Intrusion detection evasion:
How Attackers get past the burglar alarm

Corbin Del Carlo
SANS Great Lakes, Chicago Illinois
May 18 - 23 2003
GSEC Practical (version 1.4b)
September 25, 2003

Table of Contents

Abstract

The purpose of this paper is to show methods that attackers can use to fool IDS systems into thinking their attack is legitimate traffic. With techniques like obfuscation, fragmentation, Denial of Service, and application hijacking the attacker can pass traffic under the nose of an IDS to prevent their detection. These are techniques that the next generation of IDS needs to be able to account for and prevent. Since it would be almost impossible to create a product that was not vulnerable to one of these types of deception I suggest that a new type of product needs to evolve and emerge, a centralized security Management Platform.

Definitions

False Positive – A False Positive is when the IDS alerts appropriate personnel about network traffic that is not malicious. This can be the result of a faulty signature that is too easily matched by normal traffic, or abnormal traffic that does not meet existing baselines due to new system installation or software upgrades.

False Negative – A False Negative is when an IDS fails to alert when a valid attack occurs.

Network Intrusion Detection Systems (NIDS) - Network Intrusion Detection Systems use information gathered from a passive interface in promiscuous mode to detect attack patterns. NIDS, depending on placement within a network, can provide a great deal of network visibility and protect a substantial number of hosts with a single device and configuration. Most NIDS, but not all, now provide some sort of reaction such as alerting an Administrator and blocking further communication with the attacking host.

Nework Intrusion Prevention Systems (NIPS) - Some Intrusion detection systems act as intrusive monitors as apposed to passive, which allows them to fail safe instead of fail open. These are called Intrusion Prevention Systems (IPS). For the purposes of this document the terms are the same, because they can be fooled in the same ways.

Host Intrusion Detection System (HIDS) – Host Intrusion Detection Systems are software installed on the local system. Typical HIDS are very similar to virus protection software. They look for activity that matches a known attack signature and either allows the activity or prevents it. Some HIDS depend on abnormality detection; where current traffic is compared against baseline traffic, with anything that is not part of the baseline causing an alert. HIDS provides a more customizable and accurate method of intrusion detection but is much more administratively intensive than NIDS and, in an enterprise with several servers, could be substantially more expensive.

Message Digest 5 (MD5) - An algorithm that takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. [11]

Universal Resource Locator (URL) - the global address of documents and other resources on the World Wide Web. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located. [9]

Universal Resource Identifier (URI) - the generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI. [10]

Introduction

Intrusion Detection systems (IDS) are becoming more and more widely deployed to supplement the security provided by firewalls. IDSs function in the digital world much the same way as a burglar alarm does in the physical world. Like all alarms, IDSs have weak points and flaws that can be exploited by an attacker to get around the system. Some people have even begun to pronounce the death of the IDS because of its failure to identify all attacks, frequent false positive alerts and failure to provide a significant Return On Investment (ROI). [5] Yet still for all the advances in security in the physical world almost every business and many homes still have burglar alarms. I believe this will hold true in the digital world as well, no matter how secure systems become there will always be people attempting to gain access they do not need, for this reason IDS constantly needs to advance.

The purpose of this paper is to show methods that attackers can use to fool IDSs into thinking they are legitimate traffic. With techniques like obfuscation, fragmentation, Denial of Service (DoS), and application hijacking the attacker can pass traffic under the nose of an IDS to prevent their detection. These attack techniques will require that the next generation of IDS evolve into a centralized security management platform, capable of a high level of network visibility, allowing it to better protect the network.

Methods of evading Network IDS

When an attacker tries to evade an IDS they determine the weaknesses in the design and attempt to exploit these weaknesses. Each type of IDS has different strengths and weaknesses. NIDS are typically designed as passive (IDS) or intrusive (IPS) monitors of network traffic. This allows a singe NIDS appliance to protect a great deal of systems (as long as they are on the same network segment.)

