White Paper

Intrusion Detection and Prevention

Protecting Your Network From Attacks

Sarah Sorensen
Product Marketing Manager

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
408 745 2000 or 888 JUNIPER
www.juniper.net

Part Number: 200065-002

Contents

Is Your Network Protected Against Attacks?

The landscape of doing business today is significantly different from the landscape of just five years ago. Companies are more connected than ever, with the promise that network expansion will only continue. As a result, companies must grapple with how to keep their network safe, without sacrificing growth or productivity.

The first step that virtually all organizations connected to the Internet take is to install a firewall, and with good reason. A firewall acts as a perimeter guard for a network, determining what traffic to allow or deny in and out. A firewall does this by applying a policy, comprised of "accept" and "deny" rules, based on various criteria, such as a source, destination and protocol in question. By providing access control, firewalls do a good job of providing the first layer of defense. Most firewall polices allow protocols that enable organizations to do business on the Internet, such as SMTP, FTP, HTTP, SMTP and DNS, and keep out traffic that may pose a threat to the internal systems.

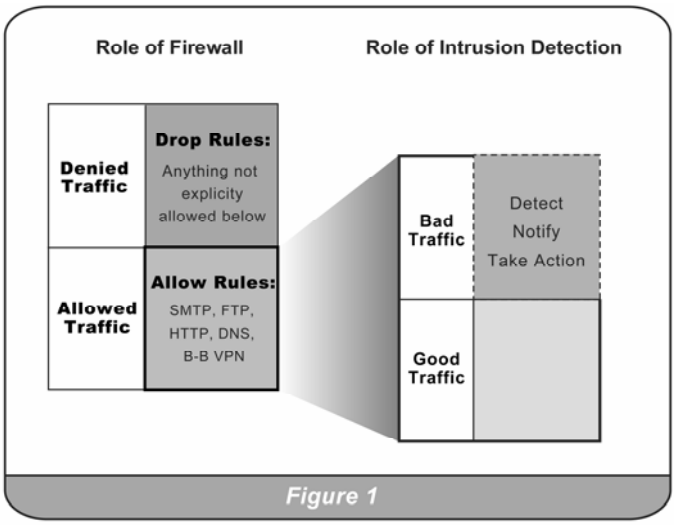


Figure 1

The second layer of defense is to detect the presence of attacks within the traffic allowed to flow within your network and to protect your network from those attacks. The most common belief today is that a passive network intrusion detection system (NIDS) can protect an organization from these attacks. Unfortunately, this belief is misplaced for the following reasons:

- **False Alarms**: Many NIDS solutions produce inaccurate results, due to their limited and poorly implemented intrusion detection mechanisms. This manifests itself in large quantities of false alarms that often overwhelm system administrators, requiring manual filtering to identify the real attacks among the false alarms. As a result, many companies ultimately ignore the information, rendering the system useless.
- **Low Manageability, High Maintenance**: Current NIDS solutions are notorious for being hard to manage and maintain, requiring a lot of time and effort to keep the sensors updated and the security policy in force.
- **Perceived Need to Outsource**: Many companies feel that if they added an NIDS to their system they would have to outsource its maintenance to a managed security service provider to derive value.
- **No Prevention of Attacks**: Current NIDS solutions do not prevent attacks. Although advertised with prevention capabilities, these products are merely detection products, with prevention mechanisms delivered as empty promises.

This white paper presents information about technological advancements that overcome these issues. It strives to demonstrate how, when implemented properly, an intrusion detection device can provide a powerful and cost-effective solution that complements a firewall in protecting your corporate assets. The technological advancements include increased intrusion detection accuracy and the ability to prevent an intrusion, all while simplifying the deployment, configuration and ongoing management of the system. It is found in Juniper Networks' NIDS, where the prevention mechanisms are built into the product as a design objective. The Juniper Networks NetScreen-IDP™ (Intrusion Detection and Prevention) product line not only accurately detects intrusions, but also takes the next logical step, namely giving corporations the ability to prevent intrusions from causing damage to network resources.

After reading this white paper, you will learn the following:

- How various intrusion detection mechanisms work, and why it's important to implement a combination of these mechanisms.
- Why the most widely known intrusion detection mechanism, packet signatures, is outdated, and how a system based on Stateful Signatures™ provides faster intrusion detection with increased accuracy.
- Why an intrusion detection and prevention solution must operate in-line to avoid being evaded and provide complete protection for your network.
- Why granular control options are necessary to maximize the multiple detection and response mechanisms.

Intrusion Detection and Prevention Requirements

What an NIDS Can Detect and Miss

Every network has bad (malicious) traffic in it. It may be someone outside trying to gain information or a disgruntled employee on the inside looking to cause havoc. Whomever the culprit, you want to know and do something about it. It is the job of an NIDS to tell you about the attack, so that you can keep it from impacting your business. The system, however, is only as good as its detection capabilities. As a result, it is critical that the system's detection mechanisms are accurate enough to differentiate between the good and bad traffic that gets into your network. The following are all possible results of intrusion detection (see Figure 2):

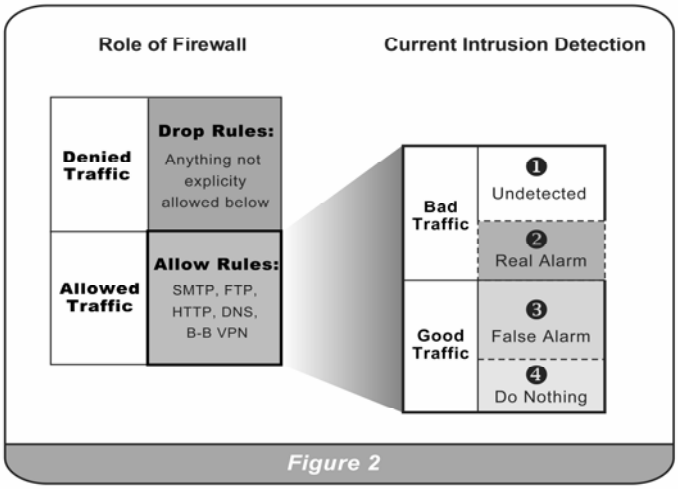


Figure 2

1. **Undetected bad traffic**
2. **Detected bad traffic**
3. **Good traffic that the system thinks is bad (false alarm)**
4. **Good traffic that the system identifies as good**

