View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarlyCommons@Penn

University of Pennsylvania
ScholarlyCommons

Technical Reports (CIS)

Department of Computer & Information Science

1-1-2013

Network Intrusion Detection and Mitigation Against Denial of Service Attack

Dong Lin
University of Pennsylvania, lindong@cis.upenn.edu

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Part of the Computer Engineering Commons

Recommended Citation

Dong Lin, "Network Intrusion Detection and Mitigation Against Denial of Service Attack", . January 2013.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-13-04.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/981
For more information, please contact repository@pobox.upenn.edu.

Network Intrusion Detection and Mitigation Against Denial of Service Attack

Abstract

The growing use of Internet service in the past few years have facilitated an increase in the denial of service (DoS) attacks. Despite the best preventative measures, DoS attacks have been successfully carried out against high-prole organizations and enterprises, including those that took down Chase, BOA, PNC and other major US banks in September 2009, which reveal the vulnerability of even well equipped networks. These widespread attacks have resulted in significant loss of service, money, and reputation for organizations, calling for a practical and ecient solution to DoS attack detection and mitigation.

DoS attack detection and mitigation strengthens the robustness and security of network or computer system, by monitoring system activities for suspicious behaviors or policy violations, providing forensic information about the attack, and taking defensive measures to reduce the impact on the system. In general, attacks can be detected by (1) matching observed network trac with patterns of known attacks; (2) looking for deviation of trac behavior from the established prole; and (3) training a classier from labeled dataset of attacks to classify incoming trac. Once an attack is identied, the suspicious trac can be blocked or rate limited.

In this presentation, we present a taxonomy of DoS attack detection and mitigation techniques, followed by a description of four representative systems (Snort, PHAD, MADAM, and MULTOPS). We conclude with a discussion of their pros/cons as well as challenges for future work.

Disciplines

Computer Engineering

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-13-04.

This technical report is available at ScholarlyCommons: https://repository.upenn.edu/cis_reports/981

Network Intrusion Detection and Mitigation against Denial of Service Attack

WPE-II Written Report

Dong Lin

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
lindong@cis.upenn.edu

April 15, 2013

Abstract

The growing use of Internet service in the past few years have facilitated an increase in the denial of service (DoS) attacks. Despite the best preventative measures, DoS attacks have been successfully carried out against high-profile organizations and enterprises, including those that took down Chase, BOA, PNC and other major US banks in September 2009, which reveal the vulnerability of even well equipped networks. These widespread attacks have resulted in significant loss of service, money, and reputation for organizations, calling for a practical and efficient solution to DoS attack detection and mitigation.

DoS attack detection and mitigation strengthens the robustness and security of network or computer system, by monitoring system activities for suspicious behaviors or policy violations, providing forensic information about the attack, and taking defensive measures to reduce the impact on the system. In general, attacks can be detected by (1) matching observed network traffic with patterns of known attacks; (2) looking for deviation of traffic behavior from the established profile; and (3) training a classifier from labeled dataset of attacks to classify incoming traffic. Once an attack is identified, the suspicious traffic can be blocked or rate limited.

In this presentation, we present a taxonomy of DoS attack detection and mitigation techniques, followed by a description of four representative systems (Snort, PHAD, MADAM, and MULTOPS). We conclude with a discussion of their pros/cons as well as challenges for future work.

Contents

1 Introduction

The growing use of Internet service in the past few years have facilitated an increase in the denial of service (DoS) attacks [20]. Despite the best preventative measures, DoS attacks have been successfully carried out against high-profile organizations and enterprises [1], including those that took down Chase, BOA, PNC and other major US banks in September 2009, which reveal the vulnerability of even well equipped networks. These widespread attacks have resulted in significant loss of service, money, and reputation for organizations, calling for a practical and efficient solution to DoS attack detection and mitigation.

Network intrusion detection system (NIDS) provides avenues to detect DoS attacks and other types of intrusions, and produces reports to help system administrator perform further analysis on the attacks. Some NIDSs aim to detect attacks in real-time and stop an attack in progress, whereas others focus primarily on providing after-the-fact forensic information about attacks to help repair damage, understand the attack mechanism, and reduce future attacks of the same type. It is well recognized that NIDS has become an essential part for developing robust and secure network systems against DoS attack. While its importance is recognized, designing and implementing effective NIDS remains challenging, due to the ever increasing scale and complexity of network traffic and unpredictability of unforeseen attacks.

Over the past few years, there have been extensive research activities developing effective DoS detection techniques. Various techniques have been proposed for detecting DoS and are evaluated systematically on network traffic trace collected from testbed or production networks [17, 18, 4]. In general, attacks can be detected by (1) matching network traffic against a set of rules that encodes patterns of known attacks [22, 24, 30]; (2) looking for deviation of traffic behavior from established profile [19, 4, 32, 7, 5]; and (3) training a classifier from labeled dataset of attacks to classify incoming network traffic [16, 21]. Depending on the choice of detection paradigms, these techniques differ in their detection rate, false alarm rate, adaptability to attack variations, among other performance metrics.

Once an attack is detected, attack mitigation mechanism uses the characterization of suspicious traffic to take defensive measures, e.g. packet filtering and rate limiting. To further reduce the impact on the target network, suspicious traffic can be blocked remotely at upstream router via pushback [10], or redirected to and from a cleaning center that has sufficient network and computation resource to filter out malicious traffic [2].

The rest of the paper is organized as follows. We first provide an overview of denial-of-service attacks in Section 2. Section 3 provides a taxonomy of the techniques and systems proposed for denial-of-service attack detection. We then present four representative systems to explain different detection techniques: we discuss Snort in Section 4, PHAD in Section 5, MADAM in Section 6, and MULTOPS in Section 7. Section 8 discusses two techniques for mitigating distributed denial-of-service attacks. Finally, Section 9 provides a comparison of surveyed systems and discusses challenges.

2 Overview

In this section, we first provide a taxonomy of intrusions against computer systems, followed by a detailed description of denial-of-service attacks that discusses their method, impact, detection, and recent incidence.

2.1 Intrusion against Computer Systems

Denial-of-service attack belongs to a broader class of intrusion against computer systems. Its detection techniques, in general, are also applicable to other types of intrusions. In practice, DoS attack is usually detected together with other types of intrusions in a single system, namely, network intrusion detection system. Here we provide a taxonomy of intrusions against computer systems.

- **Probe:** Probe describes activities that can systematically scan a network of machines to gather information or to find known vulnerabilities. Its use is typically followed by other more dangerous attacks because it provides a map of machines or service that have known vulnerability in a network.
- **Remote-to-local (R2L):** In R2L attack, an attacker who doesn't have user account on the target machine gains unauthorized access to that machine via remote connection. The specific examples include buffer overflow attack, network worm, and trojan program.
- **User-to-root (U2R):** U2R describes attacks where local users of the system obtain unauthorized access to confidential information or root privilege. These attacks typically exploit the weakness in the operating system or system programs running with root privilege, including those programs which are susceptible to buffer overflow attack.
- **Denial-of-service (DoS):** DoS attack is generally characterized by an explicit attempt to make a system or network resource unavailable to its legitimate user. This is typically achieved by (1) exploiting the software bug in the target system to crash it completely, or (2) sending large amounts of useless traffic towards target system to simply exhaust its computer or network resource.