A NIDS can detect attacks through one of two methods, either signature matching or abnormality detection. [6] The signature matching method works very similarly to today's virus scanners. Each attack has a signature of how it is carried out. When that signature is witnessed on the network the NIDS generates an alarm. With abnormality detection NIDS establish a baseline of the network traffic that is considered normal. It then alarms when conditions are not normal. [6] Abnormality detection NIDS are not as commonly deployed as signature based because of the large amount of time needed to establish the baseline and their initial high rate of false positives. Evasion of abnormality detection NIDS is more a game of luck than skill because the malicious traffic cannot exceed the abnormality thresholds established for the network being monitored. Because of the limited installation base of abnormality detection IDSs, the rest of this section will be referenced against signature based IDSs.

While a large protection scope is the biggest single advantage of NIDS it is also the largest weakness. This weakness can be manipulated through obfuscation, fragmentation, encryption, or overloading.

Obfuscation

Obfuscation is the process of manipulating data is such a way that the IDS signature will not match the packet that is passed but the receiving device with still interpret it properly. [8] For instance, sending a packet encoded differently or adding extraneous null characters. An example is the string:

"../../c:\winnt\system32\netstat.exe"

that would not be interpreted by the IDS the same way as:

"%2e%2e%2f%2e%2e%2fc:\winnt\system32\netstat.exe".

However, a Web server would interpret both strings the same under the interpretation rules of the HyperText Transfer Protocol (HTTP). [8] This particular example was very popular for a time and so both NIDS and Web Server vendors are no longer fooled by this simple example, but the principle is still solid for any interpreter that allows alternate command forms. This particular method can also get by both HIDS and NIDS if done correctly. We will revisit this issue later.

Fragmentation

Fragmentation is simply breaking an attack into multiple packets. [7] Fragmentation of packets occurs normally and hosts are equipped to handle receiving data in multiple pieces and potentially in the wrong order. A Quick example is depicted in Figure 1. In this example the attack packet "DATA" is broken into four different packets. The host at the receiving end will reassemble the fragmented packets and receive the data payload in the fragmented order and then put the packets into the correct sequence using the unique packet sequence number assigned to each packet. But a NIDS is only going to see the parts. Each part is not an attack packet so the NIDS will not alert anyone. Some NIDS's will reassemble packets to avoid fragmentation attacks. But not all NIDS have the processing overhead available to reassemble fragmented packets. But even if the NIDS that can reassemble packets it will have a physical limit to how many it can or will reassemble. So, an attacker can send a fragmented attack and simultaneously send a large amount of fragmented 'junk' packets (an example of a combination attack using the overflow method discussed later). While the IDS is attempting to reassemble all the 'junk' packets, the fragment attack may go though unnoticed by the IDS. Or if the attacker does not have the resources to flood the NIDS, the attacker can also attempt to wait out the capture buffer on the NIDS. The NIDS will, as in the example at the right, get the first three packets but not the entire attack signature, since the last part of the fragment is not received in the appropriate time interval, the first three are dropped by the time the fourth gets to the IDS and host. All four will be reassembled on the host in a successful attack but the NIDS will not alert. [7]

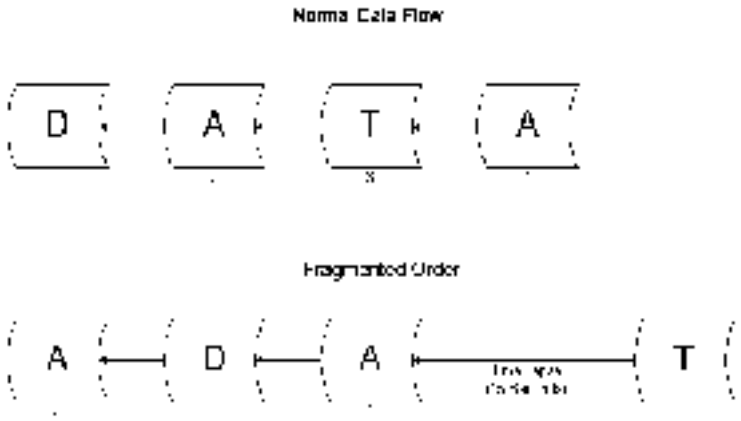

Figure 1 Fragmented Attack

Encryption

A NIDS needs to be able to examine the payload of every packet that crosses its path to be effective. This can be detrimental in multiple ways, the primary being encrypted network traffic. When SSL, SSH, and IPSec encrypted tunnels are established they prevent the NIDS from being able to interpret the packet's true payload. This allows an attacker to use a target's security against themselves. This can be particularly deadly when the system has the same root directory for both unencrypted (http) and encrypted (https) web sites. (This is the default on Microsoft IIS as soon as a certificate is installed).