1. **In Bad Traffic: Failure to identify malicious traffic as an attack**.
   The worst thing that can happen because it means the intrusion detection system failed to do its job. Failing to detect an attack can occur when an NIDS does not have adequate or comprehensive intrusion detection mechanisms in place. It also occurs when new attacks are created and then missed by poorly implemented detection mechanisms. While it is virtually impossible to detect every attack, the goal of any system should be to minimize the number of undetected attacks.
2. **In Bad Traffic: Identifying "real attacks" as an attack.**
   An ideal result of an intrusion detection system. The ability to detect bad traffic with speed and reliability is referred to as intrusion detection accuracy. All other functions of the system hinge on this capability. The more accurate the system, the more you can trust its abilities. A system must have proven accuracy before you enable it to take the necessary actions (such as dropping the connection) to secure your network.
3. **In Good Traffic: Identifying good traffic as an attack** (often called a false alarm or a false positive).
   The most troublesome and time-consuming aspect of current NIDS solutions on the market. It occurs when the NIDS sees something in legitimate and benign traffic that makes it believe there is an attack. It is detrimental because you need to investigate each and every alarm to determine whether an attack was successful and assess any resulting damage. Every moment spent investigating a false alarm reduces the time available to investigate real threats. The result is that false alarms can erode your trust in the product;

sometimes causing real attack alarms to be overlooked (the "crying wolf" effect). Most NIDSes can be tuned to try to reduce the occurrence of false alarms, however, the tuning process is often long and involved, sometimes taking weeks to accomplish. In addition, because of the management design of current NIDSes, tuning is often an all or nothing approach. This means that you must choose whether or not to look for a certain attack. If, in the interest of reducing false alarms, the detection of certain attacks is turned completely "off," those attacks will be able to go by the NIDS completely undetected. Lastly, lots of false alarms make it very difficult to reliably drop connections, which is the only way to truly prevent an attack (See Section 2.3).

4. **In Good Traffic: Identifying good traffic as good traffic.**
   An ideal result of intrusion detection mechanisms, identifying good traffic for what it is – good traffic.

Detection Techniques

The ideal performance on an NIDS is to identify as many attacks as possible and limit the number of false alarms. Unfortunately, there is no single detection mechanism available today that an NIDS can deploy to detect every type of network-based attack. As a result, only a system that combines a variety of technologies to detect different types of attacks can get the job done. Up until the introduction of Juniper's IDP system (See Section 3.1), the only way to implement a combination of detection methods was to purchase two or more products and run them side-by-side. The two most commonly available network intrusion detection mechanisms on the market today are signature detection and protocol anomaly detection. Signature-based systems look for known attack patterns (signatures) in traffic. Signature detection finds attacks for which a signature is written, but is unable to detect new attacks or many very complicated attacks. Protocol anomaly detection-based systems, on the other hand, do a good job of detecting some of the unknown attacks, but are unable to identify attacks that operate without violating any protocols. This is why a best of all worlds approach is required. The next section explains the technologies available and the capabilities of each method of detection.

Intrusion Detection Using Protocol Anomalies

Protocol anomaly detection, which is sometimes called protocol analysis, is the ability to analyze packet flows (the uni-directional communication between two systems) to identify irregularities in the generally accepted Internet rules of communication. These rules are defined by open-protocols and published standards (RFC's), as well as vendor-defined specifications for communication between networked devices. The objective is to implement an intrusion detection mechanism that identifies traffic that doesn't meet specifications or violates the relevant standards. Once an irregularity is identified, it can be used to make network security decisions. This is very effective in detecting suspicious activity, such as a buffer-overflow attack.

The advantages of protocol anomaly detection are that it can detect:

1. Unknown and new attacks, based on the fact that these attacks deviate from protocol standards
2. Attacks that bypass systems that implement other detection methods
3. Slightly modified attacks that change the format of known attack patterns, with no affect on the strength of the attack, to evade signature-based systems

Example 1: Detecting an FTP Bounce Attack

This example describes the FTP bounce attack and how it would be identified using protocol anomaly detection. The FTP bounce attack exploits a design flaw in the specifications of FTP (File Transfer Protocol). To download or upload files, a user (FTP client) must first connect to an FTP server (1). When this happens, the server requires the client to send the IP address and port number to which the file should be sent to or taken from. This is done via a mechanism called a "PORT Command." In practice, the IP address is that of the user, however, the PORT command specification does not limit the IP address to the user's address. Because of this, an attacker can tell the FTP server to open a connection to an IP address that is different from the user's address (2) and then use the open port to transfer files containing a Trojan through the FTP server onto the victim (3). Once this is accomplished, the attacker can access the victim and transfer files from the victim directly to the attacker's machine. To detect this attack, an NIDS needs to compare the requests in the PORT command with the IP address of the client. If they do not match, the NIDS needs to send an alarm. This cannot be achieved with signature matching, since the detection is not based on matching a specific text string (pattern) but rather on a relationship between two elements of a network protocol. Protocol anomaly detection, on the other hand, is designed to look at network relationships and determine whether they are acting within the normal specifications. By using protocol anomaly detection, an NIDS can parse the requests in a PORT command whenever seen and compare it to the IP address from which the PORT command arrived.

Example 2: Detecting an Undocumented Buffer Overflow Attack

This example describes a Buffer Overflow Attack and how it would be detected using protocol anomaly detection. These types of attacks exploit a common programming error that allows an unlimited amount of data to be read into a fixed-size memory buffer without checking for an overflow. As a result, the excess input data "spills" out of the buffer and goes unchecked by the application software. When this happens, arbitrary memory addresses are overwritten with excess data. If crafted correctly, the program will unwillingly execute the excess data. If successful, an attacker can run whatever they wish on the victim host.

Attackers would typically perpetrate this type of attack by trial and error. Usually they are armed with some information about how the system or application works and then systematically send data that is known to be out of range for the given system or application. At this point, since an attack pattern does not exist, the only way to detect this type of attack is to use protocol anomaly detection. This method can determine if data transmissions are abnormal and out of specification, constituting an attack.

The interesting phenomenon is that once a buffer overflow attack is discovered, publicized and characterized (either by a self-promoting attacker or as a known vulnerability), it can then be identified as an attack by a signature-based system. This is because when an attack is characterized, the steps required to perpetrate the attack are described, thereby creating an "attack pattern." This allows a signature to be written to "detect" the attack from that point forward.

