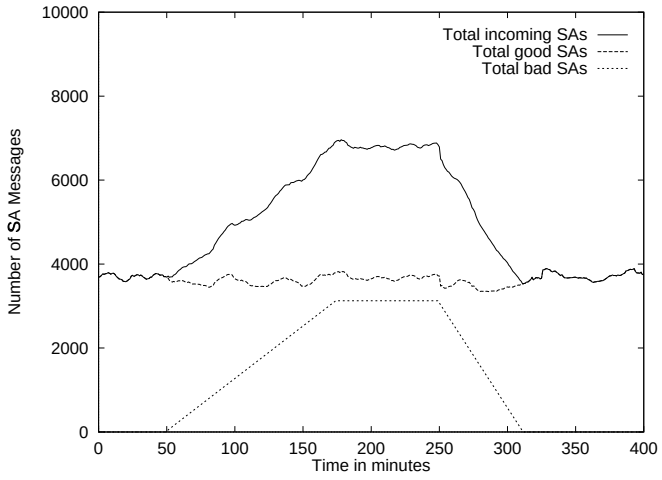

Figure 6: Workload 2 - Hybrid Ramen worm with slow SA generation rate.

- *Workload 3 - Legitimate Surge*: This workload is based on legitimate SA traffic noticed after the "9/11" incident. There was a significant increase in the number of SAs as receivers joined certain multicast groups to receive news feeds from sources like *CNN.com*. Evaluating our solutions against this workload is important because detection and deflection schemes should not generate false positives for this workload. Figure 7 shows the characteristics of this workload. The workload has peaks of about 5,000 SAs. The increase in SAs are because of new sources in the network. These are not sources sending actual traffic but are receivers

sending RTCP-style feedback to the groups.

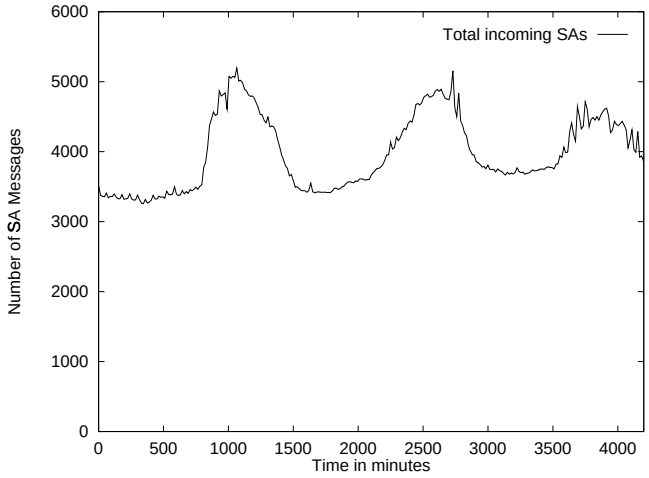

Figure 7: Workload 3 - Legitimate Surge.

Choosing Schemes. In choosing the schemes to evaluate from the ones presented in Section 4, we use two guidelines. First, the schemes selected should range in terms of sophistication in detection and deflection from simple to more complex. By choosing schemes in this range, we will be able to better compare different schemes and also use the comparisons as a basis to discuss the tradeoffs involved in choosing a scheme. Second, the schemes chosen should include ones currently available in real world deployments. This gives us the ability to compare our proposed schemes with ones that already exist. Therefore, we chose to evaluate the following schemes:

- *Scheme 1 - Static Rate-Limiting*: In static rate-limiting, a static limit is set on the over all number of SAs expected at any instant of time. If the number of SAs arriving is more than the static limit, the excess number of SAs are dropped.
- *Scheme 2 - Adaptive Threshold on the SA Arrival rate*: In this scheme, an adaptive threshold is used to predict the total number of SAs expected every minute. If the actual number of SAs arriving exceeds the threshold, the excess number of SAs are dropped.
- *Scheme 3 - Adaptive Threshold on the SA Generation rate of Sources*: In this scheme, adaptive thresholds are determined on a per source basis. If a source advertises more than the SA threshold set for it, that source is considered malicious and all SAs from the malicious source are dropped.