In this scenario the attacker uses any attack against the HTTPS Web site, such as SQL injection, buffer overflows, and directory traversals, that would work on the HTTP site. Because HTTPS uses SSL to provide a secure network connection, the traffic is encrypted and therefore flows past the NIDS without triggering an alert. Encryption can also prevent an attacker from being detected as they do other unsavory things after they have compromised the host.

The rising popularity of SSL VPNs will also likely contribute to this problem. SSL VPNs are designed to allow portable, easy to setup encrypted sessions between client stations and the corporate network. This allows people to use public Internet terminals to securely connect to their corporate private network to access electronic mail and other internal services.

The SSL VPN has two challenges for our NIDS. First challenge is that with an SSL VPN an attacker can attack the network and because the traffic is encrypted the NIDS will not detect the attack. But even if they are detected, the chances of that individual being caught are very small. With the rising popularity of free Internet access provided by restaurants, cafes, and libraries and a tool like SSL VPN's attackers can attack from anonymous locations with little fear of being tracked electronically or being caught. The second challenge is that, even if the IDS were able to decrypt traffic on the fly to report attacks, the attacker can easily setup a large number of encrypted sessions from other hosts they control that could potentially prevent the NIDS from being able to decrypt the traffic meaningful to the attack (another example of combining multiple attack techniques).

To protect against an attacker encrypting their commands, future NIDSs should alert if they see any outbound encrypted session from any host that does not normally conduct encrypted sessions, as well as any large number of inbound session initiations that would be typical of a brute force password guessing attack over SSH or an SSL VPN. [1]

Denial of Service

The last method to avoid an NIDS is by simply overloading the NIDS. This can be done many ways. The first method is to flood the unit with attacks from spoofed IP addresses, creating so many alarms that the security personnel would have a bad chance of finding the actual attacker. This method does depend on the security professional not pulling the plug in light of all the events. But to avoid this the attacker could use common attacks that the professional should be aware of and the system is patched against so that there is no perceived threat. A NIDS should differentiate between minor and major attacks, so that this situation is prevented.

The second method is to flood the NIDS with traffic so that it cannot possibly look at every packet and simultaneously slip the malicious packets past the overloaded NIDS. To be effective, a NIDS must be able to compare the packet data with the signatures of attacks on every packet it inspects. This can prove very difficult for higher speed links (above 100Mbps). Most NIDS vendors have at least one device that they claim about 600Mbps of performance, but these high speed solutions are expensive. However, NIDS able to process at these speeds would require an extremely large number of coordinated, compromised zombie hosts to flood these NIDS thus reducing the likelihood of such an attack. [1]

Methods of evading a Host Based IDS

Host Based Intrusion Detection Systems (HIDS) work differently than NIDS and therefore have different strengths and weaknesses. HIDS are designed to be installed on individual servers and workstations that you wish to protect. [6] The HIDS software can be implemented a various levels. Some work by monitoring the host for critical files that should not change and alerting appropriate personnel if they change. Other HIDS monitor the network connections, general input strings and system memory for signatures, similar to signature-based NIDS, but only for the machine on which they are installed. File monitoring HIDS like Tripwire, are useful but can be problematic because of several design flaws. Monitoring every file on the system will produce a significant number of false positives, and not all attacks will modify a file. The last type of HIDS attempts to get away from the idea of signature matching. This type of HIDS, called abnormality detection, will report when either a typical hacker action is taken or just alert whenever a system setting is changed. [12] The signature-based HIDS monitors requests made to the system and alerts based on an attack signature. However, if the signature is not matched, no alert is issued. Abnormality detection can have a very high false positive rate if it is not properly configured. While less of a problem for HIDS than for NIDS, false positive generation can still be significant enough to numb security staff into not taking

alerts seriously. As HIDS has improved, most intrusion detection systems use a combination of these elements in their product. [6] The theoretical problem with HIDS is that, if the machine is compromised, what stops the attacker from manipulating the HIDS to prevent their attacks from being alerted?

