Survey of Current Network Intrusion Detection Techniques

Sailesh Kumar, sailesh@arl.wustl.edu

Abstract:

The importance of network security has grown tremendously and a number of devices have been introduced to improve the security of a network. Network intrusion detection systems (NIDS) are among the most widely deployed such system. Popular NIDS use a collection of signatures of known security threats and viruses, which are used to scan each packet's payload. Signature based designs have low false positive rates, and they are effective and accurate in combating against the known security threats. However, they remain completely ineffective against those attacks that are yet unknown; these can be combated only after they are detected manually and a signature is created for them.

Since new threats are potentially more lethal, a number of pro-active designs have been proposed, which can detect new security events such as propagation of a new and unknown virus or worm. Such systems accomplish this by creating a profile of normal Internet traffic, and then using this profile to continuously monitor the network activity for suspicious activity. As the system senses an anomaly, or a dramatic change in traffic characteristics, it takes certain actions such as raising an alarm or discarding certain traffic. In this Survey paper, we will evaluate a number of current NIDS systems and the algorithms they employ to detect and combat security threats, both from technical and economical perspective.

Keywords:

NIDS, Anomaly Detection, Network Security, Security Signature, Pattern Matching

Table of Contents

1. Introduction

Network security has recently received an enormous attention due to the mounting security concerns in today's networks. A wide variety of algorithms have been proposed which can detect and combat with these security threats. Among all these proposals, signature based Network Intrusion Detection Systems (NIDS) have been a commercial success and have seen a widespread adoption. While, these systems already generate several hundreds of million dollars in revenue, it is projected to rise to more than 2 billion dollars by 2010.

A NIDS aims at detecting possible intrusions such as a malicious activity, computer attack and/or computer misuse, spread of a virus, etc, and alerting the proper individuals upon detection. A NIDS monitors and analyzes the data packets that travel over a network looking for such suspicious activities. A large NIDS server can be set up on the links of a backbone network, to monitor all traffic; or smaller systems can be set up to monitor traffic directed to a particular server, switch, gateway, or router. Another class of NIDS can be setup at a centralized server, which will scan the system files, looking for unauthorized activity and to maintain data integrity.

There are two primary approaches to NIDS implementation: signature based, and anomaly detection based. The first approach has become a commercial success. A signature based NIDS maintains a collection of signatures, each of which characterizes the profile of a known security threat (e.g. a virus, or a DoS attack). These signatures are used to parse the data streams of various flows traversing through the network link; when a flow matches a signature, appropriate action is taken (e.g. block the flow or rate limit it). Traditionally, security signatures have been specified as a string signature, port signature and header condition signature.

String signatures are a string of ASCII symbols that characterizes a known attack. For example, such a string signature in UNIX can be "cat "+ +" > /.rhosts" , which if executed, can cause the system to become extremely vulnerable to network attack. Simpel strings may lead to high false positives, therefore it is important to refine the string signature; for this purpose one may use a compound string signature. Such a compound string signature to detect a common Web server attack can be "cgi-bin" AND "aglimpse" AND "IFS".

Port signatures commonly probes for the connection setup attempts to well known, and frequently attacked ports. Obvious examples include telnet (TCP port 23), FTP (TCP port 21/20), SUNRPC (TCP/UDP port 111), and IMAP (TCP port 143). If these ports aren't being used by the site at a point in time, then the incoming packets directed to these ports are considered suspicious.

Header signatures are designed to watch for dangerous or illegitimate combinations in packet headers fields. The most famous example is Winnuke, in which a packet's port field is NetBIOS port and one of the Urgent pointer, or Out Of Band pointer is set. In earlier version of Windows, this resulted in the "blue screen of death". Another well known such header signature is a TCP packet header in which both the SYN and FIN flags are set. This signifies that the requestor is attempting to start and stop a connection simultaneously.

Some well known commercial NIDS include AXENT (www.axent.com), Cisco (www.cisco.com), CyberSafe (www.cybersafe.com), ISS (www.iss.net), and Shadow (www.nswc.navy.mil/ISSEC/CID), while the popular open source NIDS includes Snort, and Bro.

While signature based NIDS has been widely deployed, anomaly based NIDS have not gained popularity yet, and they have remained a topic of an ongoing interest among the research community. The critical advantage of such NIDS over signature based NIDS is its promise to detect and contain security violations before they propagate and cause any damage. Signature based systems are reactive, in that they combat against known attacks, that have already affected and damaged a number of systems before being identified; anomaly based systems are pro-active and autonomous and can ensure security without any manual interference.

Anomaly based NIDS monitors network traffic and compares it against an established baseline of normal traffic profile. The baseline characterizes what is "normal" for the network - such as the normal bandwidth usage, the common protocols used, correct combinations of ports numbers and devices - and alerts the administrator or user anomalous traffic is detected which is significantly different from the baseline. It is highly subjective to decide what can be considered normal and what an anomaly, but a widely accepted rule of thumb is that, any incident which occurs on a frequency greater than two standard deviations from the statistical norm should be considered suspicious. An example of such behavior would be if a normal user logs on and off of a machine 20 times a day instead of the normal course of 1 or 2 times. Anther example is, when a user computer is used at 2:00 AM at a time when no one outside of the business hours have access; this should also raise some suspicions. At another level, a NIDS can investigate the user patterns, such as profiling the programs that are often executed, etc. If a user in the administrative department suddenly starts to execute programs from the engineering division, or begins to compile a code, then the system can promptly alert the administrators.

Clearly, such anomaly based intrusion detection may lead to a high rate of false detection, which we call false positives. It is generally considered difficult to keep low false positives in any system that sets aggressive policies to detect anomalies. For example, it may be difficult to distinguish flash crowd from a Distributed Denial of Service attack (DDoS), thus a system may raise false alarm during a flash crowd event assuming that it is a DDoS attack. Similarly network reconfigurations and transient failures may abruptly change the traffic profile falsely raising the alarm. The second challenge concerns with the assumption made by these systems that attacks are always anomalous, which may not necessarily be true. An intelligent attacker may develop intrusion techniques which will cause minimal disruption in the underlying traffic, thus may go undetected.

The final challenge in designing these systems concerns with the availability of dataset that is representative of normal traffic. To be realistic, the assumption that there exists attack-free data for training a detector outside of simulated data is not a realistic assumption. Typical network traffic contains a large number of scans, denial-of-service attacks and backscatter, and worm activity. If not careful, this activity will become part of the normal state for an anomaly detector.

In this survey paper, we describe the in-depth design details of both signature- and anomaly based NIDS. Specifically, for signature based systems we cover the following topics:

1. An overview of the system - high level architecture and principles of intrusion detection.
2. Current well known systems and the algorithms and architectures employed by them.
    a. Signatures that are used.
    b. Hardware/system architecture.
    c. Performance and limitations.
3. Effectiveness in improving the network security.
4. The disadvantages of such systems.

For the anomaly based systems, we cover the following topics:

1. Overview of anomaly detection systems, and high level architecture.
2. Well known algorithms that are used to detect anomalous behavior within well behaved traffic.
    a. Unsupervised clustering algorithms.
    b. Entropy based methods.
    c. Data mining techniques.

3.How anomaly detection can be used to combat security threats, and the discussion of their possibly important role in the near future.
4. The disadvantages and challenges in implementing such systems and a brief description of the techniques that are commonly used by attackers to evade them.