2.2 Denial-of-service Attack

DoS Classification: Denial-of-service attack, as its name suggests, attempts to deny the service of target system from its legitimate users. Depending on exploits it uses, DoS can be further classified into vulnerability DoS and flood DoS.

- **Vulnerability DoS** exploits the weakness in system design, implementation or configuration to reduce system performance or crash it completely. As a typical example, a teardrop attack attempts to crash the target system by sending intentionally malformed packets which are incorrectly handled by the target machine.

- **Flood DoS** exploits the resource asymmetry between the attacker and victim, and attempts to exhaust the computation or network resource of the victim by sending huge amounts of malicious traffic at the same time.
  - **Distributed-denial-of-service attack (DDoS)** is a primary example of flood DoS, in which attackers or compromised hosts form a coordinated attack against a common target. Attacks can be launched from millions of compromised hosts around world in such a way that easily overload the target network and system.

    Although DDoS attack is easy to detect by measuring the traffic load on the target network, it is extremely hard to mitigate, for two reason: (1) it does not rely on any particular system weakness except for resource asymmetry. (2) it would be prohibitively expensive or even impossible to deploy enough computation and network resources at each enterprise network to combat large scale DDoS attack.

DoS Examples: DoS attack can be carried out in a variety of ways, each exploiting different vulnerability of target system. Here we present several representative DoS attacks to explain their method, impact, and detection.

- **Teardrop attack:** Teardrop attack is a vulnerability DoS attack. It involves sending mangled IP fragments with overlapping, over-sized payloads that can crash various operating systems which have a bug in their TCP/IP fragmentation reassembly code. Therefore, teardrop attack can be detected by looking at the fragmentation flag in the IP packet header; it can also be prevented by simply upgrading operating system.
- **Ping of death (PoD):** PoD is another vulnerability DoS attack, which involves sending an oversized ping packet to crash the target operating system. Specifically, PoD exploits the bug in operating systems that can not handle a ping packet larger than the maximum IPv4 packet size (65,535 bytes). It can therefore be detected by looking for oversized ICMP packet. Nowadays this attack is mostly historical because this bug is fixed in most systems.
- **ICMP flood:** ICMP flood is a typical example of reflector-based DoS flood attack. Instead of flooding the victim directly, the attacker sends ICMP echo requests to broadcast addresses of mis-configured networks, where each host on the network responses with an ICMP packet to the victim. As a result, the mis-configured network amplifies the flood on the victim. This attack can be prevented by filtering out ICMP packets with broadcast address.
- **SYN flood:** SYN flood is another form of DoS flood attack that attempts to exhaust connection state tables in the target system. Specifically, it exploits the TCP three-way handshake process by initiating many TCP connections but not completing them, until no new connections can be made by legitimate traffic. Nowadays even high capacity devices capable of maintaining states on millions of connections can be taken down by such attacks.

- **Bandwidth attack:** Bandwidth attack is a primary form of DDoS attack that simply overloads the target network by generating huge amounts of normal-looking network traffic, such as HTTP request, DNS request, etc. Nowadays bandwidth attacks routinely exceed 100Gbps. As a well known example, in Feb 2000, a number of high-profile websites were taken down for several hours, including Yahoo!, eBay, Amazon, CNN, and many others.

DoS Detection: DoS attacks can be detected in a variety of ways. Here we briefly discuss it in the context of vulnerability DoS and flood DoS separately.

- **Vulnerability DoS:** In general, vulnerability DoS can be detected on per-packet basis by examining the packet header values, and comparing the observed values with either patterns of known attacks or those of legitimate packets. For instance, teardrop attack can be detected by examining the fragmentation flag in the packet header, as we will show later in Figure 4.
- **Flood DoS:** Flood DoS is much straightforward to detect, since it typically involves abnormally large amounts of packets or connections that can be easily observed using simple statistical measures, e.g. number of connections made in a given time window. However, it can be extremely difficult to tell flood DoS traffic from legitimate traffic, because it can be composed of entirely normal-looking traffic that simply overloads victim's system capacity.

3 Taxonomy of Denial-of-service Detection

Here we present an overview of existing systems that performance DoS detection, and classify them according the taxonomy in Figure 1.

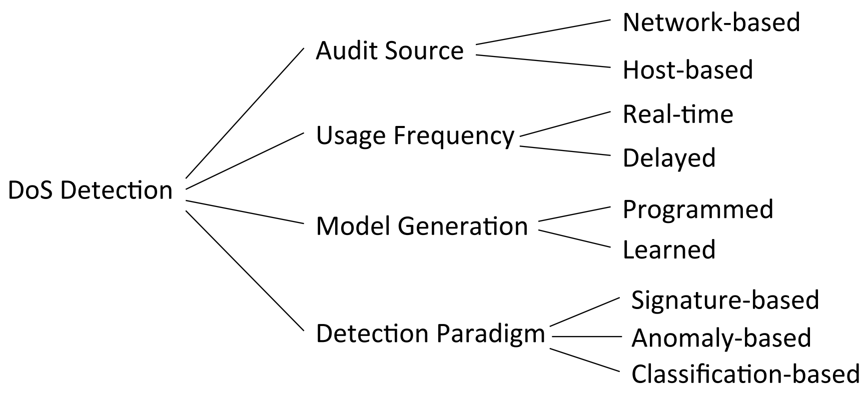

Figure 1: Classification of DoS Detection Systems

3.1 Network-based vs. Host-based Detection

Based on the source of audit data, we categorize DoS detection systems into *network-based* detection and *host-based* detection. Host-based detection employs the host operating system's audit trails as the main source of input to detect malicious activity [26, 14], whereas network-based detection builds its detection mechanism on monitored network traffic [22, 31].

Network-based and host-based detection usually differ in their deployment location and the impact on the target system. Network-based detection can be deployed at enterprise network gateway or ISP core routers. A single network-based detection system, when properly placed, can monitor a large network of heterogeneous systems with little or no impact on their performance, whereas host-based system usually has the potential to degrade host's operating system performance due to the processor time needed to perform monitoring functions.

With ever-increasing scale of distributed systems and rapid development of Internet, there has been growing interest and trend towards developing network-based detection system, largely due to its advantage in 1) monitoring a large network of heterogeneous systems simultaneously with single deployment and central management; 2) causing little or no impact on the performance of existing systems; 3) capability to effectively detect and mitigate DoS attacks. For this reason, we will focus on the network-based detection system in this survey.

3.2 Real-time vs. Delayed Detection

Based on the time of detection, we categorize detection systems into *real-time* detection and *delayed* detection. Real-time detection processes data continuously and performs DoS detection in real-time [22, 4], whereas delayed detection processes data at regular interval, which in turn delays the time of detection [9, 27].

Real-time detection has the advantage of detecting attacks and enabling mitigation response as soon as possible before further damage are inflicted. However, this may not be possible when (1) the audit data is collected from multiple source and aggregated at regular interval; and when (2) the system processing capacity can not keep up with the network throughput.