File Locations and Integrity

HIDS require different methods to avoid generating alarms. The vast majority of known HIDS evasion methods attempt to exploit the signature-based products. Evading file monitoring HIDS has fewer options for evasion besides avoiding file changing exploits. All File HIDS use either MD5 or some variant of this algorithm to make a hash of monitored files. The MD5 algorithm is defined by its author as "It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest." [11] The message digest is akin to a fingerprint. If as little as one bit changes in the original file, the computed MD5 hash will change as well. While it is possible that two files will have the same digest, it would take an attacker an infeasible amount of time to develop a file with the same fingerprint and that file is not very likely to do what the attacker wanted. [11]

The file monitor MD5 database is also usually password or key protected making substituting files difficult. The attacker must just exploit the weaknesses of the file monitoring method's design. Most file monitoring HIDS have directories that are excluded because they will often have files created and removed or modified. Temporary directories are a common location that can be used to evade file monitors. These can be used for the initial exploit code until the attacker can determine where they can safely write files, such as home directories or other locations that are not likely to be monitored. Then, once the system is compromised, the attacker simply needs to remove the file monitor or be stealthy and recomputed file hashes if they can crack the password on the MD5 database.

Most HIDS will check startup scripts, registry entries, and Operating System startup programs for changes so that an attacker cannot modify these files. Few check any tasks that are launched from these locations. For instance, the administrator installs a program that has a custom item that is added to the startup script. Such as Norton adding the Virus Protection system tray icon. An attacker can view the startup locations and then replace the binaries that the startup runs. As long as the replacement still executes the original binary, most HIDS will not alarm because the sensitive startup location has not changed and it is not checking third party application binaries.

The last problem with File Integrity based HIDS is that they tend to alarm too late to be of much use. Since they depend on the host operating system, if the operating system has been compromised there is no way for the administrator to know that the attacker did not disable the HIDS application. Or the use of root kits will allow the attacker to hide files from Operating System as well as other users, which would also prevent the HIDS from finding the files necessary to generate an alarm.

Obfuscation

Obfuscation is the most likely method to be used by an attacker because of its potential to make it past both NIDS and HIDS. Earlier we discussed simple obfuscation methods such as character substitution but now we will discuss some other more detailed methods. The first and easiest method is to manipulate the path referenced in the signature to fool the HIDS. This can be done though self-referencing directories, double slashes, and reverse traversal. Examples of these are at right. Effectively each of the examples is the same physical location. The "./" code references the current directory so any number of these can be added or subtracted. The "///" is usually interpreted as one slash by an application because a directory name cannot

Method	Example
Self Referencing Directories	`/_vti_pvt/./././././administrators.pwd`
Double Slashes	`/_vti_pvt///administrators.pwd`
Reverse Traversal	`/scripts/../_vti_pvt/administrators.pwd`

Example 1 Path Manipulation

be NULL. Reverse traversal is a bit tricky but it moves down the directory tree and then uses the "../" previous directory command to get back to the original location. These methods can be used to make the command extremely long as well. If the command is long enough, the HIDS may, in the interest of saving CPU cycles, not process the entire string and miss the exploit code at the end. [8]

Things get even more problematic for a HIDS when the attacker can declare variables, or do comparisons. While this is not always the case for an outside intrusion, if the attacker has an account on the system, it is likely that these are potential attack vectors. An attacker could easily modify the signature of the attack by making some subtle adjustments to the system. On a typical Unix system, an alias can be defined for any command. By issuing the commands in Example 2, one could hide the malicious code in friendly variables. The Single Quotes break up the signature, but when the alias is executed, they are removed and the passwd file is displayed. Also depicted in Example 2 is how this theory can be used with Environment variables in Unix or in a Windows command line session. All of these examples would cause a typical HIDS to generate an alarm. But, with the modifications presented, it would be difficult to detect. [8] It could be made even more so by using multiple variables, as in the last example, to break up the signature even further making it much more complicated to detect. To detect this condition, the HIDS would have to be able to interpret the command as the command interpreter or shell did. This same theory can be used if the system allows comparisons. All the attacker would need to do is add a comparison that would always be true (such as AND 1=1) to modify the signature enough to evade most HIDS. This method is particularly common in SQL injection attacks.