Back to Table of Contents

2. NIDS and Network Architecture

In this section, we explain how NIDS are deployed in a given network [Thakker03]. In order to maintain clarity, we consider a NIDS as a black box (in next section we discuss the architecture of NIDS in greater detail), and list the popular configurations and locations, where they are deployed to tap into the network links and detect security violations.

2.1 Early Warning Mode

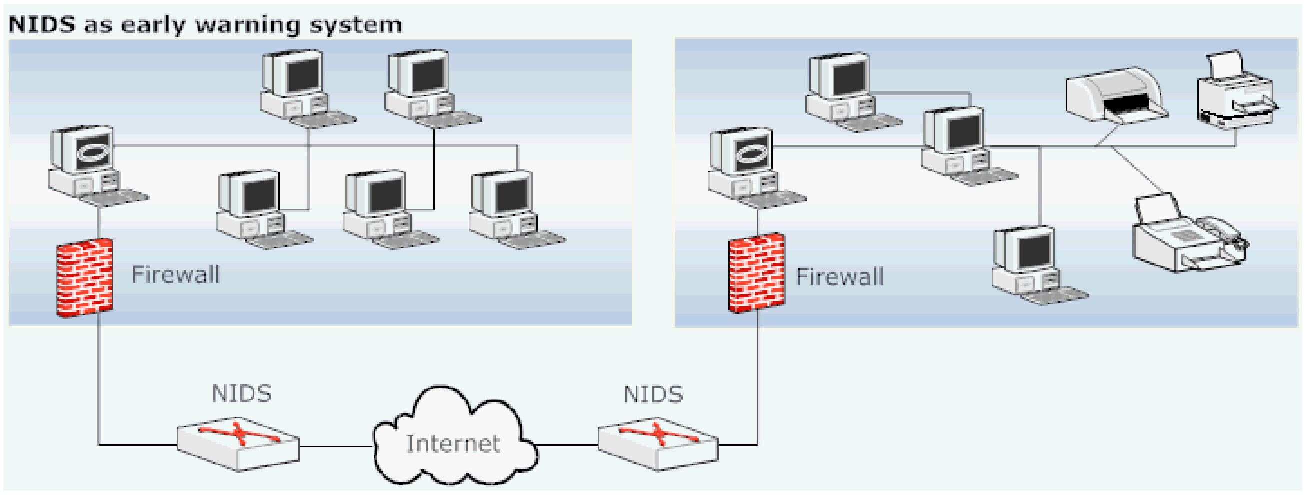

Figure 1: A NIDS as an early detection system.

In such a mode of operation, NIDS are employed outside the perimeter of the firewall (shown in Figure 1). Thus, all traffic entering the host and/or the local/enterprise network is scanned by the NIDS. The key benefit of such configuration is that the NIDS remains at a single locating tapping at a high speed link and can potentially serve a large number of hosts. Thus, the management and update of the signatures and keeping the configurations up-to-date are much easier. A drawback is that the attacks initiated by the hosts within the firewall perimeter will go undetected. Also, notice that in such architecture, it is possible that the NIDS will raise an alarm while the firewall will block the traffic, thereby effectively rendering the alarm a false one.

2.2 Internal Deployments

In such mode of operation, a NIDS is deployed such that it monitors the traffic that traverses any given link within the network, thereby providing an increased security (shown in Figure 2). Thus the NIDS is deployed near the switching nodes within the local network, and near the access routers at the network boundary. In such configurations, the NIDS will no longer monitor the traffic that has been blocked by the firewall, which will lead to a much reduced false alarm rates. A drawback however is that there will be multiple instances of NIDS, and it will become tedious to keep all of them up-to-date in say a large enterprise network. Such configurations are popular in ecommerce back end networks, consisting of web and mail servers and database and storage servers, as an increased security is desirable there. It also aids in keeping an infected server to infect the others within the network.

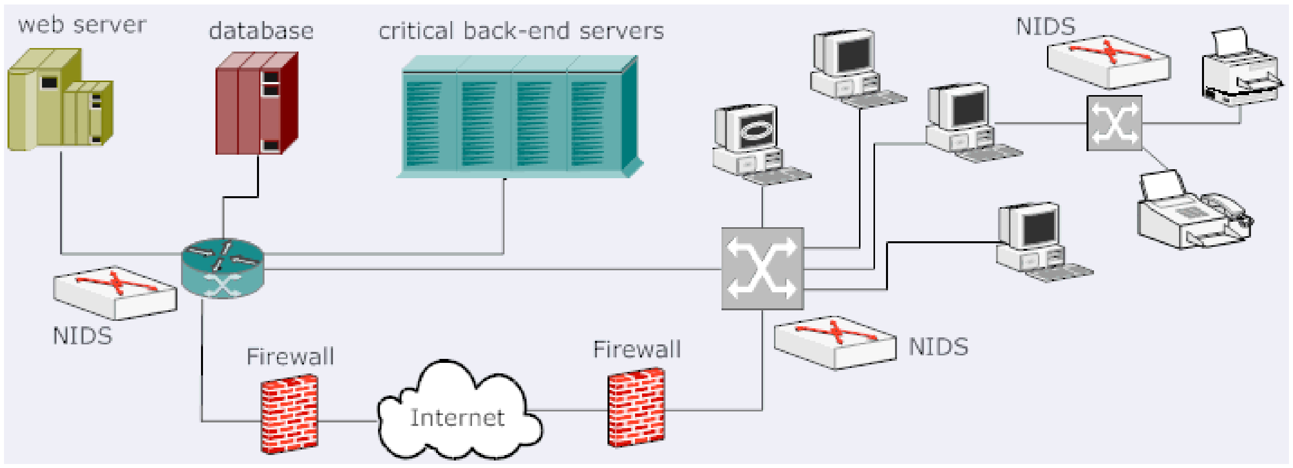

Figure 2: NIDS in complete deployment mode.

2.3 NIDS within Every Host (like an anti-virus)

In such configurations, every host has an inbuilt NIDS attached to all of its network interfaces. In a way, such architecture is similar to an anti-virus running on the host; however its key benefit is that the NIDS is decoupled from the host operating system, thus it can be separately managed by the network administrator through a central location. Nevertheless, the management can become complex when the network is large containing several host computers. It has been argued that such structures can lead to difficulty in implementation of the NIDS algorithms as a single instance of NIDS will remain unaware of the traffic traversing through the other links; thus attacks such a Distributed Denial of Service (DDoS) might go undetected. Another disadvantage arises due to the extensive use of devices such as access gateways which dynamically assigns IP addresses to the local hosts. Due to the limited local scope of these IP addresses, it sometimes becomes difficult for the host based NIDS to effectively trace the route of the packet, which affects its detection mechanism and capability.

With the description of the above 3 popular NIDS deployment configurations, we proceed to the NIDS architecture and the algorithms that are used to implement NIDS. NIDS has traditionally been designed with two popular techniques: a signature based detection and a relatively advanced implementation called anomaly based detection. We begin with the description of the signature based design.

Back to Table of Contents

3. Signature Based NIDS

A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. Signatures specify a combination of packet header and packet content inspection rules to identify the anomalous traffic flows. Packet header rule consists of a filter on packets 5-tuple (source and destination IP addresses and ports); content inspection rule consist of a string or regular expressions pattern that has to be matched against the packet payload. While packet header matching requires classification techniques that can be implemented using Ternary Content Addressable Memories (TCAM), pattern matching requires deep packet inspection that involves scanning every byte of the packet payload. Traditionally, patterns have been specified as exact match strings. Naturally, due to their wide adoption and importance, several high speed and efficient string matching algorithms have been proposed recently. These often employ variants of the standard string matching algorithms such as Aho-Corasick [Aho75], Commentz-Walter [Walter79], and Wu-Manber [Wu94], and use a preprocessed data-structure to perform high-performance matching. Among these, Aho-Corasick has been adopted most widely.