3.3 Programmed vs. Learned Detection

In *programmed* DoS detection system, the model or rules used for DoS attack detection are manually written by security experts. For instance Snort [24] performs real-time analysis on collected traffic data using rules that characterize the patterns of known attacks. However, this approach requires significant amount of manual effort to analyze and extract patterns from malicious traffic. In addition, the resulting rules may not extend well to other environment.

In recognition of this deficiency of programmed approach, there are increasing interest in developing *learned* DoS detection system where the rule-set or behavior model is learned from the traffic data using data mining or machine learning techniques in an automated way. Example systems include Honeycomb [13], which creates attack detection signatures using longest common string matching algorithm on malicious traffic captured by honeypot, and PAYL [31], which profiles byte frequency distribution of packet payload from the attack-free traffic observed in the past.

3.4 Detection vs. False Alarm rate

The key performance metric in DoS detection systems is of course the extend to which they can correctly detect attacks and avoid false alarms. The *detection rate* is defined as the percentage of actual attacks that are detected, whereas *false alarm rate* is calculated as the percentage of legitimate traffic that are classified as attacks. High false alarm rate essentially renders system impractical to deploy, because either legitimate traffic will be affected as a result of attack mitigation, or significant manual effort is required to analyze the output and exclude false alarms generated by the legitimate traffic.

Despite the interest in achieving high detection rate and low false alarm rate altogether, there is inevitable tradeoff between these two metrics in DoS detection systems. In anomaly-based DoS detection, the tradeoff is usually made by varying the detection threshold, such that high threshold provides lower false alarm rate at the cost of lowing detection rate, and vice versa.

The tradeoff between detection and false alarm rate depends on the choice of detection paradigm when designing the DoS detection system. In general, signature detection has low false alarm rate, but usually have low detection rate on unforeseen attacks; anomaly detection has potentially high detection rate, but its false alarm rate is also higher; classification-based detection is somewhere between the other two paradigms in terms of detection and false alarm rate. In recognition of this problem, hybrid systems have been designed which use both signature detection and anomaly detection to combine the benefits of both paradigms.

3.5 Denial-of-service Detection Paradigms

There are essentially three DoS attack detection paradigms, i.e. *signature detection*, *anomaly detection*, and *classification-based detection*. Here we present a brief overview of the paradigms and systems proposed, and discuss their advantages and limitations in detail.

3.5.1 Signature Detection (a.k.a misuse detection)

In signature detection the DoS detection decision is formed on the basis of knowledge accumulated of known attacks or system vulnerabilities. A repository of signatures is first generated by security expert to characterize the patterns of known attacks. These signatures are then matched against observed traffic to identify DoS attacks. An alarm is raised whenever a match is discovered. Example systems include NSM [9], Bro [22], and Snort [24].

Since signature detection looks for patterns that are known to cause security problems, it has the advantage of low false alarm rate and is able to provide forensic information about the cause of alarms, which can be used by security expert for further analysis. However, it is unable to detect any attacks that lie beyond its knowledge, thus resulting in low detection rate on unforeseen attacks. In addition, manually extracting representative signatures from large amounts of traffic data is usually time-consuming and labor-intensive, particularly with the increasing complexity and scalability of today's network traffic.

3.5.2 Anomaly Detection (a.k.a behavior detection)

Anomaly DoS detection techniques assume that an attack can be detected by observing a deviation from the normal behavior of network traffic. The profile of normal behavior is first extracted from normal traffic observed in the past or synthesized traffic that are known to be attack-free. Later in the detection mode, the observed network traffic is checked against the established profile of traffic behavior. An alarm is generated when the traffic deviates from the profile beyond a threshold. For instance, MULTOPS [8] uses disproportional packet rates to and from hosts as a heuristic to detect flood DoS attack. PAYL [31] detects occurrence of worm by profiling the byte distribution of packet payload and looking for deviation from the established profile in the monitored network traffic.

Advantages of anomaly detection lies in its capability to detect unforeseen attacks, since no priori knowledge about specific attacks is required. However, the observed anomaly does not necessarily indicate attacks, and false alarm rate can be high, because the entire scope of the behavior of network traffic may not be covered during the learning phase.

3.5.3 Classification-based Detection

Classification-based detection systems form a compound decision in view of both normal behavior of network traffic and abnormal pattern of the attacks. This is usually achieved by applying machine learning techniques to a labeled dataset, in which the attack and normal traffic are correctly labeled, to generate a classifier. The resulting classifier will then be used to classify monitored traffic into "normal" or "attack". For example, ADAM [3] and MADAM [15] build their classifier using decision tree algorithm, whereas Mukkamala et.al [21] builds the classifier using support vector machine and neural network algorithms.

In comparison to signature and anomaly detection, which derive their detection model from either attack-only or attack-free dataset, classification-based detection derives its classifier from labeled dataset containing both attack and normal traffic in an automated way that combines the benefits from both approaches.

3.6 Summary

Paradigm	**Training Data**	**Model**	**Systems**
Signature-based	Attack-only	Programmed	NSM, Bro, **Snort**, EMERALD, GrIDS, NetSTAT, Shield
		Learned	Honeycomb, Earlybird, Autograph
Anomaly-based	Normal-only	Programmed	NSM, **MULTPOS**
		Learned	PAYL, **PHAD**, IMAPIT, EMERALD
Classification-based	Labeled Mixture	Learned	**MADAM**, ADAM, ANN&SVM

Table 1: DoS Detection Paradigms

System	Ref	Year	Detection Paradigm[1]						Real-time DoS Detection	Detection Techniques
			Signature		Anomaly		Classification			
			P	L	P	L	P	L		
NSM	[9]	1990	✓		✓					rule-based
GrIDS	[27]	1996	✓							rule-based
EMERALD	[23]	1997	✓			✓			✓	rule-based
NetSTAT	[29]	1998	✓						✓	rule-based
Bro	[22]	1999	✓						✓	rule-based
Snort	[24]	1999	✓						✓	rule-based
Honeycomb	[13]	2004		✓						pattern matching
Earlybird	[25]	2004		✓						pattern matching
Autograph	[12]	2004		✓						pattern matching
PHAD	[19]	2002				✓			✓	header field distribution
IMAPIT	[5]	2002				✓				wavelet analysis
PAYL	[32]	2004				✓			✓	payload byte distribution
MULTOPS	[8]	2001			✓				✓	packet ratio
MADAM	[15]	1999						✓	✓	decision tree
ADAM	[3]	2001				✓		✓	✓	naive bayes
ANN&SVM	[21]	2002						✓	✓	artificial neural network

Table 2: Classification of surveyed systems according to the proposed taxonomy

In table 1 we conclude the classification of DoS detection paradigms with example systems. We further provide a classification of surveyed systems according to our proposed taxonomy in table 2, with a brief description of used techniques for each system.