Scheme 1 is the simplest of the schemes we have chosen to evaluate. Also, scheme 1 is in use in real world deployments. Of the remaining two schemes, both are schemes we have proposed for use in attack detection and deflection. Scheme 3 is the more sophisticated and the more complex of the two. This is because, adaptive thresholds have to be maintained for *each* source. Maintaining such thresholds requires higher processing and state maintenance than the first two schemes. We have chosen scheme 3 as a per-source solution over a per-RP

solution because the per-source solution give us a better notion of the upper efficiency bounds achievable in terms of detecting and deflecting attacks.

Applying the Schemes. We tested our three schemes against our three workloads using the simulation framework illustrated in Figure 8. The figure shows four modules: *Parser*, *Predictor*, *Detection Module*, and *Deflection Module*. All are implemented as Perl scripts.

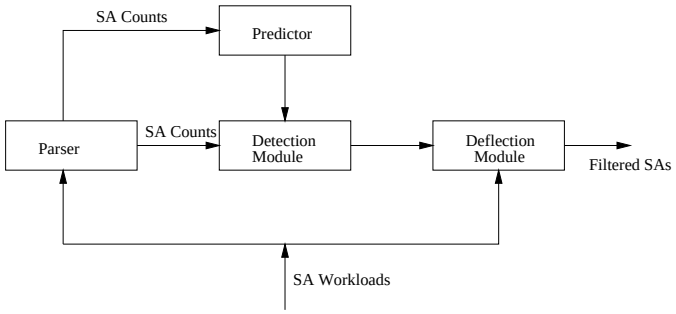

Figure 8: Simulation framework for evaluation of schemes.

The parser handles our SA workloads and updates two counters: 1) total number of SAs, and 2) SAs from each source. The counter values are passed to the predictor and the detection module. The predictor uses the counter values to predict the expected total number of SAs for the next time interval (one minute). In the case of scheme 1, the predictor simply uses the static value of 7,000 SAs. For scheme 2, the predictor dynamically determines the threshold for the expected number of SAs. For scheme 3, the predictor determines the thresholds for the expected number of SAs from every source. For doing the predictions, the predictor uses the double exponential smoothing algorithm explained in Section 4. It uses the values of 0.1 and 0.01 for α and γ respectively, which we found to be appropriate values for SA rate prediction. For δ, the value 400 and 50 are used for scheme 2 and scheme 3 respectively. δ is high enough to allow for enough legitimate increases in the total number of SAs and the number of SAs from a source. The detection module then compares the predicted value with the actual SA count. If the actual count is more than the predicted value, the deflection module is invoked to drop SAs from the SA workloads. The deflection module uses the dropping policies discussed above for the three schemes to filter SAs from the workloads.

6 Evaluation

In this section, we evaluate the three attack detection and deflection schemes discussed in Section 5 against our evaluation metrics: latency and effectiveness. Latency in detection indicates when an attack is detected after it has started. Effectiveness is used to evaluate the percentage of good SAs not dropped by a scheme and the percentage of bad SAs dropped by the scheme.

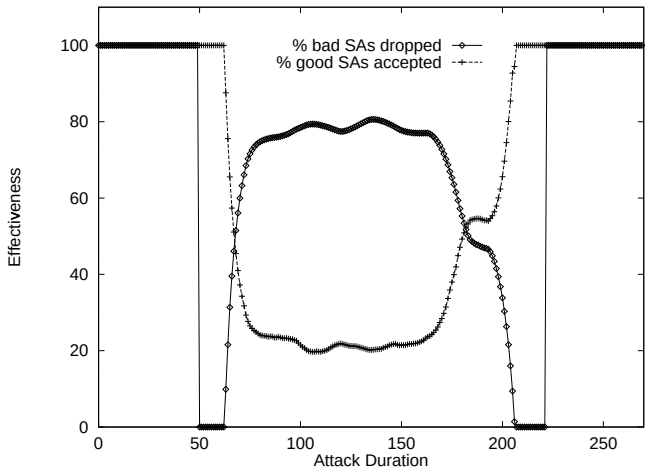

Figure 9: Effectiveness for scheme 1 against the Sapphire worm.