Method	Example
Shell Alias	`#Alias pass='more ''/etc/''passwd' #pass`
Environment Variables	`#test=/etc #more $test/passwd`
Windows Command-Line Variables	`C:\> set blah=c:\winnt\system32 C:\> set extra=\cmd.exe C:\> %blah%%extra% /c dir c:`

Example 2 Substitution

In all the examples, the theory behind these changes would work for most text based applications such as SMTP, SNMP, SQL query, or telnet. But since Web applications and Web servers are so popular, the vast majority of exploits target these systems.

The remaining obfuscation methods would only work on HTTP requests because they specifically manipulate the way HTTP pages are requested. Before we get into that here is a quick review of how a HTTP request is formatted. Example 3 shows a typical HTTP request. There are four components to a HTTP request, each separated by a space. The "GET" portion is the method and tells the server if we are transmitting or receiving information. The "/pages/index.htm" portion is the URI or Uniform Resource Identifier and tells the Web server what we want. The "HTTP/1.0" is the version number and tells the server which version of the HTTP command structure we are using. Everything after the "HTTP/1.0" is extra information such as headers that tell the server the type of browser being used, etc. [8]

```
GET /pages/index.htm HTTP/1.0
Header:…
```

Example 3 Normal HTTP request

There are several methods an attacker can use this simple format to their advantage. One way to do this is to insert a NULL character in the request after the method. As depicted in Example 4, the request is seen by the HIDS as a blank GET request, but some servers will parse the string into the different components as discussed above.

Method	Example
Null Character Injection	`GET%00 /_vti_pvt/administrators.pwd`
Malformed HTTP requests	`GET<tab>/_vti_pvt/administrators.pwd<tab>`
Parameter Hiding	`GET /index.htm%3fparam=/../_vti_pvt/admin.pwd`

Example 4 HTTP manipulation

Therefore the Method, URI, and version are separated and the NULL character goes on the end of the method. The URI will be interpreted as given (The research shows noted that this only works on Apache) and the exploit will go undetected.

Another method is to replace the space separations for each part of the request with tabs. Most Web servers will condense the white space and still return the value, but a signature based HIDS would not recognize the tab delimiters and therefore not alarm.

One last method to manipulate HTTP requests is to use parameter hiding. This requires that the HIDS signature matching engine excludes checking parameters for efficiency. HIDS often do not examine anything in the request after the question mark because the question mark denotes that the following items are parameters. Since the parameters would be different for all systems, the signature would tend to have a large number of false positives. But the request in Example 3 is a valid request that would allow the user to retrieve the named file, and since all of the signature code is after the %3f (ASCII code for "?", which denotes the beginning of the parameter list) the HIDS would not alert. [8]

Application Hijacking

Application hijacking is very difficult to do, but at the same time, if accomplished, few HIDS will detect it. NIDS will not normally look at the application layer to see that one session has gone from good to evil, and HIDS usually does not interrogate the stack far enough to see that the session is coming from a different host. As I stated earlier, application hijacking is difficult to do, but it is rising in popularity because of the availability of automated tools that do most of the work and the increasing use of stateless protocols such as HTTP that simplify hijacking. Most application hijacking follows the same basic principal. A user starts a legitimate