Intrusion Detection Using Stateful Signatures

Intrusion detection that is based on recognizing and matching attack signatures (patterns) is very straightforward. Basically it entails looking for a particular pattern in traffic that has been characterized as a known exploit or vulnerability. The evolution of NIDSes started with the implementation of a non-intrusive packet monitor, called a sniffer because of its ability to "sniff" the packets on the network. Intrusion detection vendors applied the packet-monitoring concept to build systems that performed packet signature detection. This means that the systems looked at the information contained in the packet stream and compared it to a database of known attack signatures. There are many drawbacks to this simplistic approach to intrusion detection – especially if effort is placed entirely on building up a large repository of attack signatures, without regard to how the traffic is reassembled, decoded, normalized and analyzed.

Network Transmissions – What You See Is Not What You Get

When information is transmitted over the network, the information is split into numbered TCP (Transmission Control Protocol) segments that are sent as packets. In an ideal world, the packets would be transmitted in sequence and without loss (Figure 3a). But, unfortunately, that's not the case. When a message is actually transmitted (Figure 3b), the network will deliver the packets randomly (out of sequence) or as even smaller pieces of data (called fragments), which are broken down by networking devices, such as routers, to facilitate ease of transmission. Even worse, for whatever reason, packets can get "lost." The receiving system is responsible for reconstructing the packets into a stream of information and requesting the retransmission of the missing packets, so that the entire message can be presented to the application as a whole. Figure 3a and 3b only represent a single flow from the client to the server; the reality is that most client to server communications consist of two flows, one from the client to the server and another from the server back to the client.

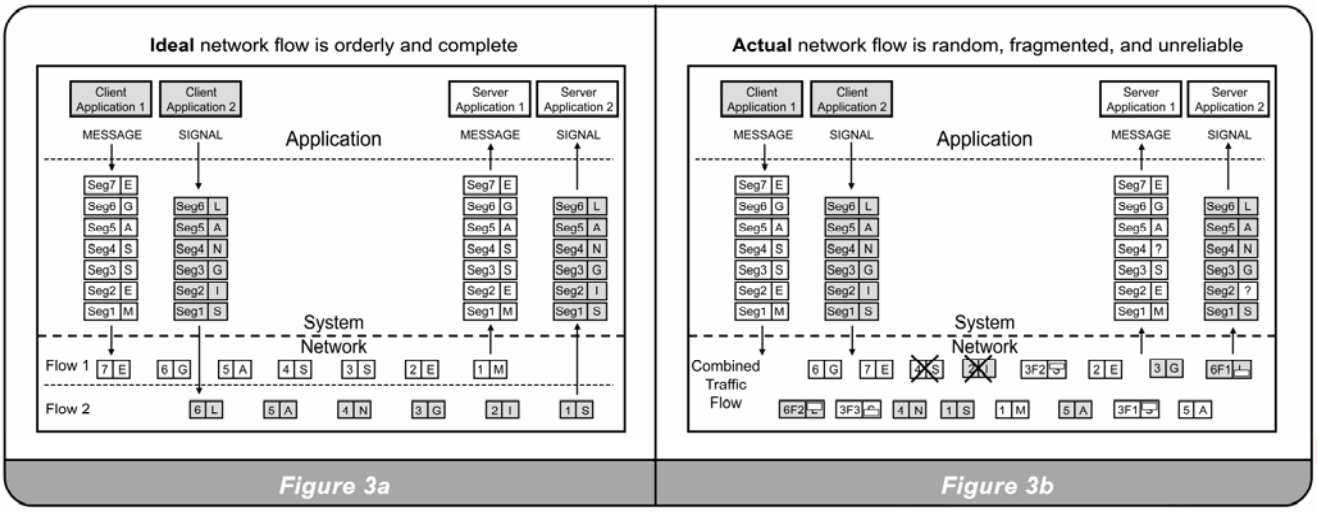


Figure 3a Figure 3b

In order to process traffic accurately, techniques must be used to eliminate the misinterpretation of data. These techniques are:

- IP de-fragmentation – the ability to properly combine fragments of packets into packets
- TCP reassembly – the ability to properly reassemble the TCP segments in the right order, while removing duplicate and overlapping data
- Flow tracking – the ability to track flows (client to server flow and server to client flow) and associate them with a single communication session
- Normalization – the ability to interpret and, if necessary, remove encoded representations and special characters from the reassembled message

Optimized Analysis = Stateful Signatures

Once packets are delivered, processed and reconstructed from the network, the next step is to accurately detect intrusions within this traffic flow. Most NIDSes on the market use packet signature detection, which means that they look at the raw bytes of each and every packet in a flow to try to find a match for an attack pattern. Some NIDSes are able to perform IP de-fragmentation and TCP reassembly to reduce misinterpretation, however, when they look for patterns they still look in the entire ordered stream of data. This introduces two problems:

- Performance is significantly hurt, since the entire flow needs to be searched
- False positives are more likely to occur, based on the simple fact that the more data a system searches, the more likely it will match a signature to irrelevant data

The good news is there is an alternative that overcomes the performance and accuracy deficiencies of packet signature detection, called Stateful Signature™ detection. This advanced detection mechanism identifies attack patterns by utilizing both Stateful Inspection and protocol analysis, which is performed as part of protocol anomaly detection. As a result, Stateful Signatures understand the context of each data byte and the state of the client and server at the time of transmission. This means that Stateful Signatures can be compared to only relevant data bytes, according to the communication state to which each signature is relevant. In other words, Stateful Signatures only look for an attack in the state of the communication where that attack can cause damage, significantly improving performance and reducing false positives.

Example 3: Detecting the SMTP Wiz Attack

This example describes the SMTP (sendmail) 'wiz' attack and how it would be detected using Stateful Signature detection versus packet signature detection. The wiz attack allows an attacker to gain root access on a host that runs the SMTP server. When successful, the attacker can take complete control over the "victim" host, use it as a platform for launching further attacks, steal e-mail and other data and ultimately gain deeper access to the network and systems. The attack is accomplished by sending the command "wiz" to the server, while the sendmail session between the client and server is in the "command mode."