6.1 Results

Figure 9 shows the effectiveness of scheme 1 against the Sapphire worm (workload 1). The X-axis of the figure plots time in minutes and the Y-axis displays the effectiveness value. The figure shows two separate lines that show the percentage of bad SAs dropped by the scheme, and the percentage of good SAs not dropped by the scheme. Although the attack starts at 50 minutes, it is not detected until 63 minutes into the workload. This is because scheme 1 uses a static threshold value of 7,000 to detect attacks. Between time 50-63, the number of SAs in the network including the bad ones is less than 7,000. At time equal to 63, the static threshold is crossed and scheme 1 detects the attack.

Once the attack is detected, it begins to drop excess SAs over the 7,000 limit. The static threshold allows for a certain number of bad SAs to be accepted. During the beginning of the attack, the number of bad SAs is small. Therefore, a majority of the bad SAs are accepted. This is why, during the beginning of the attack, the percentage of bad SAs dropped is low. As the attack progresses, the number of bad SAs in the network increases. Consequently, the number of bad SAs accepted is overshadowed by the number of bad SAs dropped. Therefore the percentage of bad SAs dropped increases during the middle of the attack. A similar reasoning can be applied to the drop in the percentage of good SAs accepted. At the beginning of the attack, the number of bad SAs in the network is small. Consequently, a higher number of good SAs are accepted. As the number of bad SAs increases, only a small percentage of good SAs are accepted and a majority of them are dropped. This explains the drop in the percentage of good SAs accepted. The reverse of the reasonings explains why at the end of the attack, the percentage of bad SAs dropped decreases and the percentage of good SAs accepted increases.

The static threshold used by scheme 1 is set such that in the case of an MSDP attack, the multicast infrastructure is capable of handling the controlled number of allowed SAs without much damage. However, scheme 1 allows a high number of bad SAs and only a low number of good SAs to be accepted. This likely results in severe loss of

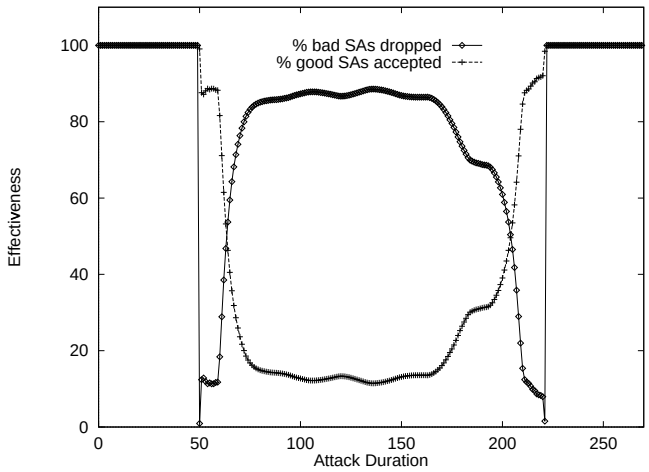

Figure 10: Effectiveness of scheme 2 against the Sapphire worm.

service as group members do not hear that sources have become active. Moreover, scheme 1 cannot accommodate increases in SA traffic above its static threshold because of legitimate use. As an example of increased legitimate SA traffic consider the legitimate surge (workload 3) shown in Figure 7. In this workload, the number of SAs in the network rises from the usual value of about 3,500 SAs to peaks of almost 5,000 SAs. If the static threshold for scheme 1 was set at 4,500 for instance, scheme 1 would result in false positives. To accommodate changing trends as noticed in the workload, scheme 2 relies on dynamic thresholds. Before discussing how scheme 2 performed against the legitimate surge, we first present scheme 2's evaluation against the Sapphire worm.

The effectiveness of scheme 2 against the Sapphire worm is shown in Figure 10. The shape of the plot is similar to the one shown for scheme 1 in Figure 9. However, the latency involved in attack detection by scheme 2 is less than one minute where scheme 1's detection latency is about 13 minutes. Also interesting is that scheme 2 achieves a slightly higher percentage in dropped bad SAs as compared to scheme 1. However, it attains a lower percentage of accepted good SAs. This is because scheme 2's dynamic threshold value is smaller than scheme 1's static threshold. Consequently, a higher percentage of bad SAs are rejected and a lower percentage of good SAs are accepted. Scheme 1 and scheme 2 together can detect attacks but cannot distinguish good from bad.