3.1 Aho-Corasick Algorithm

One of the earliest and efficient algorithms in exact multi-pattern string matching is due to Aho-Corasick [Aho75]; a number of recent systems use this technique. The algorithm enables string matching in a time linear in the size of the input. Aho-Corasick builds a finite automaton from the strings, whose structure is similar to a trie, and encodes all the strings to be searched in multiple stages. The construction begins with an empty root node which represents no partial match; subsequently nodes are added for each character of the pattern to be matched, starting at the root node and going to the end of the pattern. The strings that share a common prefix also shares a corresponding set of parent nodes in the trie. Beyond this, there are two variants of Aho-Corasick: deterministic and non-deterministic. In the non-deterministic version, the state machine trie is traversed beginning at the root node and failure pointers are added from each node to the longest prefix of that node which also leads to a valid node in the trie. Figure 3, illustrates a simple example. There are four strings: phone, telephone, test, and elephant. The automaton consists of 25 nodes in total. The bold transitions are normal ones, while the dotted ones are failure transitions. The operation of this non-deterministic implementation is straightforward. For any given input character in any given state, the character is consumed if a normal transition for the character is present at the state; else the failure transition is taken. Due to the construction, whenever a failure transition is taken the current input character is not consumed, and the failure transitions are used recursively until the character is consumed with a normal transition. It is easy to show via amortized analysis that only two state traversals per character of input string are required to consume any given input data.

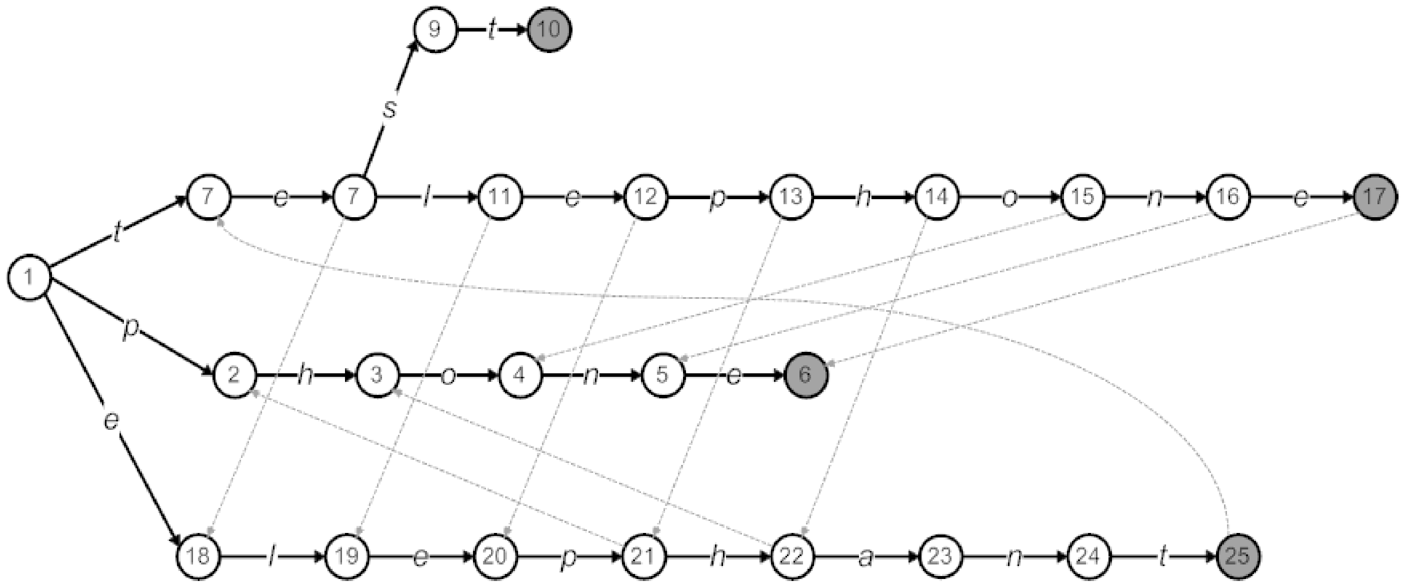

Figure 3: Aho-Corasick automaton for the four strings test, telephone, phone and elephant. Gray indicates accepting node. Dotted lines are failure transitions.

The deterministic version of Aho-Corasick automaton avoids the use of failure pointers in order to enable one traversal per input character. Instead of using failure pointers, next state from every state for every character in the alphabet is precomputed, and these transitions are added to the automaton. For an ASCII alphabet, such a construction results in 256 transitions at every state.

A large body of research literature has concentrated on enhancing the Aho-Corasick algorithm for use in NIDS. In [Tuck04], authors present techniques to enhance the worst-case performance of Aho-Corasick algorithm. Their algorithm was guided by the analogy between IP lookup and string matching and applies bitmap and path compression to Aho-Corasick. Their scheme has been shown to reduce the memory required for the string sets used in NIDS by up to a factor of 50 while improving performance by more than 30%. Many researchers have proposed high-speed Aho-Corasick based pattern matching hardware architectures. In [Tan05] authors propose an efficient algorithm that converts the deterministic version of Aho-Corasick automaton into multiple binary state machines. These state machines have significantly fewer transitions per state, which dramatically reduces the total space requirements. In [Sourdis04], the authors present an FPGA-based design which uses character pre-decoding coupled with CAM-based pattern matching.

3.2 Regular Expressions Signatures

In [Sommer03], authors note that regular expressions might prove to be fundamentally more efficient and flexible as compared to exact-match strings when specifying signatures for NIDS. The flexibility is due to the high degree of expressiveness achieved by using character classes, union, optional elements, and closures, while the efficiency is due to the effective schemes to perform pattern matching. Open source NIDS systems, such as Snort and Bro, today use regular expressions to specify rules. Regular expressions are also the language of choice in several commercial NIDS products, such as TippingPoint X505 [Tippingpoint] from 3Com and a family of network security appliances from Cisco Systems. Additionally, layer 7 filters based on regular expressions are available for the Linux operating system. Due to such wide adoption, we discuss the implementation of regular expressions signature in this sections in greater detail.

The most popular representation of regular expressions is finite state automata [Hopcroft79]. There are two primary kinds: Deterministic Finite Automaton (DFA) and Non-deterministic Finite Automaton (NFA). A DFA consists of an alphabet, which is a finite set of input symbols, a finite set of states, an initial state and a transition function, which specifies the transition from every state for every symbol in the alphabet. In networking applications, the alphabet generally consists of 256 ASCII characters. A key property of DFA is that in any given state, the transition function returns a single next state for any given input symbol; thus at any time, only one state is active in a DFA. The distinction between a NFA and a DFA lies in their transition function: instead of returning a single next state, the transition function of a NFA returns a set of states, which may be an empty set. Thus, multiple states can be simultaneously active in a NFA.

A regular expression containing n characters can be represented by a NFA consisting of O(n) states. During the execution of a NFA, O(n) states can be active in the worst case, and the processing complexity for a single input character can be O(n^2). When a DFA is constructed from the same regular expression, it may generate O(2^n) states in the worst-case. However, only one state will be active during execution, thereby leading to O(1) per character processing complexity. Clearly, there is a space-time tradeoff between NFA and DFA. NFAs are compact but slow; DFAs are fast but may require prohibitive amounts of memory. Current implementations of regular expressions patterns used in networking require gigabytes of memory, and their performance remains limited to sub-gigabit parsing rates; which makes it an important and challenging research area.