In the following sections, we are going to selectively present the design and implementation of some representative systems (shown in bold in both tables). These include: (1)Snort, a system that provides signature-based light-weight intrusion detection; (2)PHAD, an anomaly-based nonstationary model for detecting novel attacks; (3)MADAM, an automated framework for constructing models for intrusion detection using machine learning algorithm; and (4) MULTOPS, a data structure for bandwidth attack detection. These systems cover most of the detection paradigms in our proposed taxonomy;

4 Snort: Signature-based Intrusion Detection for Networks

Snort [24] is a signature-based network intrusion detection system that performs real-time traffic analysis and packet logging on IP networks. It is intended to be a lightweight cost-efficient IDS that can be deployed to monitor small and lightly utilized networks. As one of the most widely deployed open-source

[1]Abbreviation used: P stands for *Programmed*, and L stands for *Learned*

IDS, Snort's architecture and rule language serve as a representative example of signature-based IDS.

4.1 System Architecture

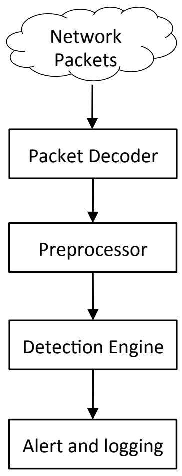

Figure 2: System Architecture of Snort

Figure 2 shows the system architecture of Snort. In attack detection mode, Snort monitors network traffic, analyzes it based on a rule set that encodes attack signature, and performs specific actions as identified in the rules that are matched by the network packets. The analysis is typically carried out in the following components:

- **Packet Decoder:** The Packet Decoder decodes the raw packets observed on the network according to the protocol that is used, from IP layer up to application layer. The decoded packet header values are stored in a data structure for later use in the Detection Engine.
- **Preprocessor:** The Preprocessor performs a variety of preprocessing other than the standard packet decoding, before the data can be analyzed by Detection Engine. These include IP fragment assembly, TCP stream assembly, packet header normalization, etc.
- **Detection Engine:** The Detection Engine carries out the actual attack detection by matching various values obtained in the previous steps against a set of rules that encodes patterns of known attacks. If a match is found, the corresponding action that is defined in rule will be executed, e.g. drop the packet, log the packet, generate alert to system administrator.
- **Logging and Alerting System:** This last component logs or generates system alerts based on the action specified in the matched rules as well as the options given at the start of the system.

4.2 Snort Rules

Snort rules are simple to write, yet powerful enough to detect a wide range of intrusive network traffic. Generally, each Snort rule is composed of action directive and attack signature. The action directive specifies the action to take when a packet matches the attack signature specified in the rule, which include: 1) *pass* rules that simply drop the packet, 2) *log* rules that write the full packet to the logging routine, and 3) *alert* rules that generate an event notification and log the full packet to enable later analysis. The attack signature specifies the combination of packet values that warrents further actions.

```
log tcp any any -> 10.1.1.0/24 22
```

Figure 3: A simple Snort rule

```
alert udp $EXTERNAL_NET any -> $HOME_NET any
(msg:"DOS Teardrop attack"; id:242; fragbits:M;)
```

Figure 4: Alert rule for teardrop attack

Figure 3 shows a most basic rule that specifies only protocol, direction, IP address and port number of interest. This rule would record all TCP packets that are destined to port 22 (ssh) and 10.1.1 class C network address space, regardless of its source IP address and port number. As a more advanced example, Figure 4 shows a rule that detects teardrop attack. This rule would fire for any UDP packet that is sent from outside network towards home network when the IP id of the packet is 242 and the "More Fragment" bit is set.

4.3 Evaluation

According to the taxonomy presented in Section 3, Snort is a signature-based network intrusion detection system that can detect DoS attack. We summarize the strength and weakness of Snort as follows.

Strengths:

- Snort employs a clear and concise rule language that describes per packet tests and actions. The ease of use simplifies and expedites the development of new attack detection rules, and allows Snort to rapidly respond to zero-day attacks. Therefore, Snort can be used to fill holes in commercial vendor's network-based IDS when signature updates are slow to come from vendor in response to new attack.

- Snort is easy to deploy and run on all of today's most popular operating systems and is not confined to a fully vested server hardware platform. It also provides an interactive GUI environment that allows user to configure the system and analyze incoming traffic. This makes Snort easy to use for even un-sophisticated users.

- Snort provides a signature repository that is actively updated on a daily basis to detect most recent attacks. The real-time access to these attack signatures enables users to respond to latest trends in intrusive activities and DoS attacks.

Limitations:

- Since the mechanism used for attack detection in Snort is based on signature matching, system administrators have to apriori specify the patterns of intrusive activity to look for. However, priori knowledge of attacks is not always available; incomplete signatures will result in undetected attacks and reduce detection rate.
- Snort is not designed to run in high speed networks without dropping packets or slowing down the traffic. Instead, it is intended to be deployed only in small and lightly utilized networks, or as a part of an integrated network security infrastructure with other systems.

5 PHAD: Nonstationary Models for Detecting Novel Attacks

PHAD [19] provides network anomaly detection by modeling normal behavior of attack-free network traffic. In contrast to other anomaly-based IDS whose model remains stationary after the model training, PHAD uses a nonstationary model that is continuously updated based on the monitored traffic. It monitors the entire data link, network, and transport layer, and assigns anomaly score to each packet based on the observed packet header values.

5.1 Learning Nonstationary Models

An important features provided by PHAD is the ability to adapt to state changes over time, as opposed to other stationary anomaly detection model which depends exclusively on the profile of training data. This is based on the observation that many types of network processes, such as the rate of a particular type of packet, are nonstationary. No matter how short or long the sample data is, the stationary model can not predict rate of events for other samples.

In order to find events that have the lowest probability, i.e. most anomalous, PHAD assigns to each event an anomaly score that is inversely proportional to the estimated probability of the event. As a simplification, the anomaly score is only assigned to novel events that have never occurred in the training data. Specifically, if an experiment is performed n times and r different values are observed in the training data, and the novel event is last observed t seconds ago in the monitored traffic, then the novel event is scored as tn/r. Note that n and r remain stationary after the model training, whereas t is continuously updated based on the monitored traffic during attack detection.

5.2 Packet Header Anomaly Detection

PHAD monitors 33 fields from Ethernet, IP, and transport layer (TCP, UDP, or ICMP) packet header for intrusion detection. For each field that has a value never observed in training, an anomaly score of tn/r is computed, and the sum of scores across different fields is assigned to the packet. If the final score exceeds a threshold, then an alarm is signaled.

$$\text{anomaly score} = \sum_i t_i n_i / r_i, \text{where field } i \text{ is novel in training.}$$

Clustering: In order to generalize continuous field values that span across a large range (e.g. TCP port), the set of observed values is clustered into a set of continuous ranges. Each newly observed value forms a cluster by itself; if the number of clusters exceeds a limit C, then two closest ones are merged into a single cluster. For purpose of anomaly detection, r is the number of times the set of clusters is updated.

Attribute	**r/n**	**Observed Values**
Ethernet Size	508/12814738	46 60-1181 1182...
Ether Dest Hi	9/12814738	x0000c0 x00105A...
Ether Dest Lo	12/12814738	x000009 x09B949...
Ether Protocol	4/12814738	x0136 x0800 x0806...
IP Header Len	1/12715589	x45
IP TOS	4/12715589	x00 x08 x10 xC0
IP Length	527/12715589	38-1500...
IP Frag ID	4117/12715589	0-65461 65462...
IP Frag Ptr	2/12715589	x0000 x4000...
IP Protocol	3/12715589	1 6 17
IP Checksum	1/12715589	xFFFF
IP Source Addr	293/12715589	12.0.169.104...
IP Dest Addr	287/12715589	0.67.97.110...
...	...	...