As discussed before, scheme 1 could falsely detect the legitimate surge as an attack. Next, we present our results from scheme 2's evaluation against the legitimate surge. Our goal is to show that while scheme 1 falsely detects this workload as an attack, scheme 2 does not. Figure 11 shows the results from our evaluation. The figure plots the duration of the workload on the X-axis. The Y-axis plots the number of SAs. Two lines are displayed, one for the workload itself and the other for the predicted number of SAs per minute expected by scheme 2. It can be seen from the figure that these two lines do not intersect. This implies that the predicted number of SAs as determined by scheme 2 is always more than the number of SAs in the network. As the predicted number is more than the actual number of SAs, scheme 2 does not believe the workload is an attack. Therefore, scheme 2 is successful in avoiding classifying the legitimate surge as a false positive.

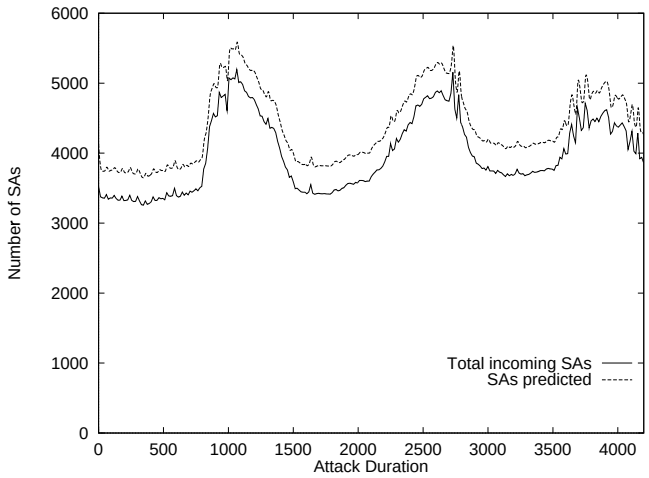

Figure 11: Acceptance of the legitimate surge by scheme 2.

From the above results, scheme 2 performs better than scheme 1 in three ways: 1) scheme 2 has a latency value of less than 1 minute whereas scheme 1's latency is about 13 minutes, 2) scheme 2 drops a higher number of bad SAs as compared to scheme 1 albeit accompanied by the dropping of a higher number of good SAs, and 3) scheme 2 can accommodate for the rise and fall in the number of SAs in its predictions. Therefore, scheme 2 can better handle cases such as the legitimate surge. However, as we show next, the feature in scheme 2 that allows it to accommodate for SA trends works against it when it is used against slow attacks such as the slow Ramen worm (workload 2).

Figure 12 shows the number of SAs in the network to be always less than the threshold predicted by scheme 2. Therefore scheme 2 will never detect the attack. The reason for this is twofold. First, the value δ used in the schemes allows for an increase in the number of SAs in case of increased legitimate use. However, in this case δ allows for an increase in the number of SAs due to bad SAs. The second reason is that the predictors interpret the increase in SAs allowed by δ as a rising trend in SAs.

To detect attacks such as ones caused by the slow Ramen worm, per-message evaluation of SA traffic is required to detect malicious behavior. Such evaluation can be performed by schemes that track either RP or source behavior. Next, we look at how scheme 3 performs against the slow Ramen worm. We have validated that scheme 3 detects the Sapphire worm as an attack and also that it does not falsely detect the legitimate surge as an attack. In the case of the Sapphire worm, scheme 3 detects the attack within one minute. Also, scheme 3 achieves 100% drop of bad SAs and 100% acceptance of good SAs. This is because, for the Sapphire worm, all the sources that caused bogus SAs to be generated were hosts that were not active participants before the attack took place. As they were not active participants these hosts never generated any good SAs. As a result all the SAs generated by these hosts were bad SAs and are therefore dropped. Note that even in the 1 minute interval when the attack is detected, none of the bogus SAs are accepted for transmission to downstream MSDP peers. This is because MSDP relies on the periodic retransmission of MSDP SA messages. Therefore, all incoming SA messages are first buffered in the SA