Let's look at a typical SMTP session more closely to understand how the attack can be detected with more accuracy using Stateful Signatures. Figure 4 illustrates the four phases of a typical SMTP session:

1. Establishing the TCP connection (3-way handshake)
2. Command mode:
    a. Setting the sender ("MAIL FROM" field)
    b. Setting the recipient ("RCPT TO" field)
3. Executing other commands
4. Data mode (the contents of the e-mail)
5. Closing the TCP connection (4-way close)

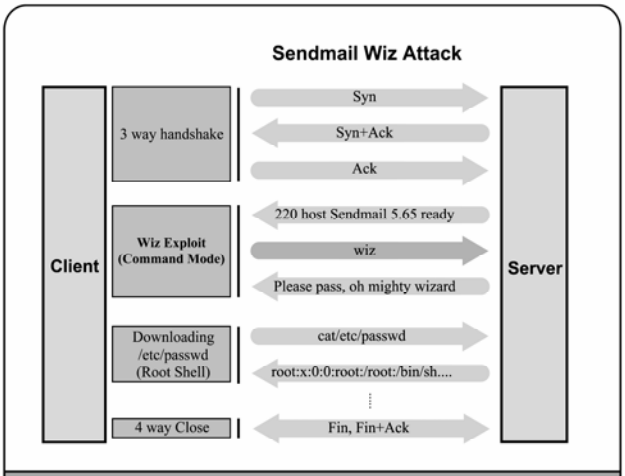


Figure 4

A system based on a packet signature detection mechanism, looks at individual packets for attack patterns, with no regard to the context of the communication and the state of the client and server. In this manner, any string, in any portion of the communication session, that contains 'wiz' would be detected. There is a high probability of triggering a false alarm because "wiz" may be found in the e-mail body (data mode) or the e-mail recipient list (command mode), as well as the "wiz" command (command mode). This may seem trivial, but actually 'wiz' appears quite a lot in e-mail. In fact, it occurs in almost every attachment that someone sends over 100K. The reason is that all e-mail attachments are encoded using 'base64' (RFC 1421), which represents binary data (Word, PowerPoint and other applications) with 64 printable characters. Six of these characters are 'w', 'W', 'i', 'I', 'z', 'Z'. Since attachment data is quite random, the mathematical probability of 'wiz' occurring in a random character stream (with every letter either upper or lower case) is 1 in 32,768 characters (32*32*32). In addition, 'wiz' shows up in common use terms found in an e-mail client or server, such as the e-mail address of thewizard@company.com or an e-mail message that contains the word "wizardry" or "wizened," etc. With packet signature detection, all of these instances would trigger an alarm.

A system using Stateful Signatures only looks for the "wiz" pattern contained within the client to server SMTP flow in the "Command Mode." Additionally, Stateful Signatures ignore e-mail addresses that may appear in that mode because they are not where the attack can be perpetrated. Stateful Signatures perform the pattern comparison against the portions of the client-server session that are vulnerable, rather than arbitrarily searching for strings in the entire session. As a result, you spend less time tuning the system and investigating false positives and gain confidence that the system's alarms are real.

Intrusion Detection Using Backdoor Detection

We have discussed how to detect attacks that violate a protocol (protocol anomaly detection) and attacks that are well known and characterized (signature detection), but we need to understand how to detect attacks that are unknown that don't violate a protocol, such as a Trojan or Worm. These attacks install and open up a backdoor on a network resource. This backdoor lays dormant until the attacker activates it and takes control over the resource. This is accomplished through a series of interactions, where the attacker sends commands and the resource obliges. Because there is no attack pattern and no protocol being violated, another method is needed to detect this interactive traffic.

Since the Worm or Trojan implantation may have been through a backdoor (modem connection, instant messaging or a home laptop that is plugged into the corporate network) or may have been missed by using another detection mechanism, the use of multiple methods of detection greatly increases the probability of attack detection.

Example 4: Installing a Worm through Instant Messaging

For example, many companies allow employees to use instant messaging. While it may be a useful communications tool, it also opens up a way for attackers to enter your network. Most Instant Messaging products, like Yahoo! Messenger™, allow users to send attachments. These attachments can contain malicious code, such as a worm, which can be installed during the download on the user's computer without them ever knowing. Once installed, the attacker can return and interact with the malicious code, instructing it to send them a file, reformat the hard drive, launch other attacks, etc. This type of attack gives the attacker complete control over that resource and ultimately your network.

Juniper has invented a method, called backdoor detection that can detect the unique characteristics of this interactive traffic. Juniper's IDP looks for all interactive traffic and then detects that which is unauthorized, based on what the administrator has defined as "allowed" in the rule-base. This method can detect virtually any backdoor, even if the traffic is encrypted and the protocol is unknown.

Intrusion Detection Using Traffic Anomalies

There are many attacks that are contained in a given communication session, but it is also important to detect the attacks that occur in traffic spanning multiple sessions. The best examples of these kinds of attacks are port and network scans. Port and network scans occur when an attacker uses a tool to determine which services are allowed and responding on a system. This is accomplished by trying each and every port on a single machine (port scanning) or a certain port on an entire network (network scanning). The attacker uses this information to exploit known vulnerabilities for the responding services on those open ports.

For these types of attacks, patterns need to be detected within the overall traffic flow and usually require some form of frequency and threshold triggers. These are the types of attacks that traffic anomaly detection is designed to identify.

Example 5: Detecting a Network Scan

This example describes a network scan "attack" and how it would be detected. As previously explained, in a network scan the attacker will try to access a specific service, such as the SMTP port, across an entire network. It is typically a precursor to an attack against that service. Neither protocol anomaly detection nor Stateful Signature detection would be able to detect this "attack," due to the fact that a scan conforms to the protocol and the pattern does not appear within a particular session. The only way to detect this "attack" is to use traffic anomaly detection, which performs pattern matching against the entire flow of traffic. As a final note, a network scan is not really an attack. It is, however, a good indicator of an imminent attack because it means that someone is trying to find out what is running on the system. If you have knowledge of a network scan, you can watch for and anticipate a following attack, which is why detecting network scans is as important as detecting the attack itself.

Pattern Matching Using Regular Expressions

Accuracy is not only affected by the types of detection methods a system uses, but also the way the system defines and looks for attack patterns. Some systems define and search for fixed patterns. This approach is highly inefficient because dozens of attack signatures exist with different permutations. The ideal method is to provide support for regular expression pattern matching. Regular expressions provide wildcard and complex pattern matching, resulting in a more accurate representation of an attack. Regular expressions also offer flexibility over the control of the system's behavior.