Table 3: Example PHAD Model [19]

Table 3 shows the example PHAD model for Ethernet and IP header fields after training on 7 days of attack-free network traffic with C = 32. It shows the name of each packet header, the observed values of n and r, and a partial list of observed values or clusters. For example, the first line says that out of 12,814,738 packets with an Ethernet size field, there are 508 times when the set of clusters are updated. Three of these clusters are 42, 60-1181, and 1182.

5.3 Application Layer Anomaly Detection

Application Layer Anomaly Detection (ALAD) can be combined with PHAD to improve detection rate. Instead of assigning anomaly scores to packets, ALAD assigns scores to incoming TCP connections that are assembled from packets. In particular, the anomaly scores are aggregated over the following five classes of attributes.

- **P(src IP | dest IP)**, where *src IP* is the external source address of the client making the request, and *dest IP* is the local host address. A separate pair of r and n is generated for each local *dest IP* based on the observed value of *srcIP* that connects to it. This class learns the normal set of clients for each host.
- **P(src IP | dest IP, dest port)**. This is similar to the class defined above except that there is a separate (r, n) for each server on each host.

It learns the normal set of clients for each server on each host, which may be different across servers on the same host.

- **P(dest IP, dest port)**. This class learns the set of local servers which normally receive requests. It should catch probes that attempts to access nonexistent hosts or services.
- **P(TCP flags | dest port)**. This model learns the set of normal TCP flag sequence for the first, next to last, and last packet of a connection. Separate models are generated for each port number, which usually indicates the type of service (e.g. mail, FTP, http). An anomaly can result if a connection is opened or closed abnormally, indicating an abuse of service.
- **P(keyword | dest port)**. This model examines the text in the incoming TCP stream to learn the allowable set of keywords for each service. A keyword is defined as the first word on a line of input. An anomaly indicates the use of a rarely used feature of service, which is common in many R2L attack.

Attribute	**r/n**	**Observed Values**
80 (HTTP)	13/83650	Accept-Charset:
		Accept-Encoding:
		Accept-Language:
		Accept:
		Cache-Control:
		Connection:
		GET
		Host:
		(5 values . . .)
25 (SMTP)	34/142610	(34 values . . .)
21 (FTP)	11/16822	(11 values . . .)

Table 4: Example ALAD Model [19]

Table 4 shows the example ALAD model for P(keyword|dest port) for ports 80, 25, and 21 after training on the same 7 days of attack-free network traffic as used in Table 3. For example, the first line says that only 13 different keywords were observed out of 83,650 TCP connections to port 80. Part of these keywords are listed in the third column.

5.4 Evaluation

Table 5 shows the detection rate of PHAD and ALAD on DARPA 1999 dataset. The anomaly threshold is configured to allow 10 false alarms per day. It shows that PHAD and ALAD can achieve 54.2% detection rate on DoS attacks without requiring any prior knowledge of attacks. Further, notice that there is almost no overlap between PHAD and ALAD. PHAD detects mostly probes and DoS attacks that exploit low level network protocols, whereas ALAD detects user behavior anomalies (all U2R and some R2L) at the application layer. We summarize their strength and limitations as follows.

Strengths:

Category	Total Detected (%)	By PHAD (%)	(By ALAD (%))
Probing	32.4	21.6	13.5
DoS	54.2	28.8	25.4
U2R	34.7	0.0	34.7
R2L	25.7	0.0	25.7
Overall	38.9	13.9	25.6

Table 5: Detection Rate of PHAD and ALAD on DARPA 1999 Dataset [19]

- There is significant non-overlap between PHAD and other systems that use signature detection. Therefore, they can be deployed in combination with other signature-based systems to improve detection performance.
- PHAD and ALAD are non-stationary models that adapt to system state changes over time. This approach avoids the problem of stationary model whose performance degrades as a result of established profile of traffic behavior becoming outdated with the evolving network traffic.

Limitations:

- There is combinatorial explosion problem of combining attributes. ALAD uses combination of the form P(X, Y) and P(Y|X), but these were ad-hoc. It will be helpful to automate the selection of attributes from among the exponential number of possibilities in an efficient way.
- Even when an anomaly is detected, PHAD can not provide much forensic information to characterize detected attack (e.g. the specific vulnerability that the attack exploits). As an anomaly-based detection system, PHAD detects anomaly by comparing the current system behavior with established profile using high level statistics. Consequently, it does not necessarily have enough information to analyze the cause of intrusion.

6 MADAM: A Data Mining Framework for Intrusion Detection

MADAM [15] is a classification-based automated framework for building intrusion detection models. The previous signature-based Snort and anomaly-based PHAD require either manual process to analyze and encode attack signature, or guesswork to select statistical measures for normal usage profiles. In contrast, MADAM enables automated selection of predictive features by using data mining algorithms, and applies machine learning algorithm on the labeled audit data to generate intrusion detection models. Therefore, MADAM presents a systematic and automated approach for building intrusion detection systems.

6.1 System Framework

Figure 5 shows the system framework of MADAM. In the process of applying MADAM, (1) raw audit data is first processed into ASCII network packet information; (2) these are in turn summarized into connection records containing a number of basic features, such as destination port and connection duration; (3)

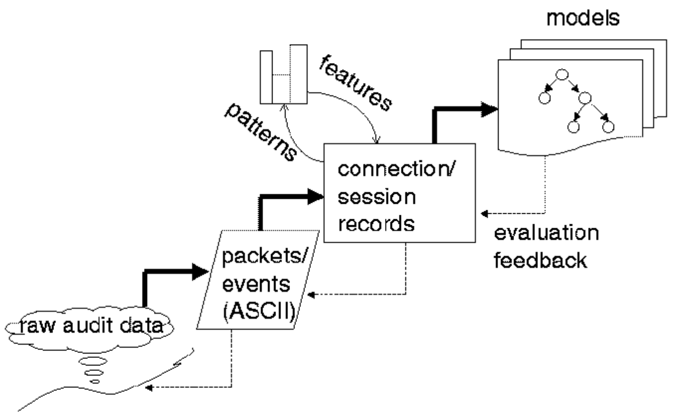

Figure 5: System Framework of MADAM [15]

data mining programs are then applied to the connection records to compute the frequent patterns; 4) finally, machine learning program is applied to the processed connection records to generate the detection model. Note that the whole process is iterative. For example, poor performance of the machine learning model often indicates that more pattern mining and feature construction is needed at earlier steps.

6.1.1 Basic Feature Construction

Raw audit data is first analyzed into network connection records using preprocessing programs, where each record has a set of predefined feature, such as connection duration, source IP, destination port, number of bytes transferred. Note that these are all basic features that characterize general aspects of the observed traffic, rather than some features that are intentionally selected for attack detection. Table 6 shows an example connection record.

timestamp	duration	service	src_ip	dst_ip	src_byte	dst_byte	flag	...
1.1	0	http	spoofed_1	victim	0	0	S0	...
1.1	0	http	spoofed_2	victim	0	0	S0	...
1.1	0	http	spoofed_3	victim	0	0	S0	...
...	...	...	...	...	...	...	...	...
10.1	2	ftp	addr1	addr2	200	300	SF	...
12.3	1	smtp	addr2	addr4	250	300	SF	...
13.4	60	telnet	addr1	addr4	200	12100	SF	...
...	...	...	...	...	...	...	...	...

