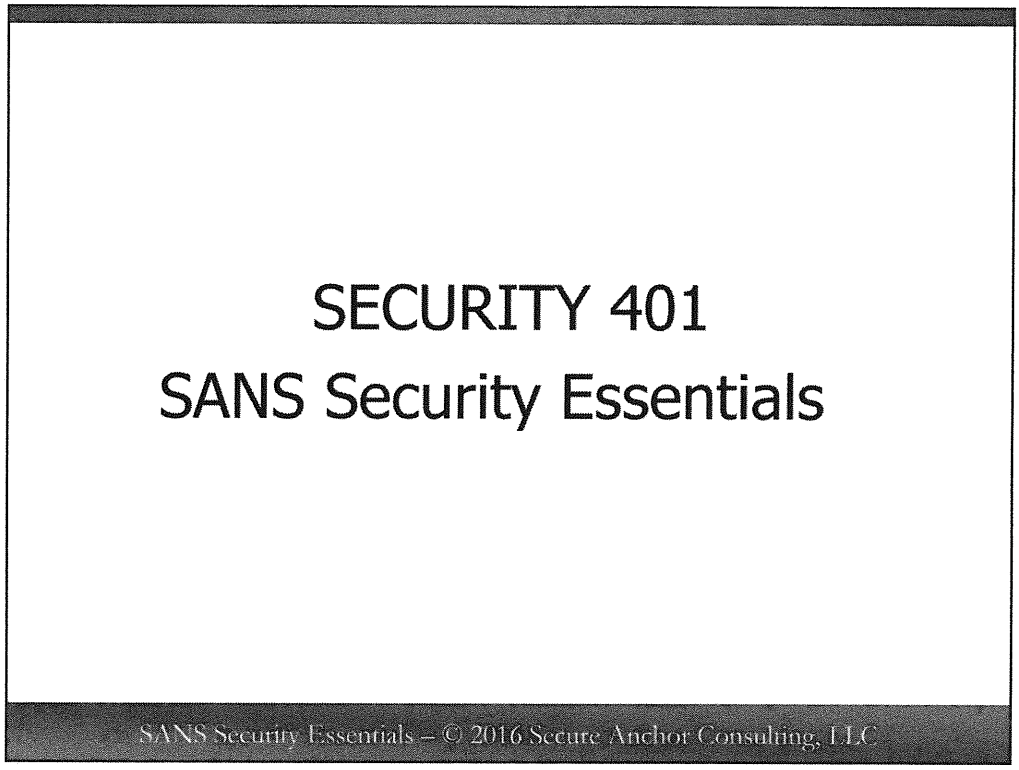
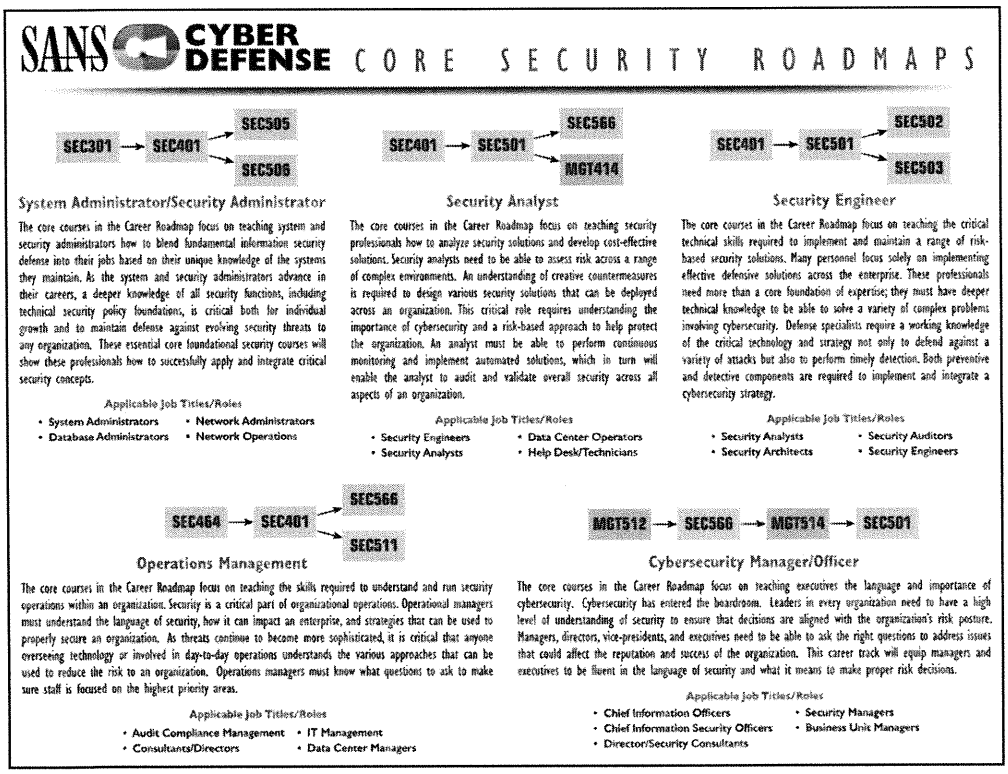
401.2
Defense In-Depth