For example, to look for an e-mail message with an executable attachment, an NIDS should look for the pattern:

name = "<some-name>.EXE"

where *some-name* may be any valid filename.

The problem is that the '=' sign can be preceded or followed by any number of spaces and tabs. For example, the pattern should match 'name = "run-me.ExE"'. NIDS products that are not equipped with regular expression matching do not have the ability to specify that there can be any number of spaces and tabs around the '=' and will only look for the "standard" case of no spaces around the '='. By surrounding the '=' with spaces and tabs, any attacker can send a virus infected executable, while hiding the attack from an NIDS not equipped with regular expression matching.

Intrusion Detection and Prevention Must Operate In-line

Most of the NIDS products on the market are unable to prevent attacks because they are "passive," sniffer-based intrusion detection systems. These systems can only "listen" to the traffic. They cannot control the traffic, by either dropping, modifying, steering or delaying packets or by injecting their own packets into the network, when appropriate. It is easy to understand why most NIDSes operate as passive devices. Any intervention with the flow of packets may be dangerous if you can't trust the results of the attack identification. To date, the results of current NIDSes are hampered by false positives and inaccurate attack identification, due to a reliance on single, poorly implemented methods of detection. However, if an NIDS can demonstrate accurate results, it is important to understand why it needs to operate in-line to provide complete protection.

Preventing Evasion

Passive intrusion detection systems will always be vulnerable to evasion techniques. This is due to the fact that passive network monitoring enables traffic ambiguity, making it very difficult to ever provide a reliable intrusion detection mechanism. This section will explain the basics of evading intrusion detection systems; how it is being done in practice and why it is always possible to evade passive intrusion detection systems. After reading this section it will be clear that the only way to stop evasion is to operate an intrusion detection system as an in-line device to eliminate traffic ambiguity.

Evading Intrusion Detection Systems: The Theory

The basic idea behind evasion is to fool the intrusion detection mechanism into "seeing" different data than the target host, often referred to as the "victim." This allows an attacker to attack the host without being detected. This applies to both signature-based and protocol anomaly-based intrusion detection systems.

To understand the idea, you first need to understand how an NIDS performs TCP reassembly. (Refer to Figure 5 for an example.) Application 1 generates information for communication as a stream of data (MESSAGE). The operating system breaks that stream into segments ("TCP segments") and sends them to the network. The receiving operation systems collects the TCP segments and converts them back to a stream of data (MESSAGE), which is then presented to the receiving application. Since the underlying network does not guarantee delivery of the TCP segments, the receiver tells the sender which segments have and have not been received. The sender can then retransmit the missing segments.

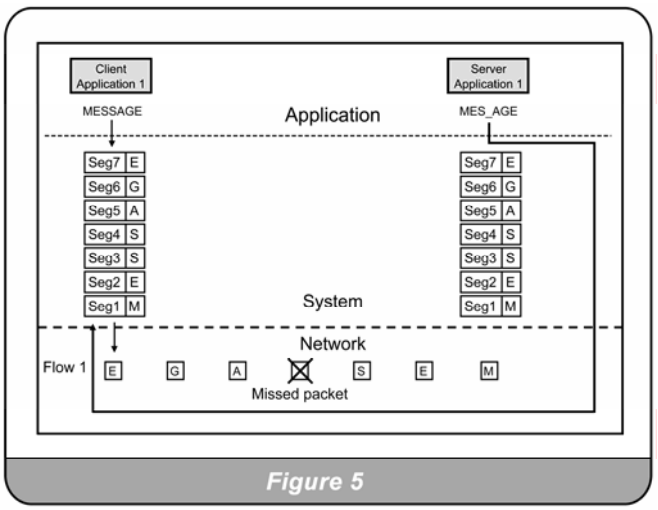


Figure 5

TCP reassembly, in the context of NIDS, is the process of collecting TCP segments in the order that the sending application sent them and extracting the application data stream from them. When performing TCP reassembly as a passive system, it is easy to look at each and every TCP segment (packet), but it very difficult to perform the TCP reassembly in a manner that is consistent with what the receiver would "see."

Upon receiving a TCP segment, the NIDS sensor needs to decide what to do with it. There are several handling options for the receiver of the TCP segments:

1. **Use the entire segment** – this is the most common scenario
2. **Use part of the segment** for TCP reassembly, while ignoring the rest of it – this happens when the segment overlaps another segment that has already been received, or if part of the segment is beyond the window (sequence number of packets that the receiver is anticipating) and is, therefore, ignored.
3. **Ignore the segment** completely – this happens if the segment was received before or is invalid. Invalid packets include those that have incorrect IP or TCP checksums, invalid TCP flags, old TCP timestamps, as well as many other situations.
4. **Did not receive the segment** - The receiver might not even receive the TCP segment because of network issues (the segment might disappear) or packet settings (such as IP TTL). This is equivalent to #3.

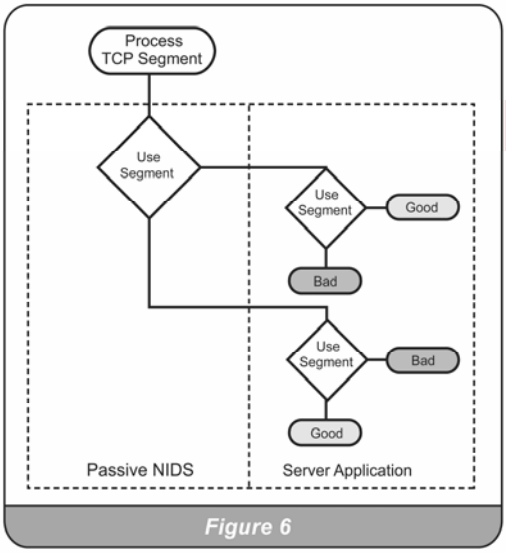


Figure 6

The NIDS sensor must figure out how the intended receiver will handle the TCP segment and handle it exactly the same way. If the NIDS makes the wrong decision:

- The NIDS sensor will use a segment that the receiver will not, thus seeing data that the receiver will ignore
- The NIDS sensor will ignore a segment that the receiver will use, thus missing data

In both cases, the NIDS sensor will be running its checks (either signature matching or protocol anomaly detection) on the wrong data. (See figure 6.)