In order to enable high parse rates, several researchers have proposed specialized hardware-based architectures which implement finite automata using fast on-chip logic. Implementing regular expressions in custom hardware was explored in [Floyd82], in which authors showed that an NFA can be efficiently implemented using a programmable logic array. Sindhu et al. [Sidhu01] and Clark et al. [Clark03] have implemented NFAs on FPGA devices to perform regular expressions matching and were able to achieve very good space efficiency. In [Moscola03], authors have used such forms of NFAs that reduce the total number of simultaneously active states and demonstrated significant improvement in the data throughput.

These hardware based implementation approaches have two common characteristics: 1) due to the limited amount on-chip storage, they use NFA to keep the total number of states small, and 2) they exploit a high degree of parallelism by encoding automata in the parallel logic resources. These design choices are guided partly by the high degree of computation parallelism available on FPGA/ASIC and partly by the desire to achieve high throughput as such levels of throughput might be difficult to achieve in systems that store automata in memory. While such choices seem promising for FPGA devices, they might not be acceptable as the expression sets need to be updated frequently in today's NIDS. It might prove difficult to quickly re-synthesize and update the regular expressions circuitry on a NIDS in deployment. Therefore, regular expression engines which use memory rather than logic, are often more desirable as they provide higher degree of flexibility and programmability.

Commercial content inspection engines like Tarari's RegEx [Tarari] already emphasize the ease of programmability provided by a dense multiprocessor architecture coupled to a memory. Content inspection engines from other vendors also use memory-based architectures and report packet scan rates up to 4 Gbps. In this solution space, the transitions of the automaton are stored in memory in a tabular form, which is addressed with the state number and the input symbol. Every state traversal requires at least one memory access. Consequently, it becomes critical to keep as few active states as possible in order to limit the number of memory accesses and maintain a high parse rate. DFAs are therefore preferred over NFA; however a large number of complex regular expressions often creates DFA with exponentially large number of states. Rather than constructing a composite DFA from the entire regular expressions set, [Yu06] have proposed to partition the set into a small number of subgroups, such that the overall space needed by the automata is reduced dramatically. The proposed partitioning method keeps the number of DFAs small while containing the exponential blowup in the number of states. They also propose architectures to implement the grouped regular expressions on both general-purpose processor and multi-core processor systems, and demonstrate an improvement in throughput of up to 4 times. In order to further boost the parsing performance, it is possible to construct DFAs that can consume multiple characters during a single state traversal. However, for the NIDS signatures, such DFAs tend to have exponential number of states in the number of input characters that are consumed at once.

3.3 Architecture of Signature based NIDS

Once a high speed pattern matching architecture has been devised, the other components of NIDS can be built around it. The block schematic diagram of a reference NIDS is shown in Figure 4. The protocol analyzer block reassembles a TCP stream, because packets within a TCP streams can arrive out of order, or can be duplicated. Moreover, packets within a TCP flow at a high speed links get multiplexed with packets from other flows, therefore a storage component is required to store the state of a TCP connection upon multiplexing. In high speed systems, the total number of active TCP connection can reach to a million, therefore the storage component can become a substantial cost driver. Most NIDS are designed only to produce alarms. However, some commercially available NIDS such as from Cisco Systems provides a fair amount of countermeasure capability (illustrated by the countermeasure box), which ranges from shutting down the malicious TCP connection to modifying the router database and/or filter list. Such a capability enables the NIDS to quickly prevent attacks as soon as an initial event is detected, without having to wait for the human intervention. Even systems that don't provide such capabilities can be hooked into certain specific applications to achieve a similar effect.

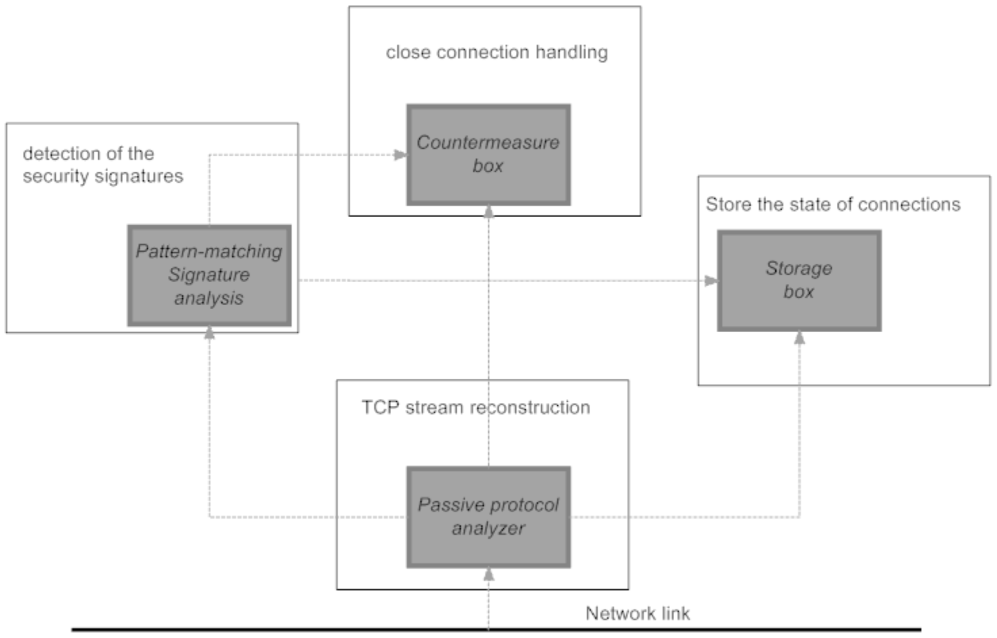
close connection handling
Countermeasure box
detection of the security signatures
Pattern-matching Signature analysis
Store the state of connections
Storage box
TCP stream reconstruction
Passive protocol analyzer
Network link

Figure 4: Block schematic of the components in a signature based NIDS.

Some of the well known NIDS that employ such architecture are listed below.

Snort:
Snort is the most popular open source NIDS. The capabilities that Snort offers as an intrusion detection and prevention system have made it the most flexible, powerful, and price conscious solution Snort has all of the technical capabilities that commercial IDS and IPS tools have. Its main capabilities include stateful inspection, pattern matching through an advanced rules language, and protocol anomaly detection.

Bro:
Bro is a network IDS developed by the Lawrence Berkeley National Laboratory of the Department of Energy and is used quite heavily in federal, military and research labs. Bro is an open source, Unix-based NIDS that passively monitors network traffic and looks for anomalous traffic behavior.

Tipping Point:
A product of 3Com, TippingPoint NIDS is the current commercial leader. The TippingPoint intrusion detection and prevention systems are an in-line device that can be inserted seamlessly and transparently at any location within a network. As packets pass through the device, their payload is fully inspected and matched against the signatures to determine whether they are malicious or legitimate. These products can support gigabit per second throughput with complete stateful packet inspection.