SANS

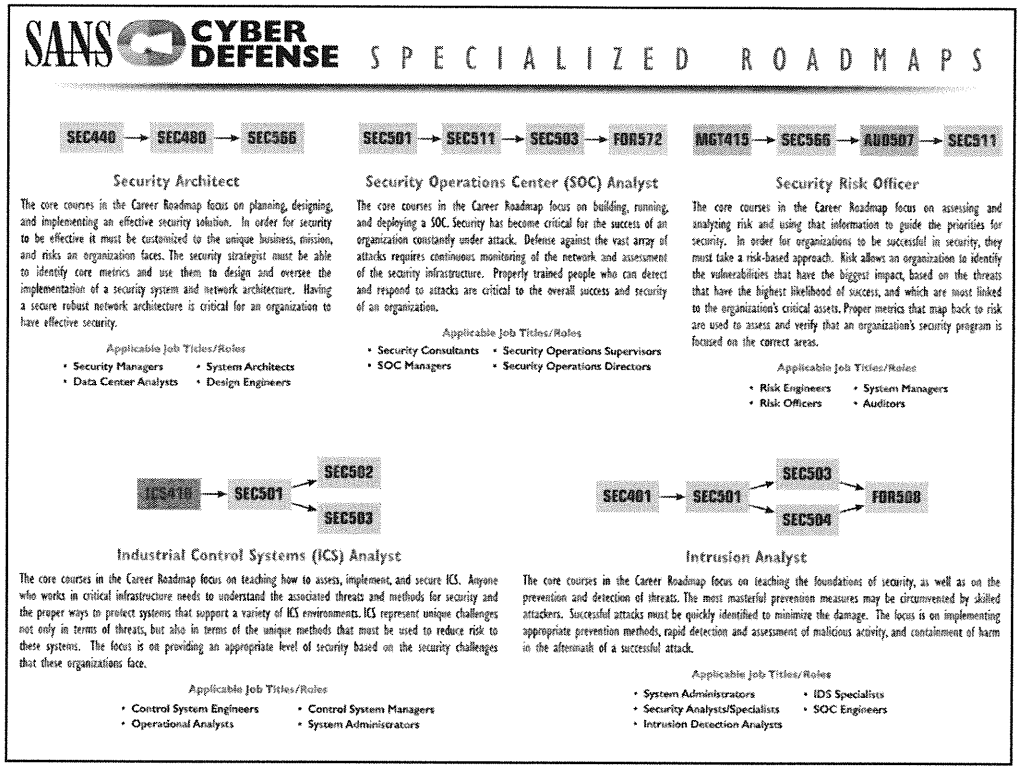
THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org