cache of the router and are then retransmitted only at the start of the next retransmission period. This wait-time experienced by all SAs before retransmission allows scheme 3 to drop all bogus SAs immediately after detection. Therefore, scheme 3 achieves 100% values for both effectiveness metrics. With the legitimate surge, the increase in the number of SAs were for new sources that joined the well known multicast groups streaming traffic from sources like CNN.com. Each new source generated a small number of SAs, less than the threshold allowed by scheme 3. Therefore scheme 3 does not detect the legitimate surge as an attack.

Figure 13 shows the effectiveness of scheme 3 in detecting the slow Ramen worm. The attack starts at time equal to 50 minutes. The latency involved in detecting the attack is less than 1 minute. As soon as the attack is detected, the malicious source identified by scheme 3 is blocked. As a consequence, none of the bad SAs advertised by the malicious source are accepted. This is why the percentage of bad SAs dropped is 100% throughout. The source identified as malicious also advertises 122 good SAs. These are also dropped because scheme 3's SA drop policy says that when a malicious source is detected, *all* SAs from the source are dropped. Therefore, the percentage of good SAs accepted by scheme 3 reduces to about 96%. If the malicious source never actually transmitted any good SAs, the percentage of good SAs accepted would have been 100%.

6.2 Discussion

We have made a number of observations based on the evaluation results presented above. Below, we organize our observations into two categories: observations related to attack detection and observations related to attack deflection:

Attack Detection. Scheme 1 relies on a static threshold for detecting attacks. If the static threshold is set too low, there is the possibility that a legitimate surge can result in a false positive. On the other hand, if the threshold is set too high, it is possible for subtle attacks like the slow Ramen worm to go undetected. The threshold value should be set based on monitoring SA traffic and determining a value that is not too high to let subtle attacks through and not too low to generate false positives for legitimate usage. Static thresholds, however, are inherently susceptible to intelligent attacks. Detecting attacks just using static thresholds is, therefore, quite prone to error. The advantage of static thresholds is that they can control and therefore limit the number of SAs being exchanged between MSDP peers. Moreover, it is easy to implement and already deployed in the Internet.

Scheme 2 and scheme 3 rely on dynamic thresholds for predicting SAs expected every minute. We saw that scheme 2 fails to detect the slow Ramen attack because its δ value allowed for the increase in the number of bogus SAs to affect the threshold value. As a result, the prediction scheme interpreted the increase as a valid change in trend and the attack went undetected. If the δ value were set to such a low value that the slow attack would be detected, the threshold might not be dynamic enough, and legitimate increases, as seen for traffic surge noticed after the “9/11” incident, might be falsely identified as an attack. Scheme 3, on the other hand, was able to detect the slow Ramen attack because it uses a per-source analysis of the received SAs. However, the resource requirements for scheme 3 are quite high because it needs to calculate and keep track of thresholds for each and

every source. To reduce the resource requirements of scheme 3, one possible simplification is to predict *one* threshold for all sources instead of per-source thresholds. Another simplification Would be to predict thresholds based on source prefixes. It is important to note, however, that although resource requirements for such modified schemes are considerably lower, these modifications may be detrimental to characterization accuracy.

Although scheme 3 is effective against the slow Ramen worm, the same workload but with distributed SA generation can always trick it into not detecting the attack. Although such a distributed attack is difficult to orchestrate, it is important to know that even scheme 3 is not perfect in its attack detection capability.

Based on the arguments above, we believe that a "sieve" approach could be adopted as an attack detection solution. In the "sieve" approach, several of the schemes discussed in Section 4 could be applied simultaneously. For example, one approach could be to use dynamic thresholds in addition to setting a static threshold. In the case of the slow Ramen attack, the dynamic threshold might fail to detect the attack, but at least we are guaranteed that the static threshold would detect the same attack and subsequently control the flow of SAs in the network.