Back to Table of Contents

4. Anomaly Detection based NIDS

With the description of signature based NIDS, we now focus on anomaly detection for NIDS. Although not yet commercially available, these have been hailed as the future of the NIDS design. The key to the value and effectiveness of anomaly based NIDS is that they can automatically infer attacks which are yet unknown, and therefore undetectable by signature based NIDS. An anomaly detection technique generally consists of two different steps: the first step is called training phase wherein a normal traffic profile is generated; the second phase is called anomaly detection, wherein the learned profile is applied to the current traffic to look for any deviations. A number of anomaly detection mechanisms has been proposed recently to detect such deviations, which cam be categorized into statistical methods, data-mining methods and machine learning based methods. We present a brief description of each of them, and introduce some well known and recent algorithms in each category.

4.1 Statistical Anomaly Detection

A large number of statistical schemes assume that an anomaly will result in the deviation of certain traffic characteristics from normal, in terms of the volume (number of bytes, packets, a certain set of IP addresses or ports) [Barford02]. Such volume based schemes are successful in identifying large traffic changes such as bandwidth flooding attacks.

A number of alternative schemes argue that volume based schemes might not be effective if the attacker is smart enough to keep the disruptions caused by the attacks below certain levels. For example, an attacker can simply reduce the rate at which it is scanning ports, thereby keeping the traffic volume more or less unaffected. Therefore a number of algorithms aim at detecting fine changes in the behavior of traffic and/or the relative distributions of various traffic characteristics. Authors in [Lakhina05] have proposed to use entropy as a tool to summarize various traffic features. They show that the analysis of the traffic feature distributions can lead to sophisticated and fairly accurate detection mechanism. It will enable a highly sensitive detection of a wide range of anomalies, which will augment the detections made by the volume-based methods. Authors in [Kim04] [Kohler02] propose to use address correlation properties to detect anomalies. Such a scheme examines the packet headers rather than the packet payload, to look for the correlation between various header fields using the wavelet analysis.

Statistical anomaly detection engines can be added to the signature based systems, in order to automatically detect unknown attacks and possible generate a signature. SPADE (Statistical Packet Anomaly Detection Engine) is one such system than can be added to the Snort, however the current version lead to very high false positive rates.

4.2 Machine Learning to Detect Anomalies

Machine learning is an algorithmic method wherein an application automatically learns from the input and the feedbacks to improve its performance over time. Unlike statistical methods, which aims at determining the deviations in traffic features, machine learning based methods aims at detecting anomalies using some mechanism, and then based upon false positive or not, improving the mechanism.

One of the widely used such method is based upon Bayesian network model. Bayesian network is a graphical model which assigns probabilistic relationship between various variables of interest. Such a model can determine interdependencies between variables in the events of small data loss; additionally they can also predict the future interdependencies. In [Valdes00], the authors have applied Bayesian networks to detect anomalies on burst of traffic. They have demonstrated that the resulting system can detect DDoS attacks that will otherwise go undetected if each of the attack components is examined separately. Bayesian networks have also been used to aggregate and suppress alarms which eases the life of the administrators. In [Kruegel03], the authors propose a multi-sensor Fusion approach where the outputs collected from different sensors are uniquely aggregated to produce a single alarm.

4.3 Data Mining Algorithms to Detect Anomalies

Data mining consists of an advanced set of techniques, that essentially takes a set of data as input and detects the patterns and deviations which is otherwise difficult to detect. Thus, it becomes natural choice to not only detect anomalies, but also to construct the profiles of normal traffic. A number of data mining techniques have been applied.

Fuzzy logic algorithms have been employed to use a set of fuzzy sets and rules. In FIRE [Dickerson00], the authors propose to use relatively simple data mining techniques to process the network data and generate a set of fuzzy rules for every feature under observation, that will detect individual attacks based on each of the features. FIRE fails to establish any standard model that will represent the current system state; it rather relies on the attack specific rules for the detection. Genetic algorithms, which find approximate solutions to the optimization and search problems, have also been used in anomaly detection. These algorithms are often used to specific features to detect deviations from the normal profile, and based upon the false positive responses, they are also used to fine tune the parameters. Clustering has also been employed to detect anomalies. Clustering is a detection technique to find patterns in data with multiple dimensions. Clustering based system significantly cuts on the amount of training information that must be fed in order to detect anomalies. Some well known clustering based algorithms are MINDS [Ertoz04], and [Ramaswamy00].

With the brief description of both signature- and pattern based NIDS, we are now ready to discuss the benefits and drawbacks of each of them.

Back to Table of Contents

5. Strengths and Limitations of NIDS

NIDS today have become extremely valuable in enhancing the security of the networks and end hosts; they however have a number of key drawbacks. In the deployment of NIDS, it therefore is important for the network administrator to be aware of its strengths and limitations. In the section we discuss these properties.

5.1 Strengths of NIDS

NIDS can perform the following functions to enhance the security.

(1) Measurements and analysis of typical and atypical user behavior [Estan03]. For example an anomaly based NIDS is capable of detecting high volume traffic flows, flash crowds, load imbalance in the network, sudden changes in demand of a port usage, sudden surge of traffic from/to a specific host, etc [Lakhina05].

(2) Detection of known worms, viruses, and exploitation of a known security hole. Signature based NIDS can detect these events with fairly high degree of accuracy. An appropriate signature will also ensure a low false positive probability.

(3) Some advanced NIDS systems also enable recognitions of patterns of system events that correspond to a known security threat [OSHBIDS07].

(4) Enforcement of the security policies in a given network. For example a NIDS can be configured to block all communication between certain sets of IP addresses and or ports. A NIDS can also be used to enforce network wide access controls.

(5) Anomaly based NIDS can also recognize, with a certain false positive probability, new attacks and abnormal patterns in the network traffic, whose signatures are not yet generated. This will alert the network administrator early, and potentially reduce the damage caused by the new attack.

5.2 Limitations of NIDS

(1) A mere Workaround:
A number of researchers have argued that a NIDS is more or a less a workaround for the flaws and weak or missing security mechanisms in an operating system, an application, and/or a protocol [Bace01].
(2) False Positives:
NIDS comes with a bane, i.e. false positives. A false positive is an event when a NIDS falsely raises a security threat alarm for harmless traffic. Signatures can be tuned precisely to reduce such false positives, however fine signatures create a significant performance bottleneck, which is the next limitation of NIDS. Current Anomaly based algorithms lead to even higher false positives [Kim04] [Lakhina05].
(3) Performance issues:
Current signature based NIDS systems use regular expressions signatures which creates a significant performance bottleneck. In order to reduce false positives long signatures are required which further reduces the performance. The data throughput of current NIDS systems is limited to a few gigabit per second [Kumar05] [Yu06].

(4) Encryption:
The ultimate threat to the very existence of the signature based NIDS systems is the increasing use of data encryption. Everybody dreams to encrypt their data before transmission. Once the packet payloads are encrypted, the existing signatures will become completely useless in identifying the anomalous and harmful traffic [Tanase02].
(5) New and sophisticated attacks:
Commercial NIDS which are signature based are unable to detect new attacks whose signatures are not yet devised. Anomaly based NIDS can detect such attacks but due to the limitations of the current anomaly detection algorithms, an intelligent attacker can always develop attacks that remain undetected.
(6) Human intervention:
Almost all NIDS systems require a constant human supervision, which slows down the detection and the associated actions. Some recent systems such as Network Intrusion Prevention Systems (NIPS) [Cisco] can automatically take pre-programmed actions but these are limited only to the well known attacks.
(7) Evasion of signatures:
A number of researchers have argued that it is not difficult for an attacker to evade a signature [Varghese06]. Additionally there has been an increase in polymorphic worms [Kolesnikov04] [Newsome05] which can automatically change their propagation characteristics thereby effectively changing their signatures. Such worms also pose a critical threat to the current NIDS.