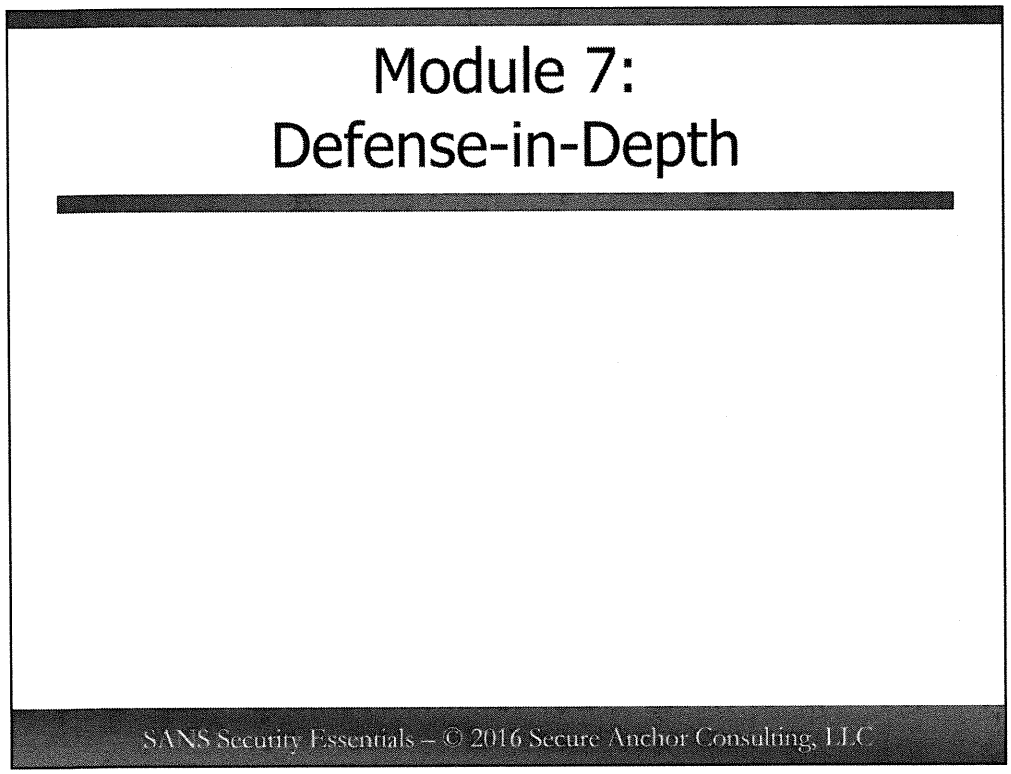
This page intentionally left blank.



This page intentionally left blank.