Evasion in Practice

Now that the basics of NIDS TCP reassembly are understood, the problem of NIDS evasion can be restated:

NIDS evasion happens when an NIDS sensor makes a mistake in using or not using a part or an entire TCP segment for TCP reassembly. Such a mistake will make the NIDS see different data than the sender and receiver of the TCP flow.

In practice, to evade the NIDS sensors, the attacker constructs TCP segments that make it virtually impossible for the NIDS to determine whether or not the "victim" will accept them. If the victim does accept them, it is impossible for the NIDS to determine which portion of the

segments it would use. Such TCP segments are referred to as "*ambiguous TCP segments.*"

The attacker then retransmits the same TCP segment with different data, triggering either one of the following scenarios:

- The NIDS sensor used the original dummy TCP segment in its reassembly and the victim has dropped it. The second TCP segment, which contains the attack, is ignored by the NIDS sensor (since it has already seen that segment before) and used by the victim (since it ignored the previous one); or
- The NIDS sensor ignored the original attacking TCP segment in its reassembly, while the victim used it. The second TCP segment, which is a dummy, is accepted by the NIDS sensor and ignored by the victim.

Constructing Ambiguous TCP segments

This section presents several different methods for constructing ambiguous TCP segments. While some of these methods may have a way for an NIDS to deal with them, most do not have a theoretical algorithm that would enable an NIDS to determine what the victim will do with the TCP segments.

Effortless construction of ambiguous TCP segments

There exist some very simple techniques to create ambiguous TCP segments. A good NIDS must be able to eliminate these ambiguities in order to properly interpret the traffic being transmitted. The techniques to create ambiguous segments are:

- Invalid TCP checksums – the victim will certainly drop a packet with an invalid TCP checksum. A NIDS that fails to validate the TCP checksum will be accepting packets that are dropped by the victim.
- Out-of-window data – a receiver only accepts data within a certain window, called the "receiver window." The receiver will ignore data out this window. However, not being the actual receiver, it is very difficult for an NIDS that's performing a passive reassembly to determine exactly what the current receiver window is. An attacker can therefore send data that is marginally in or out of the victim's receiver window and confuse the NIDS.

Exploiting inconsistent RFC interpretation by TCP implementations

The various RFC's that provide the specifications for TCP (RFC 793, RFC 1323 and others) are quite complex and leave room for interpretation by the implementer. In addition, some TCP implementations have deliberately or accidentally deviated from the specifications. The result is the same TCP segment can be accepted or rejected by different TCP stack implementations, which gives attackers many opportunities to construct ambiguous TCP segments.

Example 5: Overlapping TCP segments

This example describes how overlapping TCP segments create ambiguity and how this ambiguity can be prevented.

An attacker can send varying contents in overlapping TCP segments, i.e. two of the same segments that contain different data. Different TCP stack implementations interpret the order differently, some use the first received, while others use the last received. For example, Windows always takes data from the older segment, while Solaris takes it from the newer one. BSD and Linux are not consistent and sometimes prefer the older TCP segment and sometimes the newer one. Without knowing the exact behavior and circumstance of the victim's TCP implementation, an NIDS is not able to determine what the right behavior is and whether or not to send an alarm.

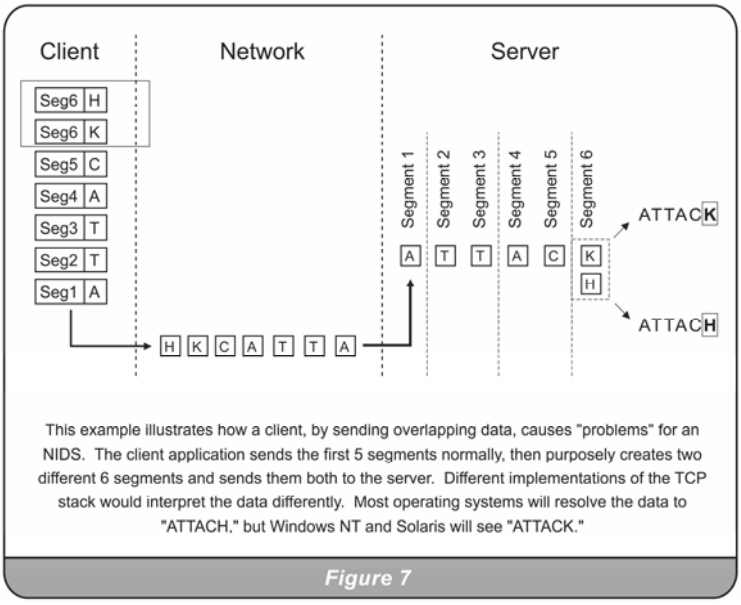


Figure 7

Preventing NIDS Evasion

The good news about all NIDS evasion techniques is that it is relatively easy to detect ambiguous packets. For example, if an intrusion detection system sees two overlapping TCP segments with different content, it is certainly an ambiguity and probably an evasion attempt. However, while detection is possible, it is not feasible to determine what the target host would do with the ambiguous packets. This means that it is impossible to determine whether the target host has been compromised or not. Moreover, when seeing multiple ambiguous packets over the same connection, an intrusion detection device knows that there **is** an attack but determining **which** attack is being attempted is impractical. The reason is that there are potentially millions of different combinations for the way the ambiguous packets would be reassembled but only a single combination that reveals the attack. Finding that single combination can take hours, days and even years, which is not practical for a real-time intrusion detection system.

This leads us to the following conclusion:

Once an ambiguous packet is received at its target, the intrusion detection system becomes blind and cannot determine which attack is being attempted and whether it was successful or not.

Meaning, the only way to avoid intrusion detection evasion is to prevent ambiguous packets from ever reaching their destination. This, of course, can only be achieved if the intrusion detection device is in the path of the data, i.e. running in-line.

Preventing Intrusions

The most significant drawback to a passive intrusion detection system is that it is ***passive*** and cannot control whether or not the attack is allowed to reach its target. Therefore, passive systems can really only be used for detecting attacks, rather then preventing them. However, it has been suggested that there are mechanisms available to passive devices that are able to stop attacks. These mechanisms are:

- Sending TCP resets
- Signaling to a firewall or router to block the traffic

This section will explain these mechanisms and demonstrate why they are unable to truly prevent an attack.