Attack Deflection. The effectiveness of scheme 1 depends entirely on the static threshold. With the dynamic threshold schemes, the threshold can be set high or low by changing δ. If the threshold for any of these schemes is high, a high percentage of good SAs will be accepted, and a low percentage of bad SAs will be dropped. On the other hand, if the static threshold is low, a low percentage of good SAs will be accepted, and a high percentage of bad SAs will be dropped. Another observation is that the dropping policy significantly affects the effectiveness of a scheme. Scheme 1 and scheme 2 use a naive dropping policy. The dropping policy drops *all* SAs above their threshold. Scheme 3 blocks all SAs from the source identified as malicious. The effectiveness of the schemes could be further improved, however, with the help of intelligent dropping policies. For example, if "Last In First Out" (LIFO) policy is used, the latest entries in the SA cache would be dropped. This will increase the probability that the older good SAs in the cache are maintained and the newer bad SAs are dropped.

7 Conclusions

Multicast is vulnerable to various types of denial of service attacks. In this paper, we have looked at one class of DoS attacks related to the Multicast Source Discovery Protocol (MSDP). MSDP attacks are extremely easy to launch and have widespread repercussions that can have crippling effects on the multicast infrastructure. DoS attacks caused by the Ramen and Sapphire worms are specific instances of attacks that have already exploited the weaknesses in MSDP. As multicast achieves ubiquitous deployment and is widely used in the future, attacks specifically targeted against MSDP are likely to become common-place. Effective attack detection and deflection solutions are, therefore, needed.

Solutions currently exist to prevent MSDP DoS attacks. These solutions rely primarily on rate limiting as a measure to limit SA storms. We showed in this paper that such solutions are not effective in preventing attacks for two main reasons. First, such solutions rely on static thresholds that need manual changes as SA traffic profiles

change. Second, these solutions resort to indiscriminate dropping of SAs without prior SA characterization.

To overcome problems with existing solutions, we have proposed several schemes that have a varied range of sophistication. The main principle used by these schemes is that they rely on adaptive thresholds to learn from MSDP usage trends and automatically adapt to changing traffic patterns. In addition, once an attack is detected, these schemes rely on SA characterization to selectively drop bogus SAs. In choosing which one of these schemes to deploy, performance and effectiveness tradeoffs need to be considered. While the simpler schemes require only low processing capabilities, they are, however, less effective as compared to the more complex schemes. The complex schemes can be very effective in preventing the attacks. However, they require considerably high processing capabilities in addition to higher state maintenance.

In this paper, we looked only at classes of solutions that can be applied without modifying the existing MSDP protocol specification. MSDP is vulnerable to DoS attacks because it relies on a flooding mechanism to exchange SAs. Additionally, a majority of these source advertisements are useless to many domains that receive them because they generally lack receivers that are interested in those sources. This leads us to believe that MSDP in fact could be modified to use a selective SA forwarding policy instead of the flooding model it uses today. Modifying MSDP to make it more secure and scalable is important for the stability of multicast in the Internet. Use of MSDP will continue until alternate multicast routing protocols that do not need MSDP are deployed. As this does not seem likely for many years, securing MSDP needs to be addressed, sooner rather than later. Finally, while the security concerns and the solutions we have discussed are specific to multicast, the lessons learned can be applied in solving problems in other evolving network services.

Acknowledgments

The authors gratefully acknowledge Anshuman Kanwar and Vishal Mittal for their support and feedback on the paper.

References

[1] K. Sarac and K. Almeroth, "Supporting multicast deployment efforts: A survey of tools for multicast monitoring," *Journal of High Speed Networks–Special Issue on QoS for Multimedia on the Internet*, vol. 9, no. 3,4, pp. 191–211, March 2000.

[2] T. Hardjono and G. Tsudik, "IP multicast security: Issues and directions," *Annales de Telecom*, 2000.

[3] P. Rajvaidya and K. Almeroth, "Analysis of routing characteristics in the multicast infrastructure," in *IEEE Infocom*, San Fransisco, California, USA, April 2003.

[4] B. Fenner and D. Meyer, "Multicast source discovery protocol (MSDP)," Internet Engineering Task Force (IETF), draft-ietf-msdp-spec-*.txt, June 2003.