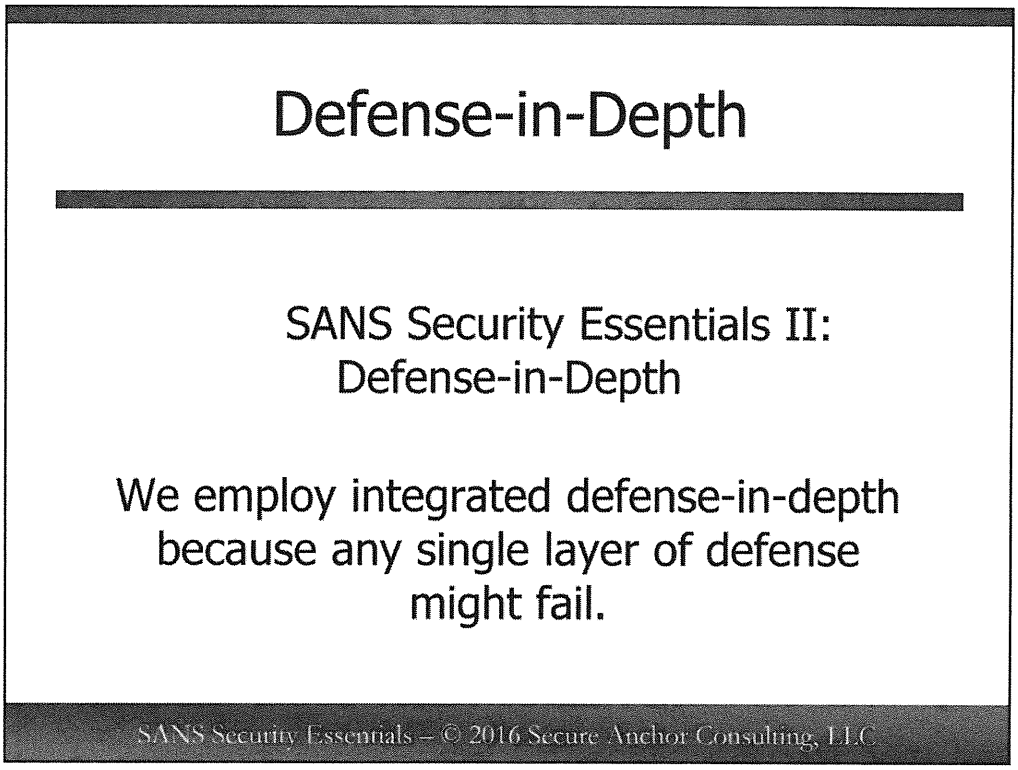


This page intentionally left blank.



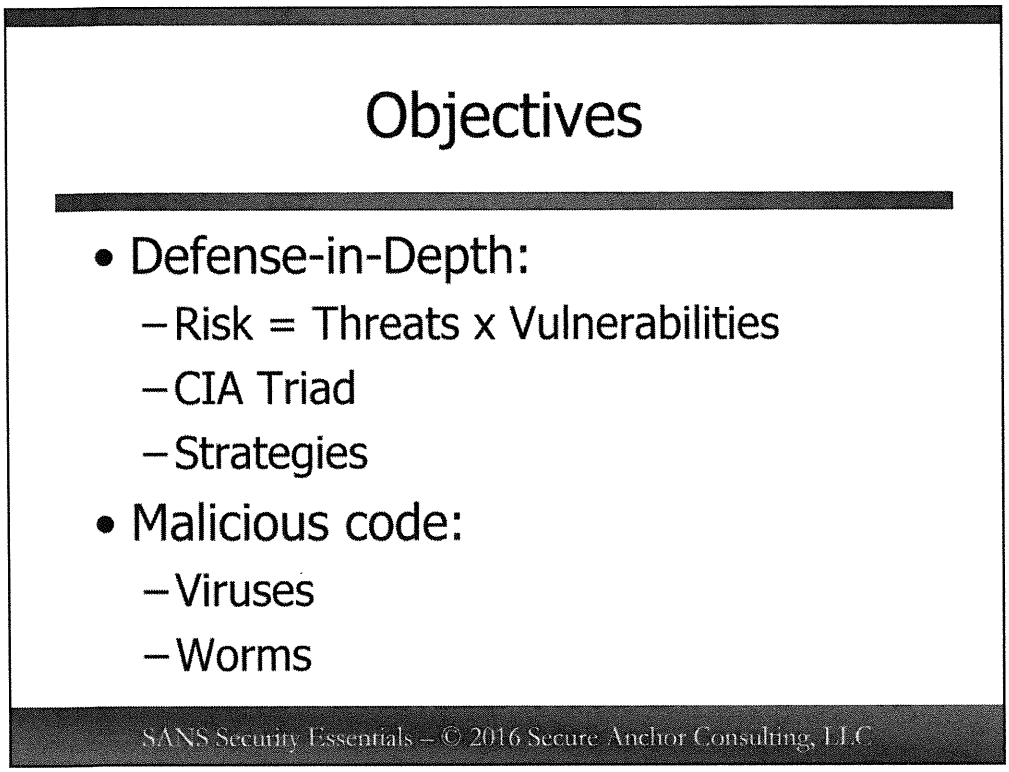
Module 7: Defense-in-Depth

This section intentionally left blank.