Back to Table of Contents

6. Common Attacks and Vulnerabilities and Role of NIDS

Current NIDSs requires substantial amount of human intervention and administrators for an effective operation. Therefore it becomes important for the network administrators to understand the architecture of NIDS, and the well known attacks and the mechanisms used to detect them and contain the damages. In this section, we discuss some well known attacks, exploits, and vulnerabilities in the end host operating systems, and protocols.

6.1 Attack Types

Confidentiality: In such kinds of attacks, the attacker gains access to confidential and otherwise inaccessible data.

Integrity:
In such kinds of attacks, the attacker can modify the system state and alter the data without proper authorization from the owner.

Availability:
In such kinds of attacks, the system is either shut down by the attacker or made unavailable to general users. Denial of Service attacks fall into this category.

Control:
In such attacks the attacker gains full control of the system and can alter the access privileges of the system thereby potentially triggering all of the above three attacks.

6.2 Attacks detected by a NIDS

A number of attacks can be detected by current generation of NIDS. Some of these are listed and described below.

6.2.1 Scanning Attack

In such attacks, an attacker sends various kinds of packets to probe a system or network for vulnerability that can be exploited. When probe packets are sent the target system responds; the responses are analyzed to determine the characteristics of the target system and if there are vulnerabilities (illustrated in Figure 5). Thus scanning attack essentially identifies a potential victim. Network scanners, port scanners, vulnerability scanners, etc are used which yields these information:

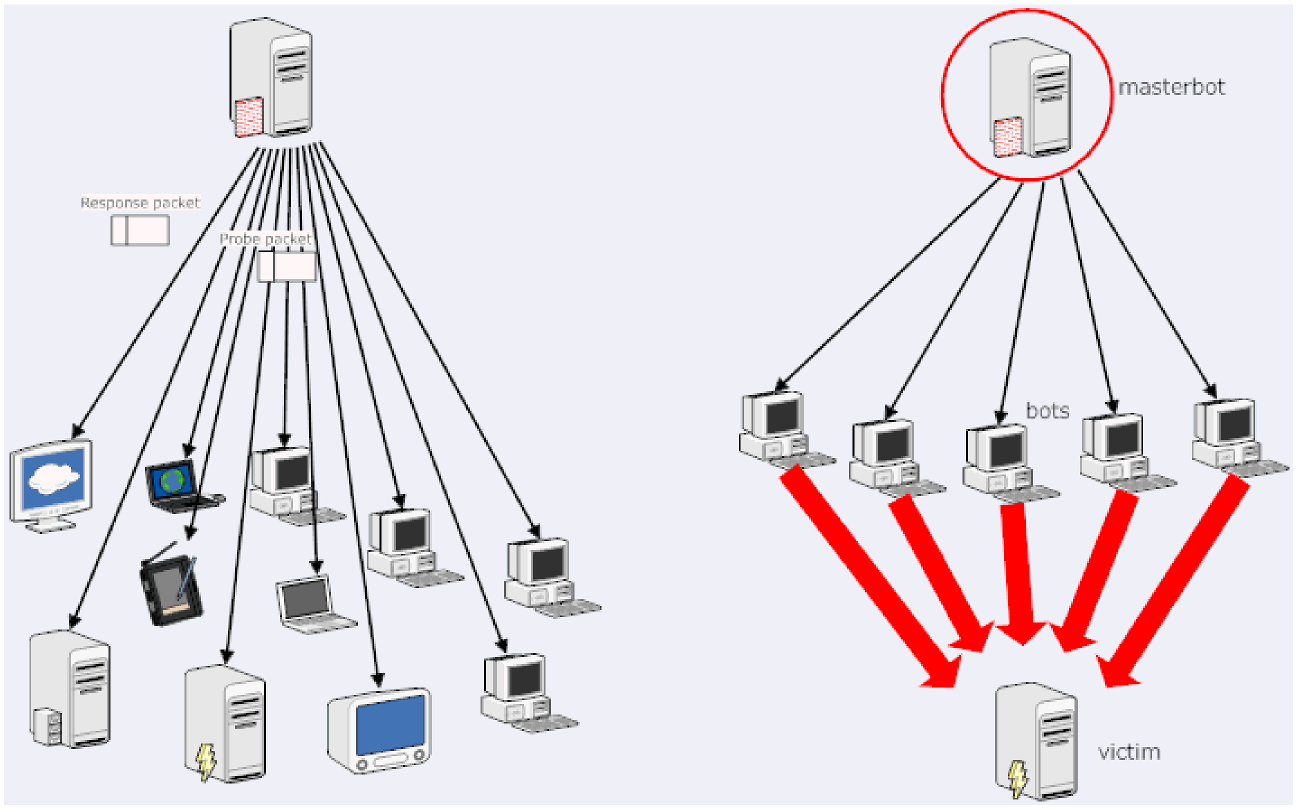

Figure 5: Left diagram shows a scanning attack where a single attack host scans a number of victims. Right diagram shows a denial of service attack (DDoS in this case), wherein an attacker uses a number of compromised hosts to attack a given victim.

The network topology.
The type of firewall used by the system.
The identification of hosts that are responding.
The software, operating systems and server applications that are currently running.
Vulnerabilities in the system.

Once the victim is identified, the attacker can penetrate them in a specific way. Scanning is typically considered a legal activity and there are a number of examples and applications that employ scanning. The most well known scanning applications are Web search engines. On the other hand independent individual ay scan a network or the entire Internet looking for certain information, such as a music or video file. Some well-known malicious scanning include Vertical and Horizontal port scanning, ICMP (ping) scanning, very slow scan, scanning from multiple ports and scanning of multiple IP addresses and ports. NIDS signatures can be devised to identify such malicious scanning activity from a legitimate scanning activity with fairly high degree of accuracy [Lam05].

6.2.2 Denial of Service (DoS) Attacks

A Denial of Service attack attempts to slow down or completely shut down a target so as to disrupt the service and deny the legitimate and authorized users an access [Mirkovic05]. Such attacks are very common in the Internet where a collection of hosts are often used to bombard web servers with dummy requests (illustrated in Figure 5). Such attacks can cause significant economic damage to ecommerce businesses by denying the customers an access to the business. There are a number of different kinds of DoS attacks, some of which are mentioned below.

6.2.2.1 Flaw Exploitation DoS Attacks

In such attacks, an attacker exploits a flaw in the server software to either slow it down or exhaust it of certain resources. Ping of death attack is one such well known attack. A ping of death (POD) is a type of attack on a computer that involves sending a malformed or otherwise malicious ping to a computer. A ping is normally 64 bytes in size (or 84 bytes when IP header is considered); many computer systems cannot handle a ping larger than the maximum IP packet size, which is

65,535 bytes. Sending a ping of this size can crash the target computer. Some limitations of the protocol implementation also lead to vulnerability which can be exploited to implement DoS attacks such as DNS amplification attack [Randal06], which uses ICMP echo messages to bombard a target. For these attacks, a signature can be devised easily, such as to determine a ping of death attack a NIDS needs to check the ping flag and packet size.