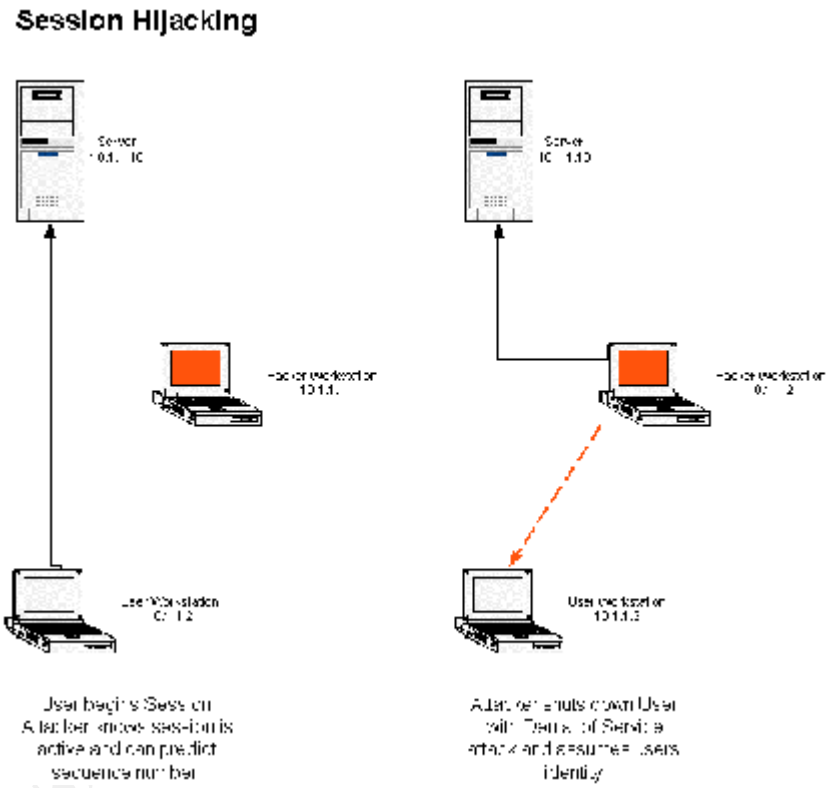

Figure 1

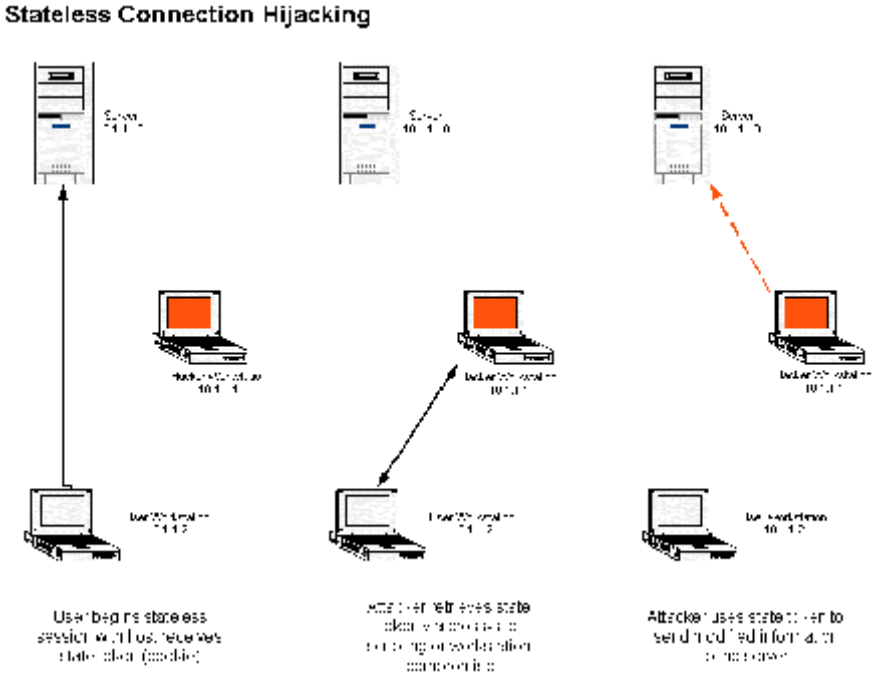

Figure 2

session. The attacker locates and then incapacitates the legitimate user's PC and assumes their session and effectively assuming their legitimate authentication credentials (figure 1). [2] A variant of this is that with a stateless protocol like HTTP, the attacker does not need to incapacitate the original user. The Attacker just needs to steal their session token (figure 2). This can be accomplished either through Cross-site scripting or by attacking the PC's that contain the session tokens. Client PC's tend to not be as well guarded and can prove an easy target. [13]