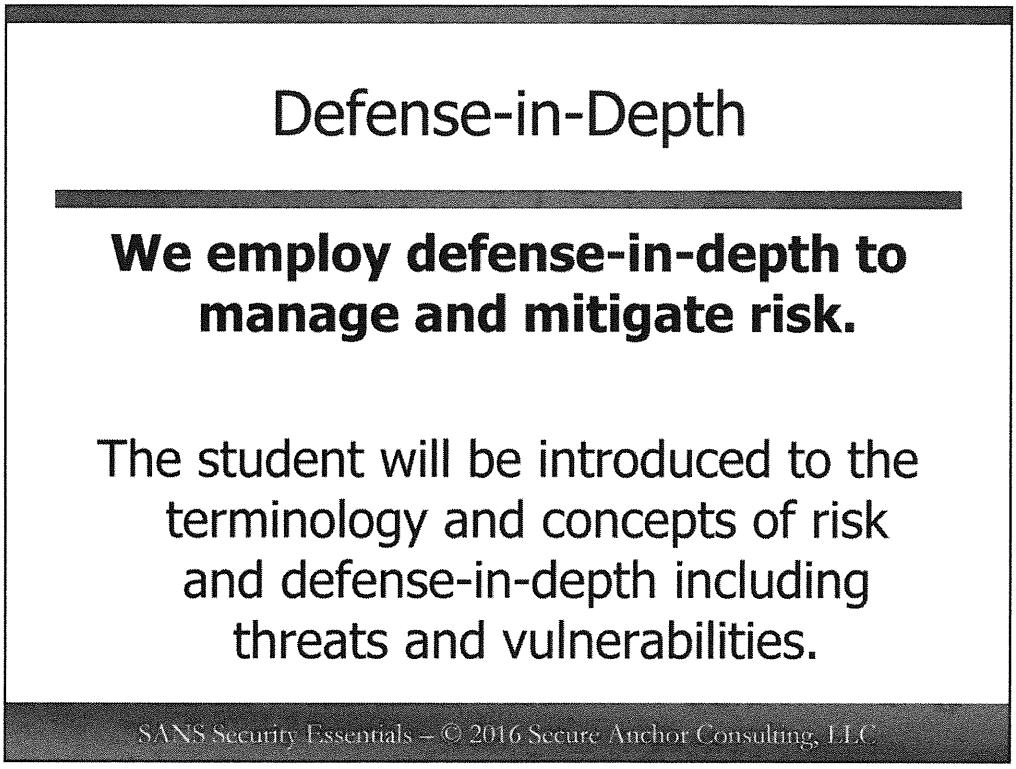
Introduction: Defense-in-Depth

This section intentionally left blank



Objectives

In this module, we look at threats to our systems and take a "big picture" look at how to defend against them. You learn that protections need to be layered: a principle called defense-in-depth. We explain some principles that will serve you well in protecting your systems and use real-world attacks from history, which were wildly "successful," to illustrate them. We examine why the attacks were successful and, more importantly, what measures could have been taken to lessen the impact or to stop them altogether—practical defense-in-depth.



Defense-in-Depth

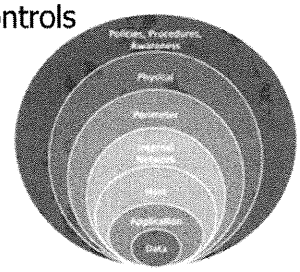
Learning Objective: Defense-in-Depth

The student is introduced to the terminology and concepts of risk and defense-in-depth including threats and vulnerabilities.

What is Defense-in-Depth?

- There is no magic solution when it comes to network security
- Any layer of protection might fail
- Multiple levels of protection must be deployed
- Measures must be across a wide range of controls
- Integrate defense-in-depth

Prevention is ideal but detection is a must; however, detection without response has minimal value.