Table 6: Network connection records [15]

6.1.2 Automatic Feature Construction

Frequent episode analysis is applied to both normal connection data and attack data. The resulting patterns are then compared to find the attack-only patterns. Briefly, frequent episode analysis can discover those time-based sequence of audit events which frequently occur together, and use those patterns to provide guidelines for incorporating temporal statistical measures into intrusion detection models.

For example, consider the SYN flood attack record shown in table 6. Here the attacker used many spoofed source addresses to send a lot of S0 connections to HTTP port of the victim in a short time interval. Table 7 shows the top attack-only pattern extracted from this record using frequent episode analysis.

Frequent episode	Meaning
(service = http, flag=S0, dst_ip = victim), (service = http, flag = S0, dst_ip = victim) $\rightarrow$ *(service = http), flag = S0, dst_ip = victim* [0.93, 0.03, 2]	93% of the time, after two http connections with S0 flag are made to host victim, within 2 seconds from the first of these two, the third similar connection is made, and this pattern occurs in 3% of the time.

Table 7: Example Attack Pattern [15]

Each of these intrusion patterns is used as a guideline for incorporating additional features into the connection models to build better classification model. In particular, for a given intrusion pattern with reference feature F_0 and a time window of w seconds, the following features are added.

- A count of these connections.
- Let F_1 be service, src_ip or dst_ip other than F_0. If the same F_1 value is in all item sets of the episode, add a percentage of connections that share the same F_1 (e.g. http) value as current connection; otherwise, add a percentage of different values of F_1.
- Let V_2 be a value (e.g. S0) of a feature F_2 (e.g. flag) other than F0 and F1. If V_2 is in all the item sets of the episode, add a percentage of connections that have the same V_2 ; otherwise, if F_2 is a numerical feature, add an average of the F_2 values.

To provide a better understanding of the output from automatic feature construction, Here we provide a summary of temporal and statistical features that are constructed by MADAM from network audit data.

- The "same host" features that examine only the connections in the past 2 seconds that have the same destination host as the current connection: the count of such connection, the percentage of connections that have the same service as current connection, the percentage of different service, the percentage of S0 flag, and the percentage of REJ flag.
- The "same service" features that examine only the connections in the past 2 seconds that have the same service as current connections: the count of such connections, the percentage of different destination host, the percentage of S0 flag, and the percentage of REJ flag.

6.1.3 Classification

Provided with the labeled audit data and a set of features constructed from previous steps, machine learning algorithm can be used to automatically learn the classification model. Specifically, MADAM uses RIPPER [6], a classification rule learning algorithm, to generate rules for classifying connection records into "normal" or "attack". Table 8 shows an example RIPPER rule for detecting smurf DoS attack.

RIPPER rule	Meaning
smurf :- service=ecr_i, host_count > 5, host_srv_count > 5	If the service is icmp echo request, and for the past 2 seconds, the number of connections that have the same destination host as the current one is at least 5, and the number of connections that have the same service as the current one is at least 5, then this is a smurf attack (a DOS attack)

Table 8: Example RIPPER Rule for Detecting Smurf Attack [15]

Meta-classification: Finally, meta-classification mechanism can be used to combine the results from multiple detection models to further improve detection performance. In particular, three base classification models can be generated based on different sets of features to detect different categories of attacks. The output from these base models are converted into meta-level records that consist of four features: the three predications from each of the base models, plus the true class label. RIPPER is then applied to learn the rules that combine the results to make a final prediction on a connection.

6.2 Evaluation

Category	**Old (%)**	**New (%)**
Probing	97.0	96.7
DoS	79.9	24.3
U2R	75.0	81.8
R2L	60.0	5.9
Overall	80.2	37.7

Table 9: Detection Rate of MADAM on DARPA 1998 Dataset [15]

Table 9 shows the detection rate of MADAM on DARPA 1998 dataset. Here, new intrusions refer to those that did not have corresponding instance in the training data. Note that MADAM achieves 79.9% detection rate on known DoS attacks and 24.3% detection rate on novel DoS attacks. We summarize the strength and weakness of MADAM as follows.

Strengths:

- MADAM provides an automated framework for selecting predictive features and building intrusion detection models by applying data mining algorithms. This automatic approach eliminates the need to manually analyze and encode attack signatures, as well as guesswork in selecting predictive statistical features for normal traffic behavior.

- Enabled by the use of meta-learning, MADAM provides adaptability and extensibility in building intrusion detection systems. Meta-learning improves detection performance by combining results from multiple models. Thus new mechanisms can be introduced into existing IDS without significant re-engineering.

Limitations:

- The classification-based approach adopted in MADAM requires a labeled dataset that contains both target intrusions and normal traffic, which itself is hard to obtain. Improper labeling in the training dataset may lead to false positive or false negative.
- The MADAM assumes the presence of representative target attacks in the training dataset. Consequently, the features constructed based on available attack instance are potentially specialized to known attacks.

7 MULTOPS: A Data Structure for Bandwidth Attack Detection

MULTOPS [8] is a data structure that enables router or network monitors to detect ongoing bandwidth attack. It uses disproportional packet rates to and from hosts and subnets as a heuristic to detect attacks. To keep track of ratio of forwarding packets to reverse packets for each address prefix, a tree-shaped data structure is used to collect these statistics in real-time to look for disproportional packet rate behavior. Further, tree expansion and contraction are used to avoid memory explosion and reduce CPU utilization.

Unlike prior systems that examine the value of packet payload and header fields for attack detection, MULTOPS relies only on the destination IP address extracted from the IP header. This is due to the distinct nature of bandwidth attacks: attackers release high volume of network traffic towards victim to exhaust its network or system resource where attack traffic (i.e. header and payload value) can be completely indistinguishable from legitimate traffic.

7.1 Proportional Packet Rate Assumption

MULTOPS is based on the assumption that under normal operations, the packet rate in one direction is proportional to the packet rate in the opposite direction. The heuristic appears to hold broadly for most TCP and UDP applications: (1) TCP acknowledges every single—or every k—received packets, and, therefore, has proportional packet flows; (2) Although UDP does not provide delivery confirmation, in benign UDP-based applications there is usually a mechanism in the application layer that makes sure the other party has received the packets, similar to TCP ACK. Note that the second assumption is empirically validated on UDP traffic collected from a large number of production networks [4].

MULTOPS collects packet rates to and from address prefixes so that, given an address prefix P, ratio $R(P)$ of forward packet with destination address prefix P to reverse packet with source address prefix P can be calculated. Under normal circumstances, ratio $R(P)$ is close to some constant k for all prefix P. If ratio $R(P)$ drops below R_{min} or exceeds R_{max}, then future packets destined for a host with address prefix P may be dropped. The sensitivity of MULTOPS can be tuned by changing the values of R_{min} and R_{max}.