TCP Reset

The basic mechanism behind a TCP reset is that the intrusion detection device sends a reset packet to both the client and the server when it detects an attack. There are several problems with this approach. First, it takes a short while for the intrusion detection device to determine that an intrusion has been attempted and that a reset packet should be sent. During that time, the offending segment and, most likely, some of the packets that follow have already been transferred to the target network and reached their "victim." As a result, any reset that is sent upon detection may be too late. The second folly of TCP resets is that they only work for TCP protocols. They do not work for UDP-based protocols, such as DNS. The third, and most significant, problem has to do with the way TCP resets work. A TCP reset must carry a valid sequence number in order for the server to accept it. To be valid, the sequence number has to be within a certain, relatively small "*receiver window*." A sophisticated attacker can transmit its attacking segments so fast and have the receiver window change so rapidly that a passive device will have a very hard time determining which sequence number to put in its reset packets. Therefore, most attempts at a TCP reset would fail to stop the attack; sometimes it works, but usually it doesn't.

Firewall Signaling

Firewall signaling really is not prevention at all. Instead, it is an active feedback mechanism to the access control system (typically a firewall or a router). Firewall signaling gives the intrusion detection system the ability to tell a firewall what type of traffic should be "blocked" at the firewall. The desired outcome is an adjustment of the firewall policy to protect against future attacks. This sounds like a great idea, but, as it turns out, the danger of this method outweighs its benefits. It is relatively easy for an attacker to spoof the source of its attack and make the NIDS believe that it is being attacked by another IP address. If an attacker, for example, spoofed the IP address of a major Internet Service Provider's proxy server, such as America Online (AOL), and the NIDS signals the firewall to block that IP address, then service to the entire AOL community will be blocked. Sometimes, the attacker doesn't even need to spoof an IP address to achieve this result, they can simply send an attack as an AOL user. This makes the NIDS-Firewall combination an opening for simple, yet potentially devastating, denial-of-service (DoS) attacks.

Active Prevention Devices Must be In-line

The only method of preventing intrusions is to operate as an in-line device, which gives the network security device the ability to drop offending packets at the point of contention. This allows the system to handle all types of traffic and, when combined with the appropriate intrusion detection mechanisms, provide the highest level of security possible.

Manageability

An integral element of any NIDS is how you interact with the system. If the NIDS is difficult to configure and manage, it will be difficult to benefit from its capabilities, even if it contains all of the detection mechanisms and response options in the world. The best solutions on the market today are the ones that provide very granular control of system capabilities, using a centralized, rule-based management scheme. This approach has proven effective in the management of firewalls. Applied to an NIDS, it enables you to control all devices under management with a single, logical Security Policy that is made up of individual rules. The concept behind rule-based management is to use an easy to define sequential set of logical expressions (or rules), containing the basic format of matching criteria and associated action specifications. This format gives you exacting control over what traffic the system looks for and what it does when that particular traffic is detected. The centralized, rule-based approach also enables you to determine how you would like each rule applied. For example, you can apply a rule to one or all of the sensors and then make the rule live with the push of a single button. With a centralized management system, you do not need to spend the time and effort to update each and every sensor, instead, you simply update the central Security Policy and the corresponding sensors get automatically updated. This approach simplifies configuration and signature updates and improves the accuracy of the overall system, by allowing you to apply functionality to particular devices based on the vulnerabilities and needs of the network.

The Juniper Approach

Juniper has taken an innovative approach to network security with its Intrusion Detection and Prevention™ product line (Juniper Networks NetScreen-IDP™ 10, 100, 500 and 1000). Built from the ground up, Juniper's IDP is the first product on the market to use many of the advanced technologies discussed in this white paper. The first innovation implements a technique called Multi-Method Detection (MMD™). With MMD, Juniper utilizes eight different detection methods, including protocol anomaly detection, traffic anomaly detection, some advanced techniques, such as Stateful Signature™ and backdoor detection, and others that are outside the scope of this white paper, to maximize the amount of attacks detected, while reducing false alarms. The second innovation is Juniper's IDP operates as an in-line solution, which is the only way to protect against evasion techniques and provide true intrusion prevention. The third innovation delivers a centralized, rule-based management framework. The centralized management allows administrators to control Juniper's IDP system at a very granular level, while simplifying Security Policy and signature updates.