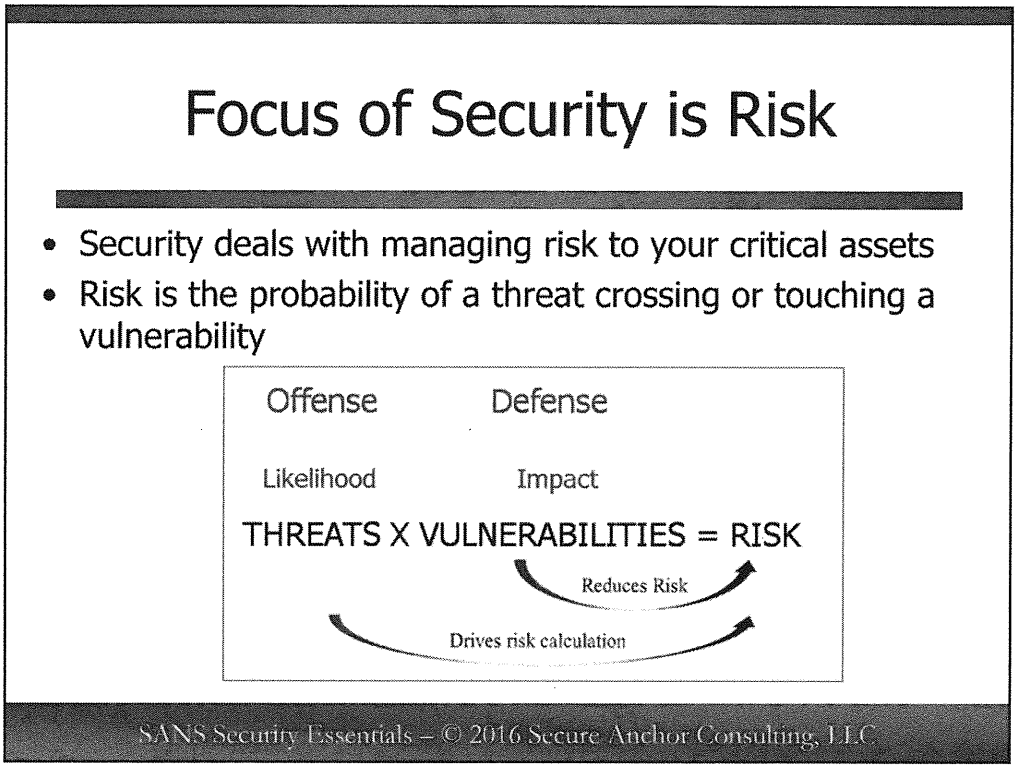
Layers of Defense in Depth

SANS Security Essentials – © 2016 Secure Anchor Consulting, LLC

What is Defense-in-Depth?

Network security is a comprehensive, integrated approach in which multiple solutions are tiered together to accomplish a goal. There is no single security solution that will make an organization secure, because any single measure could be bypassed (and miss an attack all together) or compromised. When protecting any entity, take the President, for example, there are many people, measures, and systems put in place to keep him secure. The same robust approach needs to be applied to your network or any critical asset at your organization.

When it comes to network security, there is no silver bullet. Multiple measures that complement each other must be put in place across a variety of control options. For example, you would deploy a preventive measure such as a firewall, a detective measure such as an IDS, and a deterrent measure such as a guard at your front gate just to name a few. Even if one of the measures failed, the other measures would be able to detect the attack before there was a problem—or catch an attack in action—to minimize the amount of damage caused. You can find additional details on deploying defense-in-depth across an organization http://www.bizforum.org/whitepapers/microsoft-5.htm.