7.2 Data Structure

In order to carry out the detection technique proposed above at network routers where throughput can be more than 100Gbps, an efficient design and implementation of data structure is needed to keep track of packet rate with reasonable precision while maintaining acceptable memory footprint. To meet this requirement, MULTOPS maintains a tree of nodes that collects packet rate statistics for subnet prefixes at different aggregation levels. Further, it dynamically expands and contracts its shape to (1) locate the longest common prefix of victims' IP address(es), and (2) avoid (maliciously intended) memory exhaustion.

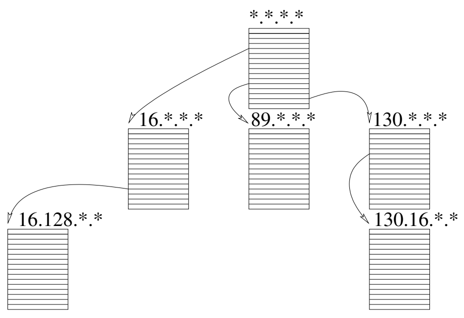

Figure 6: MULTOPS Data Structure [8]

Figure 6 shows a sample MULTOPS Data Structure. MULTOPS is organized as 4-level 256-ary tree to cover the entire IPv4 address space. Each node in the tree is a table consisting of 256 records, each of which consists of 3 fields: traffic rates to and from the represented address prefix, and a pointer pointing to a node in the next level of the tree. A table stores packet rates to and from IP addresses with a common 0-bit, 8-bit, 16-bit, and 24-bit prefix, depending on the level of the tree. Thus, the root node contains the aggregate packet rates to and from address 0.*.*.*, 1.*.*.*, 2.*.*.*, etc.

Expansion: If the to-rate or the from-rate for an address with n-bit prefix P exceeds the expand threshold, MULTOPS creates a child node under the record for prefix P to keep track of packet rates for addresses with (n+8)-bit prefix P'. Lowering this expand threshold increases precision of MULTOPS, but also increases its memory use. Figure 7 shows how a new node is added to the tree to keep track of all packet rates to and from addresses with prefix 130.16.120.*

Contraction: The reverse of expansion is contraction. Contracting a record

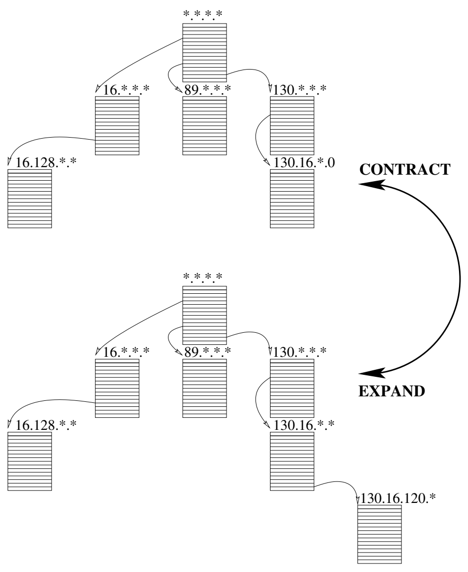

Figure 7: Expansion and Contraction [8]

involves removing a subtree from under a record. A subtree is contracted when the aggregate packet rate for that subtree drops below R_{max}. Contraction is done to constrain memory use and to avoid (maliciously intended) memory exhaustion. Figure 7 shows how a node is contracted.

7.3 Evaluation

MULTOPS is a programmed anomaly-based system that focuses exclusively on detecting DDoS bandwidth attacks at network router. We summarize its strength and limitation as follows:

Strengths:

- MULTOPS enables routers or network monitors to detect ongoing bandwidth attacks which create disproportional packet flows between sender and receiver.
- MULTOPS performs detection at router rather than at the victim network, so that malicious packets can be stopped before they cause any damage.

Limitations:

- MULTOPS may cause collateral-damage to legitimate traffic when it rate-limits or drops all packets destined to those hosts that are identified as being under attack.
- MULTOPS fails to detect those attacks that deploy a large number of proportional flows to cripple a victim. For example, attackers can use FTP or HTTP connections with relatively low but still proportional packet rates.

8 Distributed Denial-of-service Mitigation

Once an attack is detected, immediate action can be taken to mitigate its impact on the network. This typically involves *rate limiting* or *blocking* suspicious traffic at the local network. Although such mitigation approach can effectively stop DoS vulnerability attack, it is essentially useless against DDoS attack, because the presence of malicious traffic on the local network itself is already denying the use of resources to legitimate users.

Two alternative approaches have been proposed to mitigate DDoS attack by filtering out suspicious traffic before they can reach victim. The first approach, called *Pushback* [10], filters out suspicious traffic at upstream routers closer to source of attack, whereas the second approach redirects traffic to a *cleaning center* [2] that performs traffic cleaning before sending them back to the original destination.

8.1 Pushback

In Pushback [10], once an attack is detected at the victim, the characterization of the suspicious traffic, such as source or destination IP address of packets, can be sent to the upstream router, which blocks the traffic that matches the given characterization.

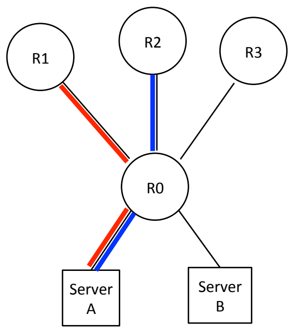

Figure 8: DDoS attack in progress

To illustrate Pushback, consider the network in Figure 8. R_n are ISP routers with processing capacity 100Gbps; A, B are commercial servers. Suppose DDoS attack is launched against server A via router R1 and R2 with a throughput of 60Gbps each, denoted in red and blue. Consequently, router R0 is overloaded, causing collateral damage to the legitimate traffic from R3 to B. Simply blocking traffic destined to server A at router R0 will not help mitigate its impact, since it is still overloading processing capacity at server A.

By using Pushback, R0 can request server R1 and R2 to block all the traffic that are destined to server A. This approach immediately provides two benefits: (1) server B can now successfully receive legitimate traffic via router R0; (2) router R1 and R2 can perform more complex analysis to filter out malicious traffic, since the traffic load on them is within their capacity. Further, this process can be repeated until it reaches the router that is closest to the source of attack.

8.2 Cleaning Center

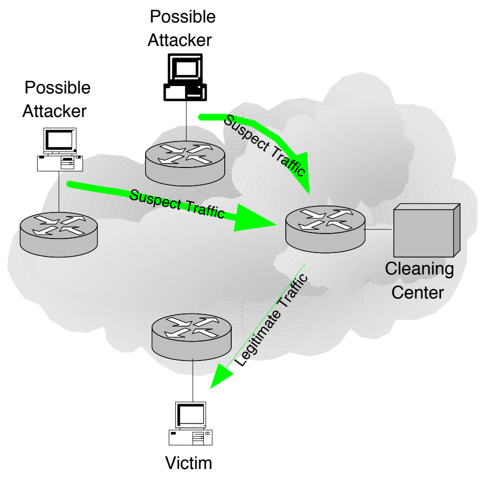

Figure 9: Cleaning Center Operation [2]

An alternative approach to address the problem of DDoS attacks is to redirect traffic to and from a *cleaning center* [2] that performs the necessary traffic cleaning when a set of destination hosts is under attack.