6.2.2.2 Flooding DoS Attacks

In a flooding attack, an attacker simply sends more requests to a target that it can handle. Such attacks can either exhaust the processing capability of the target or exhaust the network bandwidth of the target, either way leading to a denial of service to other users. DoS attacks are extremely difficult to combat, as these do not exploit any vulnerability in the system, and even an otherwise secure system can be targeted. A more dangerous version of DoS attack is called Distributed Denial of Service attack (DDoS), which uses a large pool of hosts to target a given victim host. A hacker (called botmaster) can initiate a DDoS attack by exploiting vulnerability in some computer system, thereby taking control of it and making this the DDoS master (Figure 5). Afterwards the intruder uses this master to communicate with the other systems (called bots) that can be compromised. Once a significant number of hosts are compromised, with a single command, the intruder can instruct them to launch a variety of flood attacks against a specified target [FBI05].

6.2.3 Penetration Attacks

In penetration attack, an attacker gains an unauthorized control of a system, and can modify/alter system state, read files, etc. Generally such attacks exploit certain flaws in the software, which enables the attacker to install viruses, and malware in the system. The most common types of penetration attacks are:

User to root: A local user gets the full access to every component of the system.

Remote to user: A user across the network gains a user account and the associated controls.

Remote to root: A user across the network gains the complete control of the system.

Remote disk read: An attacker on the network gains access to the inaccessible files stored locally on the host.

Remote disk write:
An attacker on the network not only gains access to the inaccessible files stored locally on the host, but can also alter them.

6.3 Role of NIDS in Combating Attacks

A NIDS can detect attacks, and anomalous conditions, additionally they can also provide a number of key information which can be used to identify the nature of attack, its origin and propagation characteristics. First and foremost, most NIDS often reports the location of the attacker or hacker (from where the attack has been triggered). However, the location is commonly expressed as an IP address, which is not reliable information, as the smart attackers often change the IP address in the attack packets, which is called IP address spoofing.

The key to determine the importance of the source IP address reported by the NIDS is to classify the attack and then determine if the attack requires the reply messages to be seen or not. In attacks where reply packets are required, IP source address spoofing can not be done. In attacks such as a one way DoS flooding attack, the attacker need not examine the reply, and can easily spoof its address. However, Modern NIDS can also report the route that the attack packets have taken. The route information contains key pieces that can be used to trace the hacker in spite of the source address spoofing. A large variety of attacks such as scanning attacks and penetration attacks, etc requires the attacker to examine the reply messages, in which case tracing them becomes much easier.

6.3.1 Excessive Attack Reporting

Today, an NIDS serving large enterprise network reports a significant number of attacks, which often overwhelms the operators. It often becomes impossible to manually examine each of these reports. The problem is because some NIDS raises alarm whenever a misuse is detected, wherein an attacker attempts to access a host. Modern NIDS therefore have started to include the capability to aggregate these reports into a smaller number of subsets that is much easier to examine

[Lakhina05]. They also classify the attacks into different levels of threats and present the most serious threats to the operator first.

In fact, when the signatures of an attack are designed, a security level is attached to them. The security level depends not only upon the seriousness of attack, but also upon the accuracy of the signatures. For example, if a signature is such that it leads to more than 90% false positives, then even though the threat is serious, it may accompany a low level of threat, in order to ease the task of the administrator. Nonetheless, these security levels are highly subjective and configurable and an operator can easily alter them based upon his needs.

6.4 Computer Vulnerabilities and NIDS

Since most of the attacks exploit a known or unknown vulnerability in the computer, a NIDS usually reports the type of vulnerability that an attacker is trying to exploit. Such information is important in keeping the system up to date, fixing the bugs and eliminating the vulnerability. Here we discuss some of the well known vulnerability which has been exploited in the past.

6.4.1 Buffer Overflow

A buffer overflow (also referred as buffer overrun) is a programming bug that can lead to illegal program termination or memory access exception. Such vulnerability has been exploited a number of times to breach of system security, including the Morris worm, the Code Red worm and the SQL Slammer worm. Most recently, buffer overflows present in licensed Xbox games have been exploited to allow unlicensed software, including homebrew games, to run on the console without the need for hardware modifications, known as modchips. Buffer overflow exploits are generally well fingerprinted and all known exploits have fairly accurate signatures.

6.4.2 Input Validation Error

In such vulnerability, the system does not check the input for integrity and correctness before processing them. This can lead to a number of exploits, wherein an attacker can send a specific sequence of inputs that will lead to either the failure of the system, or will give the attacker an unauthorized access. Again a NIDS can easily detect such events, and raise an alarm.

6.4.3 Boundary condition error

Boundary condition error is a form of input validation error where the input results in the system crossing some security boundary. For example the system may run out of memory, disk space, and network bandwidth. A simple example of such vulnerability is "divide by zero", wherein a careless implementation may lead to a crash. A NIDS can detect such conditions, and take appropriate actions.

6.4.4 Access control vulnerability

This might arise due to the faulty implementation of the access control. This may include giving an unauthorized access to a user, or providing illegitimate remote access between two separate network domains. A NIDS can effectively tackle such exploits by examining the IP and application level headers, and checking the source and destination hosts. Access control vulnerability may also arise due to configuration errors, which can also be detected with the appropriate NIDS signatures.

Back to Table of Contents

---

7. Future of NIDS

One of the key challenges with NIDS has traditionally been performance. Most NIDS employ deep packet inspection which limits the performance. However we have seen that a wide variety of high performance algorithms have been proposed recently, which enhances the performance. Current systems can easily scale to multi gigabit throughputs, and in future performance is likely to become less of an issue.

A more challenging problem is the analysis and correlation once an alarm has been raised. The central issue here is that this operation is performed by human, and no matter how skilled the NIDS analyst is, this operation will remain slow and error prone. In future, this step is going to be automated. Recently a number of new products have been delivered, dubbed as intrusion prevention system, which not only detects an attack but also takes an appropriate action upon certain attacks. In future, as the accuracy of detection will increase further, more and more automated response will be implemented with the NIDS.

The increasing use of encryption adds another dimension to the problem. It necessitates that the NIDS be placed on the hosts system where the data can be decrypted. However this will require coordination between the NIDS deployed at various hosts, leading to a distributed implementation of NIDS. At the center of this model will lie a familiar concept - the management station. This device will be similar to the NIDS consoles used in a multi-sensor context, and it will analyze the flagged traffic that has been reported potentially anomalous by a number of distributed client computers. Thus the central component will be solely responsible for analysis and correlation rather than traffic capturing and filtering from a diverse environment.

With the mounting security concerns, the future of IDS is surely promising; however it is important for the above model to emerge. Host machines need to aid the central NIDS component in looking for the behavior (network or system) that is malicious or abnormal. Current well known schemes such as signature based detection can be used here. Additionally, these clients can efficiently run sophisticated anomaly detection algorithms, as data rates over there will be relatively low.

The key challenge then remains in devising the algorithms that can detect anomalies with a fairly high degree of confidence. Although this is an active research topic, it still is questionable when such algorithms will be devised that can be used in a commercial setting. There is another aspect that requires attention in the future - standards concerning the NIDS reporting. In the immediate future, an NIDS protocol will be established and a standardized reporting format will become a requirement. A number of other standardization will likely occur once NIDS mechanisms become better understood and well implemented.

Back to Table of Contents

8. Summarizing NIDS

In this survey paper, we describe the design and architecture of a number of different NIDS and the various configurations, in which they are employed in the network. Specifically we focus on two important classes of NIDS: signature based and anomaly based. We thoroughly investigate their benefits and drawbacks, and discuss a number of attack and vulnerabilities than they can combat. Finally we discuss the future trends in this space, where we argue that a more distributed version of NIDS is on the horizon and that the NIDS mechanisms need to be standardized.