Focus of Security is Risk

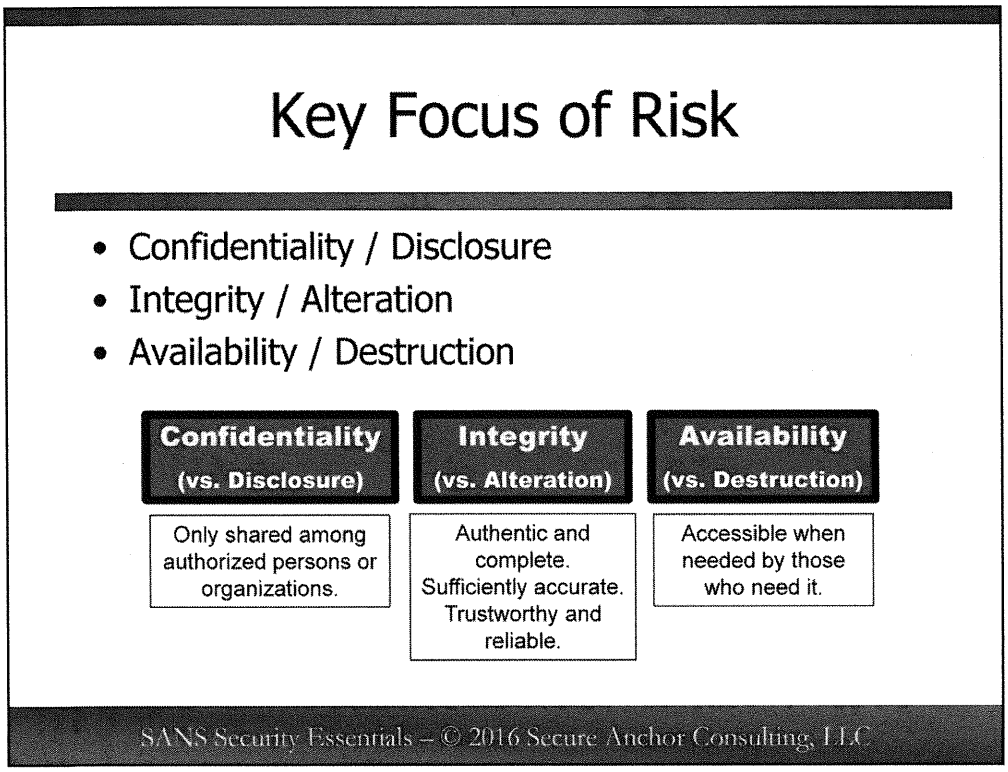
Risks, threats, and vulnerabilities are highly interrelated. Their relationship can be expressed by this simple formula:

Risk (due to a threat) = Threat x Vulnerability (to that threat)

This formula shows that risk is directly related to the level of threat and vulnerability you, your systems, or your networks face. Here is how the formula works:

- If you have a very high threat, but a very low vulnerability to that threat, your resulting risk will be only moderate. For example, if you live in a high-crime neighborhood (thus, high threat) but you keep your doors and windows locked (so you have a low vulnerability to that threat), your overall risk is moderate.
- If you have a high vulnerability to a threat (by keeping your doors and windows unlocked), but the threat itself is minor (by living in a safe neighborhood), once again you have only a moderate risk factor.
- If, however, you have a high level of threat potential (a high crime area) and your vulnerability to that threat is very high (no locks), you have a very high-risk factor.

We must understand that it is impossible to completely eliminate all risk. Therefore, the job of the security professional is to constantly track, manage, and mitigate risk to an organization's critical assets.