Man In the Middle Attack

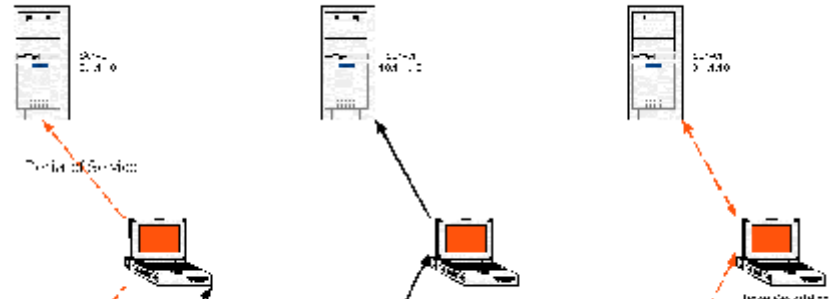

Figure 3

The last variant of this method is the "Man in the Middle" attack where the application hijacker pretends to be the legitimate user to the server and pretends to be the server to the legitimate user (figure 3). The attacker can then modify the session however they see fit. [4] The attack can take a significant amount of time to research and investigate and will potentially never get detected at all. Why is it not detected? The HIDS is designed to protect the host operating system, not it's applications or network interfaces. It does not examine the TCP/IP sequence numbers, that are dropped by the network interface because they were not correct. A NIDS might do this, but most NIDS are not examining enough of the session data to determine if the sequence number is valid. In the second attack, the HIDS on the server or the NIDS (when the Internet is involved) are not going to be able to detect an attack on the client workstation. Also, the HIDS is not examining the application state. It is not feasible for a HIDS to maintain a table of the state of the application for all active sessions. First the HIDS would have to understand all the different states that the particular application can be in and the valid methods to invoke them. While this approach might be possible for a packaged application, it is not something a commercial HIDS vendor is going to be able to do for custom developed Web applications.

However, "Man in the Middle" attacks are the hardest to execute for an attacker. It requires a significant amount of information about both the client and the server operating systems and application protocols. But as far as the HIDS is concerned, it might detect a brief Denial of Service (DoS) attack, but since the server believes the attacker is the client, the HIDS is not likely to generate an alert. Without more visibility, the HIDS is defenseless against this type of attack.

Potential Solutions

The IDS market is getting much more crowded and complicated, with new vendors and devices being introduced. Most solutions follow the same basic idea, but some have more advanced features that will need to be applied universally to be effective.

The addition of a central system logging and reporting "brain" is essential to the effectiveness of IDS. Both NIDS and HIDS suffer from a lack of a complete understanding of each situation to make an accurate assessment. The system log files from all the servers and network devices on any external connection point should be sent to a central device

or server. This central device needs to have an understanding of how the network operates, what are normal operations, what are not, and understand, in detail, how the applications on the local network operate and interact. With a centralized server that has a complete view of the network, the applications and their interaction, such a device can make a much more informed decision regarding any questionable activities. One way that is gaining popularity is to add a vulnerability assessment capability to the IDS engine [3]. By allowing this central server to understand the network, and know which systems are vulnerable to which attacks, the IDS can much more intelligently assess the potential threat and determine the need for an alert. Simply put, if an attacker is attempting to use SunOS vulnerabilities on a Microsoft Windows system, this is of much less severity than an attacker using Microsoft vulnerabilities against a Microsoft system. Even better, if the vulnerability being exploited has been patched and the device knows this though a vulnerability assessment, the system can log the attack but not alert the administrator, significantly decreasing the number of false positives that are brought to the administrator's attention.

Just like its anti-virus cousin, the IDS analysis engine also needs to be periodically updated. Signature-based engines are too easily deceived. With the amount of data that is passing over the typical network and the 1,000+ signatures that most engines need to account for, an IDS is going to have a hard time keeping up with the bandwidth they have to protect. The engine needs to be more efficient as well as use hybrid methods of detection. Signatures will only be able to detect so many attacks and slight modifications to the attacks will likely allow them to go undetected.

Newer engines will need to recognize attack signatures but also alert based on an event list or abnormal network activity detection. For instance, if an IP address runs a port scan, that is an event that would be detected by a signature, but most people would put it as a low priority because it happens all the time. But then the same IP address runs a few ICMP requests, then a connection attempt on UDP port 53. At this point the IDS should alarm. All of these events taken individually would be things that a human operator would make a low priority item because to an administrator each of these signatures could conceivably generate hundreds of alerts a day. But if the IDS only alerts after all three actions have been taken within a set period of time, this is much less likely to be a false positive condition and administrators are much more likely to respond.

The IDS attack matching engine needs to use the strengths of both types of detection. Some attacks have an easy signature to recognize and should be immediately recognized, but abnormality detection methods need to be added to traditional signature engines. Some events need to be alerted simply because it is a major change to the operating environment. For instance if every time a new listen socket is created on a server, the IDS system would test it to make sure that the new service did not respond with a telnet prompt or remote control options, then the IDS would act appropriately depending on the systems response.