In comparison to detecting and mitigating DDoS attack at each individual enterprise network, cleaning center provides this service at ISP level for multiple networks, thus achieving two advantages: (1) the cost of purchasing specialized equipment and employing security experts is greatly amortized among many enterprise customers; (2) the capacity of backbone is large enough to absorb the attack traffic without having to drop network packets.

9 Discussion

In this section, we conclude our survey paper with a comparison of the systems that we have presented in previous sections, followed by a discussion of the challenges.

9.1 Comparison of Denial-of-service Detection Techniques

Here we evaluate the each of the four presented denial-of-service detection systems with regard their key performance metrics. The results are summarized in Table 10, where the cells with the highest performance on each line are denoted in red color.

- **Detection rate of known DoS vulnerability attack:** Among the systems we have surveyed, Snort has the highest detection rate on known DoS vulnerability attacks by using its actively updated attack signatures. MADAM also provides high detection rate due to its uses of a labeled dataset of known attacks. PHAD has moderate detection rate since it

Performance	Snort	MADAM	PHAD	MULTOPS
Detection rate of known vulnerability attack	High	High	Medium	Low
Detection rate of unknown vulnerability attack	Low	Low	Medium	Low
Detection rate of flood attack	Yes	Yes	No	Yes
False alarm rate	Low	Low	Medium	Low
Ease of model construction	Medium	High	High	High
DoS flood attack mitigation	No	No	No	Yes

Table 10: A Summary of Performance Evaluation for Surveyed Systems

does not have any prior knowledge of attack. MULTOPS is not able to detect DoS vulnerability attacks.

- **Detection rate of unknown DoS vulnerability attack:** Snort has the lowest detection rate on unknown attacks due to its reliance on existing attack signatures. MADAM has somewhat better, but still low, detection rate on known attacks. PHAD does not require prior knowledge of specific attacks, and achieves moderate detection rate. Again, MULTOPS is not able to detect DoS vulnerability attacks.

- **DoS flood attack mitigation:** MULTOPS is the only system here that provides mitigation against DoS flood attacks before they can reach victim. The other three systems can not mitigate DoS flood attacks.

- **False alarm rate:** Both Snort and MADAM have low false alarm rate due to their focus on detecting known attacks. PHAD, as an anomaly based detector, has relatively high false alarm rate. The false alarm rate of MULTOPS depends on the effectiveness of proportional packet rate assumption.

- **Ease of training:** Snort requires the most effort to manually analyze and write attack signatures. MADAM provides an automated framework for building detection models, thus requiring much less manual effort. Similarly, PHAD constructs the behavior profile of normal traffic in an automated way. MULTOPS is the easiest to build among all four systems, since only a predefined threshold for expected packet ratio is required.

9.2 Challenges

- **Extensibility:** Extensibility is critical in today's network computing environment. Specifically, an extensible detection system should 1) allow additional modules or functionalities to be incorporated into existing system without significant re-engineering; (2) enabling emerging attacks to be detected without having to re-train the entire model. Currently, most anomaly-based systems (e.g. PHAD) need to completely re-train the behavior model when the traffic changes. More recent systems, such as Bro [22], have made extensibility their primary goal.

- **Forensic Information:** In addition to detecting attacks, an ideal detection system should also provide forensic information that could help a security analyst identify important characteristics of the attack and take immediate response. Relevant information includes the system vulnerability exploited, the common substring in the packet payload, and the impact on the system. Currently, there lacks a standard and focus on providing these information.

- **Scalability:** It is becoming increasingly difficult for uniprocessor systems to process rapidly growing network throughput with increasingly complex analysis required for attack detection. In order to meet this performance requirement, a scalable design that can exploit today's multi-core processors is needed. That requires careful design, however, to (1) evenly balance load across cores and (2) support exchange of states across threads. It is a popular area of research in recent years to parallelize existing intrusion detection systems on commodity PC [11] or specialized hardware [28].

- **DDoS Attack Mitigation:** Despite the growing popularity and interest, DDoS attack remains a major threat facing enterprise networks of all size, specifically, for two reasons: (1) it would be prohabitively expensive or even impossible to deploy enough computation and network resources at each enterprise network that can combat DDoS attack. (2) dropping suspicious packets usually cause collateral damage to legitimate packets, because the attack traffic behaves like the normal traffic from legitimate users. As an alternative solution, efforts have been made to mitigate DDoS attack at upstream router [10] or at cleaning center [2] which has enough resource to filter out attack traffic.

10 Acknowledgements

I would like to thank Prof. Jonathan M. Smith for chairing my WPE-II committee, as well as Prof. Boon Thau Loo and Matt Blaze for sitting on the committee.

References

[1] Cyber warfare incident. http://en.wikipedia.org/wiki/Cyberwarfare#Incidents.

[2] S. Agarwal, T. Dawson, and C. Tryfonas. Ddos mitigation via regional cleaning centers. Technical report, Sprint ATL Research Report RR04-ATL-013177, 2003.

[3] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: Detecting intrusions by data mining. In *In Proceedings of the IEEE Workshop on Information Assurance and Security*, 2001.

[4] A. G. Bardas, L. Zomlot, S. C. Sundaramurthy, and X. Ou. Classification of udp traffic for ddos detection. 2012.

[5] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Internet Measurement Conference: Proceedings of the 2 nd ACM SIGCOMM Workshop on Internet measurment*, 2002.

[6] W. W. Cohen. Fast effective rule induction. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 1995.

[7] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to ddos attack detection and response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, 2003.

[8] T. Gil and M. Poletto. *MULTOPS: a data-structure for bandwidth attack detection*. 2001.

[9] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, 1990.

[10] J. Ioannidis and S. Bellovin. Implementing pushback: Router-based defense against ddos attacks. 2002.

[11] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park. Kargus: a highly-scalable software-based intrusion detection system. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 317–328. ACM, 2012.

[12] H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[13] C. Kreibich and J. Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM Computer Communication Review*, 2004.

[14] S. Kumar and E. H. Spafford. A pattern matching model for misuse intrusion detection. 1994.

[15] W. Lee and S. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 2000.

[16] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, 1999.

[17] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 2000.

[18] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, 2000.

[19] M. Mahoney and P. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[20] D. Moore, G. Voelker, and S. Savage. Inferring internet denial-of-service activity. Technical report, DTIC Document, 2001.

[21] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, 2002.

[22] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 1999.

[23] P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling response to anomalous live disturbances. In *Proceedings of the 20th national information systems security conference*, 1997.

[24] M. Roesch et al. Snort-lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, 1999.

[25] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[26] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, et al. Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, 1991.

[27] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. Grids-a graph based intrusion detection system for large networks. In *Proceedings of the 19th national information systems security conference*, 1996.

[28] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *Recent Advances in Intrusion Detection*, 2008.

[29] G. Vigna and R. A. Kemmerer. Netstat: A network-based intrusion detection approach. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, 1998.

[30] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *ACM SIGCOMM Computer Communication Review*, 2004.

[31] K. Wang, G. Cretu, and S. Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advances in Intrusion Detection*, 2004.

[32] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, 2004.