[5] P. Rajvaidya and K. Almeroth, "A router-based technique for monitoring the next-generation of internet multicast protocols," in *International Conference on Parallel Processing (ICPP)*, Valencia, Spain, September 2001.

[6] P. Rajvaidya and K. Almeroth, "A scalable architecture for monitoring and visualizing multicast statistics," in *IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM)*, Austin, Texas, USA, June 2000.

[7] P. Rajvaidya and K. Almeroth, "Building the case for distributed global multicast monitoring," in *Multimedia Computing and Networking (MMCN)*, San Jose, California, USA, January 2002.

[8] D.E. Denning, "An intrusion-detection model.," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, February 1987.

[9] H. Holbrook and B. Cain, "Source-specific multicast for IP," Internet Engineering Task Force (IETF), draft-ietf-ssm-arch-*.txt, November 2003.

[10] S. Bhattacharyya, "An overview of source-specific multicast (SSM) deployment," Internet Engineering Task Force (IETF), RFC 3569, July 2003.

[11] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, January/February 2000.

[12] W. Fenner, "Internet group management protocol, version 2," Internet Engineering Task Force (IETF), RFC 2236, November 1997.

[13] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, G. Liu, and L. Wei, "PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, pp. 153–162, Apr 1996.

[14] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol extensions for BGP-4," Internet Engineering Task Force (IETF), RFC 2283, February 1998.

[15] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using boolean function minimization techniques," in *IEEE Infocomm*, New York, New York, USA, March 1999, pp. 689–698.

[16] C.K. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," in *ACM Sigcomm*, Vancouver, CANADA, August 1998, pp. 68–79.

[17] H. Chang, S. Wu, and Y. Jou, "Real-time protocol analysis for detecting link-state routing protocol attacks," *ACM Transactions on Information and System Security*, February 2001.

[18] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

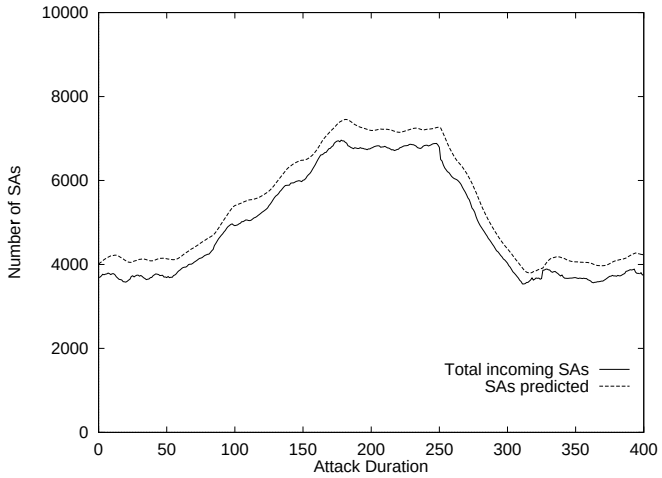

Figure 12: Failure of scheme 2 in detecting the slow Ramen worm.

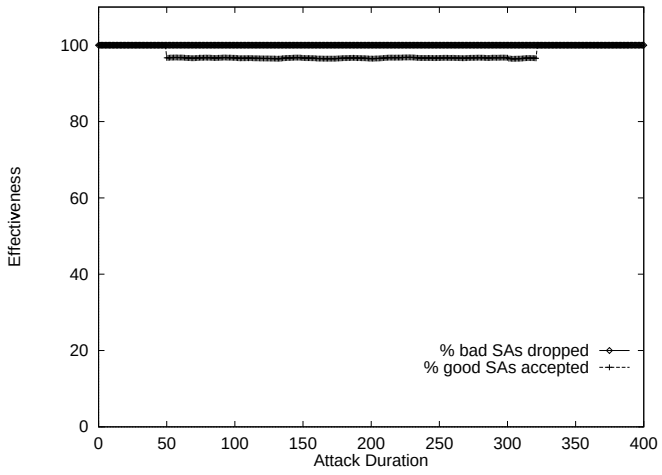

Figure 13: Effectiveness of scheme 3 against the slow Ramen worm.