The ability to gather much more information than a traditional IDS system and having the system analyze action patterns by some users is not a technically easy task. It would require a HIDS agent installed on any server that has an interface exposed to a network you wish to monitor. It would require a central server that is collecting and analyzing the information from the HIDS agents as well as potentially doing some basic NIDS services to detect Denial of Service attacks. This central server can also receive the log events so that it knows when login attempts fail, or access to a directory is granted when it should be denied. This central server should also do vulnerability assessments to have an intelligent idea about the hosts on the network. This centralized Security Management system then needs a fuzzy logic point system on which to base the alert decision. Some items need to be alerted always and these would merit a full point on the system. But some items are not very important, unless done in conjunction with other items, so these items would have part of a point and the system would only alert when the full point was gained. Like every other IDS system this logic would require the IDS to be "tuned" to the local network but that problem is pretty much unavoidable unless everyone in the world runs the exact same network. The centralized server would also need to have the ability to lock down the network automatically because not all organizations have the luxury of 24x7 monitoring. To do this the centralized server must understand the basic layout of the network. Most IDS only have the understanding of inbound or outbound, but to prevent the IDS from being used against the

organization the IDS should know that if the attack is coming from the inside not to take down the inside network right away as apposed to an attack from the outside.

Lastly, the IDS needs a full featured alerting system. Many systems on the market only notify though email, which is a problem if the attack is a Denial of Service attack against your Mail server. The Centralized Security Management system should be able to alert on voice pager, digital radio pager, text pager, email, or if all fails calling the administrator and playing a recording.

While this type of system is still a few years from really being ready for deployment it is where the IDS market needs to go. IDS will not go away because, even if you lock every door, your business needs to know when someone breaks a window.

References

[1] Cohen, Fred. "50 Ways to defeat your Intrusion Detection System." May 2003. URL: http://all.net (September 27, 2003)

[2] Cohen, Lieton "Session hijacking, Source-routes" February 1999 URL:http://honor.trusecure.com/pipermail/firewall-wizards/1999-Feburary/004642.html (September 27, 2003)

[3] Conrey-Murray, Andrew "Vulnerability Assessment Tools Find New Uses." Network Magazine September 2003: 40-42.

[4] Giovanni, Cortez "Bypassing Secure Web transactions via DNS corruption." April 27, 1999 URL:http://downloads.securityfocus.com/library/middleman.pdf (Sepember 27, 2003)

[5] Hulme, George V. "Gartner: Intrusion Detection On The Way Out" June 13, 2003. URL: http://www.informationweek.com/story/showArticle.jhtml?articleID=10300918 (September 27, 2003)

[6] Kaleton Internet "Combination of Misuse and Anomaly Network Intrusion Detection Systems." March 2002. URL: http://packetstormsecurity.nl/papers/IDS/ (September 27, 2003).

[7] Ptacek, Newsham "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection." January 1998 URL: http://downloads.securityfocus.com/library/ids.ps (September 27, 2003)

[8] Rain Forrest Puppy. "A look at whisker's Anti-IDS tactics." December 1999. URL: http://www.wiretrip.net/rfp/txt/whiskerids.html (September 27, 2003)

[9] "Uniform Resource Locator (URL)" September 2003 URL: http://www.webopedia.com/TERM/U/URL.html (September 27, 2003)

[10] "Uniform Resource Identifier (URI)" September 2003 URL: http://www.webopedia.com/TERM/U/URI.html (October 23, 2003)

[11] Rivest, R. "The MD5 Message-Digest Algorithm." April 1992. URL: http://www.ietf.org/rfc/rfc1321.txt (September 27, 2003)

[12] Sasha, Beetle "A Strict Anomaly Detection Model For IDS." Phrack Magazine Volume 0xa Issue 0x38, May1, 2000. URL: https://www.phrack.com/show.php?p=56&a=11 (September 27, 2003)

[13] Skoudis, Ed "Cross-site scripting Issues and Defeats." 2002 URL:http://www.counterhack.net/xss.ppt (September 27 2003)