Back to Table of Contents

References

[Paxson03]	[1] R. Sommer, V. Paxson, , "Enhancing Byte-Level Network Intrusion Detection Signatures with Context," ACM conf. on Computer and Communication Security, 2003, pp. 262--271. citeseer.ist.psu.edu/sommer03enhancing.html
[Lakhina05]	[2] A. Lakhina, et al., "Mining Anomalies Using Traffic Feature Distributions," Proc. ACM SIGCOMM 2005. www.sigcomm.org/sigcomm2005/paper-LakCro.pdf
[Estan03]	[3] C. Estan, S. Savage, and G. Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic," In ACM SIGCOMM, Karlsruhe, August 2003. www.sigcomm.org/sigcomm2003/papers/p137-estan.pdf
[OSHIDS07]	[4] Open Source Host-based intrusion detection system, 2007. http://www.ossec.net/

[Kim04] [5] S. Kim, A. L. N. Reddy, and M. Vannucci, "Detecting Traffic Anomalies through Aggregate Analysis of Packet Header Data," In Networking, 2004. www.ece.tamu.edu/~reddy/papers/skim_net04.pdf

[Bace01] [6] Rebecca Bace and Peter Mell, "NIST Special Publication on Intrusion Detection Systems," 16 August 2001. csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

[Tanase02] [7] Matthew Tanase, The Future of IDS, 2002. http://www.securityfocus.com/infocus/1518

[Varghese06] [8] G. Varghese, A. Fingethut, and F. Bonomi, "Detecting Evasion Attacks at High Speeds without Reassembly," Proc. ACM SIGCOMM, 2006. portal.acm.org/citation.cfm?id=1159951

[Kolesnikov04] [9] Oleg Kolesnikov, and Wenke Lee, "Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic," 2004. citeseer.ist.psu.edu/678163.html

[Newsome05] [10] J. Newsome, B. Karp, D. Song, "Polygraph: automatically generating signatures for polymorphic worms," Proc. IEEE Security and Privacy, 2005. www.cs.berkeley.edu/~dawnsong/papers/polygraph.pdf

[Lam05] [11] Alex Lam, "New IPS to Boost Security, Reliability and Performance of the Campus Network," Newsletter of Computing Services Center, 2005. http://www.cityu.edu.hk/csc/netcomp/dec2006-5.htm

[Vaughn07] [12] Vaughn, Randal and Evron, Gadi (2006), "DNS Amplification Attacks," March 17, 2007. www.isotf.org/news/DNS-Amplification-Attacks.pdf

[Mirkovic05] [13] Internet Denial of Service: Attack and Defense Mechanisms, by Jelena Mirkovic, Sven Dietrich, David Dittrich and Peter Reiher, Prentice Hall PTR, ISBN 0131475738, 2005. safari.phptr.com/0131475738

[FBI05] [14] FBI agents bust 'Botmaster', Reuters News Service, November 4, 2005 www.webmasterworld.com/forum9/9538.htm

[Thakker03] Dhawal Thakker, Choosing the right intrusion detection system, 2003. http://www.expresscomputeronline.com/20030929/security16.shtml

[Barford02] [16] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," In Internet Measurement Workshop, Marseille, November 2002. pages.cs.wisc.edu/~pb/paper_imw_02.pdf

[Kumar05] [17] S. Kumar et al., "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," Proc. ACM SIGCOMM, 2005. portal.acm.org/citation.cfm?id=1159952

[Kohler02] [18] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed Structure of Addresses in IP Traffic," In Internet Measurement Workshop, Marseille, November 2002. www.imconf.net/imw-2002/imw2002-papers/189.pdf

[Kruegel] [19] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian Event Classification for Intrusion Detection," in 19th Annual Computer Security Applications Conference, Las Vegas, NV, 2003. www.cs.ucsb.edu/~wkr/publications/acsac03bayes.pdf

[Valdes00] [20] A. Valdes and K. Skinner, "Adaptive Model-based Monitoring for Cyber Attack Detection," in Recent Advances in Intrusion Detection Toulouse, France, 2000, pp. 80--92. citeseer.ist.psu.edu/672825.html

[Dickerson00] [21] J. E. Dickerson and J. A. Dickerson, "Fuzzy network profiling for intrusion detection," in 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS), Atlanta, GA 2000. home.eng.iastate.edu/~julied/publications/NAFIPSpaper2000.pdf

[Ramaswamy00] [22] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," in ACM SIGMOD international conference on Management of data, Dallas, TX, USA, 2000, pp. 427 - 438. citeseer.ist.psu.edu/512378.html

[Ertoz04] [23] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas, "The MINDS - Minnesota Intrusion Detection System," in Next Generation Data Mining Boston: MIT Press, 2004. www.cs.umn.edu/research/MINDS

[Hopcroft79] [24] J. E. Hopcroft and J. D. Ullman, "Introduction to Automata Theory, Languages, and Computation," Addison Wesley, 1979.

[Bro] [25] Bro: A System for Detecting Network Intruders in Real-Time. http://www.icir.org/vern/bro-info.html

[Roesch99]	[26] M. Roesch, "Snort: Lightweight intrusion detection for networks," In Proc. 13th Systems Administration Conference (LISA), USENIX Association, November 1999, pp 229-238. www.snort.org/
[Aho75]	[27] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," Comm. of the ACM, 18(6):333-340, 1975. portal.acm.org/citation.cfm?id=360825.360855
[Walter79]	[28] B. Commentz-Walter, "A string matching algorithm fast on the average," Proc. of ICALP, pages 118-132, July 1979. portal.acm.org/citation.cfm?id=682242
[Wu94]	[29] S. Wu, U. Manber," A fast algorithm for multi-pattern searching," Tech. R. TR-94-17, Dept. of Comp. Science, Univ of Arizona, 1994. citeseer.ist.psu.edu/wu94fast.html
[Yu05]	[30] Fang Yu, et al., "Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection", UCB tech. report, EECS-2005-8. www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-76.pdf
[Tuck04]	[31] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," IEEE Infocom 2004, pp. 333--340. citeseer.ist.psu.edu/tuck04deterministic.html
[Tan05]	[32] L. Tan, and T. Sherwood, "A High Throughput String Matching Architecture for Intrusion Detection and Prevention," ISCA 2005. portal.acm.org/citation.cfm?id=1069807.1069981
[Tippingpoint]	[33] TippingPoint X505. www.tippingpoint.com/products_ips.html
[Cisco]	[34] Cisco IOS IPS Deployment Guide. www.cisco.com
[Tarari]	[35] Tarari RegEx. www.tarari.com/PDF/RegEx_FACT_SHEET.pdf

Back to Table of Contents

List of Acronyms

ASIC	Application Specific Integrated Circuit
DDOS	Distributed Denial of Service Attack
DFA	Deterministic Finite Automaton
DNS	Domain Name System
DOS	Denial of Service Attack
FIRE	Fuzzy Logic Rule Engine
FPGA	Field Programmable Gate Array
FTP	File Transmission Protocol
ICMP	Internet Control Message Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
MIND	Minnesota Network Intrusion Detection
NFA	Non-Deterministic Finite Automaton
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention System
OSHBIDS	Open Source Host-based intrusion detection system
POD	Ping of Death
SPADE	Statistical Packet Anomaly Detection Engine
TCAM	Ternary Content Addressable Memories
TCP	Transmission Control Protocol

Back to Table of Contents

Last Modified: December, 2007

This paper is available at: http://www.cse.wustl.edu/~jain/index.html