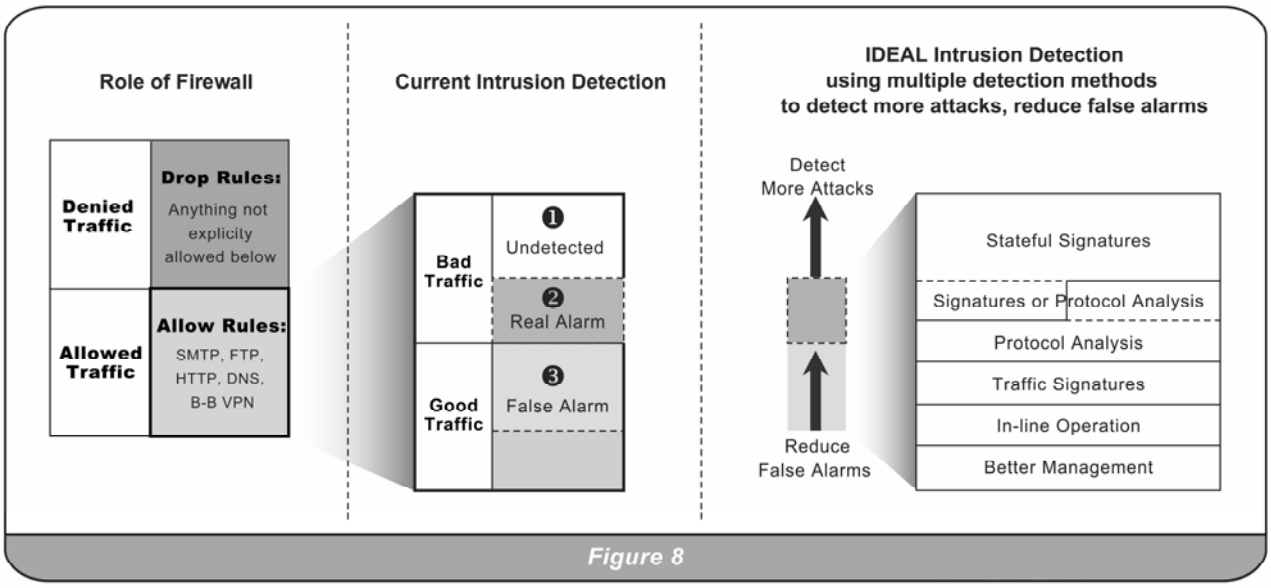


Figure 8

Multi-Method Detection (MMD) Improves Accuracy

Aware that no single approach can detect all network intrusion attempts, Juniper built a system designed to optimize the detection of suspicious traffic with MMD. By implementing eight different detection methods, including protocol anomaly, Stateful Signature, traffic anomaly, backdoor, Syn-flood, IP spoof and layer 2 detection, as well as a network honeypot, Juniper's IDP system can accurately identify intrusions. Unlike other vendor approaches that use a single intrusion detection mechanism to drive their overall product architecture, the Juniper architecture was developed to use all of the available detection mechanisms. These methods share information and work together to identify, in the most efficient manner, all types of attacks at both the network and application layer. The detection mechanisms are optimized to perform analysis at very high data rates, with virtually no performance degradation. You no longer need to decide to buy either a protocol anomaly-based system or a signature-based system or, even worse, spend the money on two or more products, because

Juniper's IDP can provide you comprehensive coverage. Plus, because the system is implementing MMD, administrators can trust that the alarms they get are real and don't have to worry about wasting time investigating false alarms.

Packet Processing for Accurate Data Representation

In order to properly interpret and represent the traffic flow, Juniper's IDP system has a variety of packet processing techniques to ensure accurate data representation. These techniques are:

1. *IP de-fragmentation* and *TCP reassembly* to properly reconstruct traffic, so that it is viewed the same way the intended target system would see it.
2. *Flow Tracking* to match up multiple connections as a single session for more accurate analysis.
3. *Protocol Normalization* to decode data-streams into a common format, so that accurate analysis can be performed.

In-line operation provides real protection

Juniper's IDP is designed to work in-line, in the path of packets. In this configuration, IDP is usually placed behind the firewall, inspecting each and every packet going in and coming out of each of the protected networks. When Juniper's IDP detects malicious traffic, it can drop the connection, so that it never gets onto the network. Naturally, you are given full control over exactly what types of traffic justify dropping the connection. In contrast, when a passive NIDS detects malicious traffic, its only real recourse is to send a TCP reset to try and stop the attack. Unfortunately, due to the nature of TCP resets (see section 2.3.2.1) you cannot be certain the attack was stopped in time. This means that you need to take the time to investigate whether the attack ever reached its "victim." Then you have to learn how the attack was perpetrated and try to tune the NIDS to protect against similar future exploits. Finally, if the attack was successful, you have to assess the damage caused by the attacker, constituting both hard and soft costs wasted in the process. With Juniper's IDP, you can drop the offending traffic and be certain that the traffic never made it to its target.

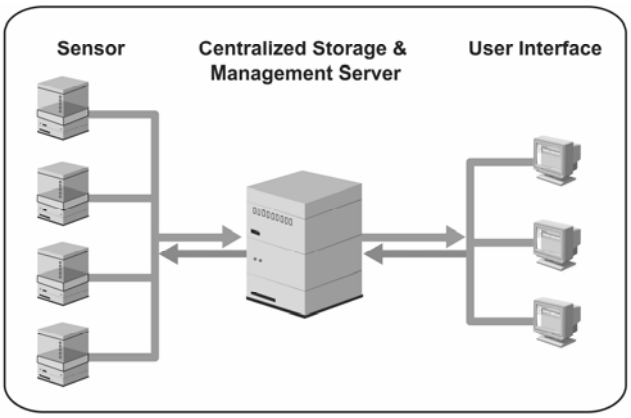
There are many advantages to operating Juniper's IDP in the path of packets:

- Attacks can be stopped (dropped) the moment they are detected, guaranteeing their failure and preventing any damage that would have resulted from a successful attack.
- You can be certain that intrusions that were dropped were unsuccessful, so you only need to investigate on a case-by-case basis. This frees up your time, so you can concentrate on other projects.
- Common methods of evading IDS devices can be avoided, by having Juniper's IDP verify the packets for both structure and content before they get to their intended target. This prevents an attacker from taking advantage of ambiguities.

Note that, as an in-line device, Juniper's IDP can be implemented either as a router or a transparent switch, which requires no IP address and routing modifications. Juniper's IDP can also be deployed as a passive NIDS (sniffer), with the benefit of MMD but, of course, without the ability to prevent attacks. Finally, to facilitate fault tolerant operations, Juniper's IDP system supports high-availability configurations.

Centralized, Rule-based management provides greater control

Juniper's IDP system uses a centralized, rule-based management approach to offer:



- True three-tier management architecture, comprised of a detection and enforcement tier (sensor), a management tier (server) and an application tier (user interface). Multiple user interfaces can connect to a single management server to perform all management operations.

- Rule-based management, providing granular control over how IDP behaves. You set the rules by specifying the source, destination, service and attacks that need to be searched for and matched, in order for the rule to apply. The rule then specifies what to do when those attacks are detected, such as drop or allow the connection, and how to log the attack.

- Centralized Security Policy, allowing the application of the same Security Policy to as many enforcement points as required. Deviation from one device to another does not require a new Security Policy; it is simply accomplished by specifying to which device each of the individual rules within the Security Policy applies.

- Closed Loop Investigation, enabling you to move freely from summary reports to individual logs to the rules that triggered the log to the packet data of the log. This ability to correlate data points and move between levels of information makes it easy to understand exactly what is going on in your network and immediately react to protect against new threats.

Summary

Firewall systems do a good job of controlling which traffic is permitted to come in and out of your network. However, it is inevitable that some of the traffic that they allow is malicious in nature. You need a second layer of defense to complement your firewall and detect and prevent all types of attacks. To date, passive network Intrusion Detection Systems (NIDS) have been employed to try and detect network attacks. Unfortunately, current intrusion detection solutions generally implement only a single intrusion detection mechanism, accounting for a lot of false positives and missed attacks. In addition, they are passive, so they cannot stop an attack, and are notoriously difficult to manage.

Juniper has developed the IDP system to overcome these deficiencies, giving you a product that protects your corporate assets. Juniper's IDP system brings many proven and well-known concepts together in a single product, delivering a solution that you can trust. These capabilities include:

- Multi-Method Detection (MMD™)
- In-line Operation
- Centralized, Rule-based Management

Look for these capabilities the next time you think about reliable network security.