Key Focus of Risk

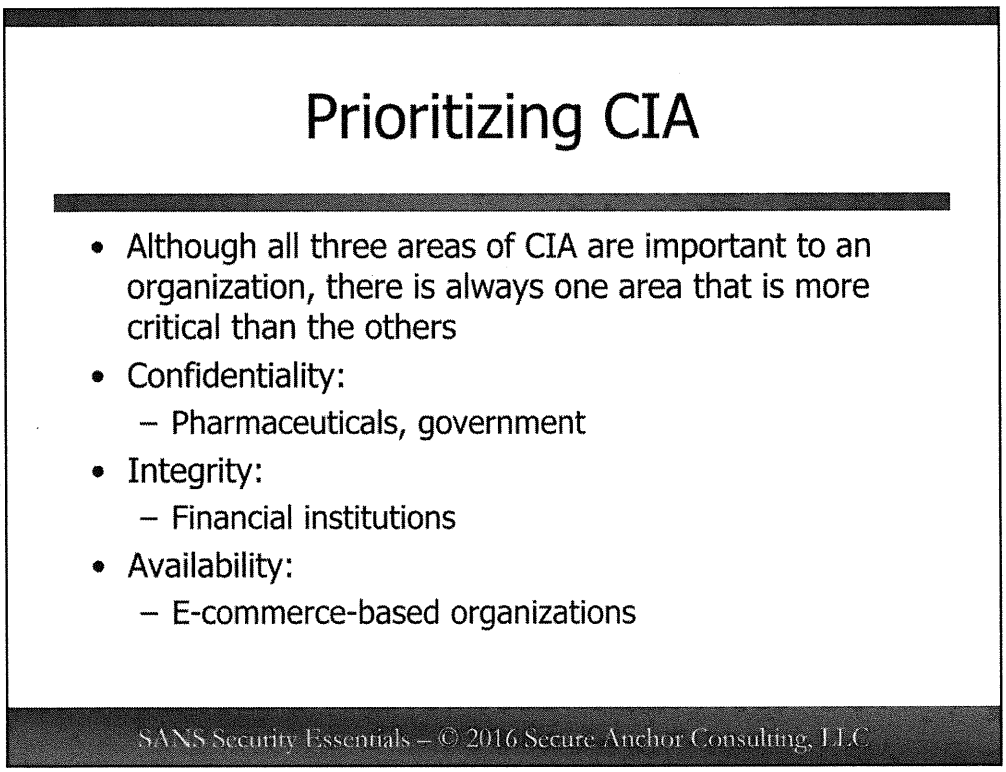
Introduction
Security is all about understanding and controlling risk to your critical assets. More specifically, we are trying to minimize the risk and protect the *confidentiality, integrity,* and *availability* of our critical systems. In deploying measures to protect our systems, we need to deploy a defense-in-depth approach to security.

Confidentiality, Integrity, and Availability
Traditionally, information security professionals focus on ensuring confidentiality, integrity, and availability. Simply "CIA," is "infosec" jargon. These are three bedrock principles about which we will be concerned. When first exploring any new business application or system, it is a good habit to begin by thinking about confidentiality, integrity, and availability—and countermeasures for protecting these, or the lack thereof. Attacks can come against any or all of these.

Example
You have been assigned to oversee the security of your employer's new e-commerce site, its first attempt at conducting business directly on the Internet. How do you approach this? What should you consider? What could go wrong?

Think C-I-A: confidentiality, integrity, and availability. Customers will expect that the privacy of their credit card numbers, their addresses and phone numbers, and other information shared during the transaction be ensured. These are examples of confidentiality. They will expect quoted prices and product availability to be accurate; the quantities they ordered and the prices to which they agreed not to be changed; and anything downloaded to be authentic and complete. These are examples of integrity. Customers will expect to be able to place orders when convenient for them, and the employer will want the revenue stream to continue without disruption. These are examples of availability.

Keep in mind that the dimensions we have been discussing can be interrelated. An attacker can exploit an unintended function on a web server and use the cgi-bin program "phf" to list the password file. Now, this would breach the *confidentiality* of this sensitive information (the password file). Then, in the privacy of his own computer system, the attacker can use brute-force or dictionary-driven password attacks to decrypt the passwords. With a stolen password, the attacker can execute an *integrity* attack when he gains entrance to the system. And he can even use an *availability* attack as part of the overall effort to neutralize alarms and defensive systems so they cannot report his existence. When this is completed, the attacker can fully access the target system—and all three dimensions (confidentiality, integrity, and availability) would be in jeopardy. Always think C-I-A.



Prioritizing CIA

Which pillar of the CIA triad is most important to your organization? At SANS, we rely on our online resources for class registration and online training. Without our online resources, we are unable to provide services to our students. Because we cannot operate without students, our priority is availability.

After availability, the next most important dimension of CIA is integrity. SANS is the most trusted source for computer security training, so our information must be correct. Because the bulk of our information is protected by copyright, even though we have some trade secrets, confidentiality is the least important CIA pillar to SANS.

Different organizations will have different priorities in the CIA triad. Confidentiality is usually very important to healthcare-oriented organizations; and integrity is important to financial institutions. Understanding what the priorities are for your organization is a tremendous help in prioritizing security plans for your organization, from design to incident response.

What is a Threat?

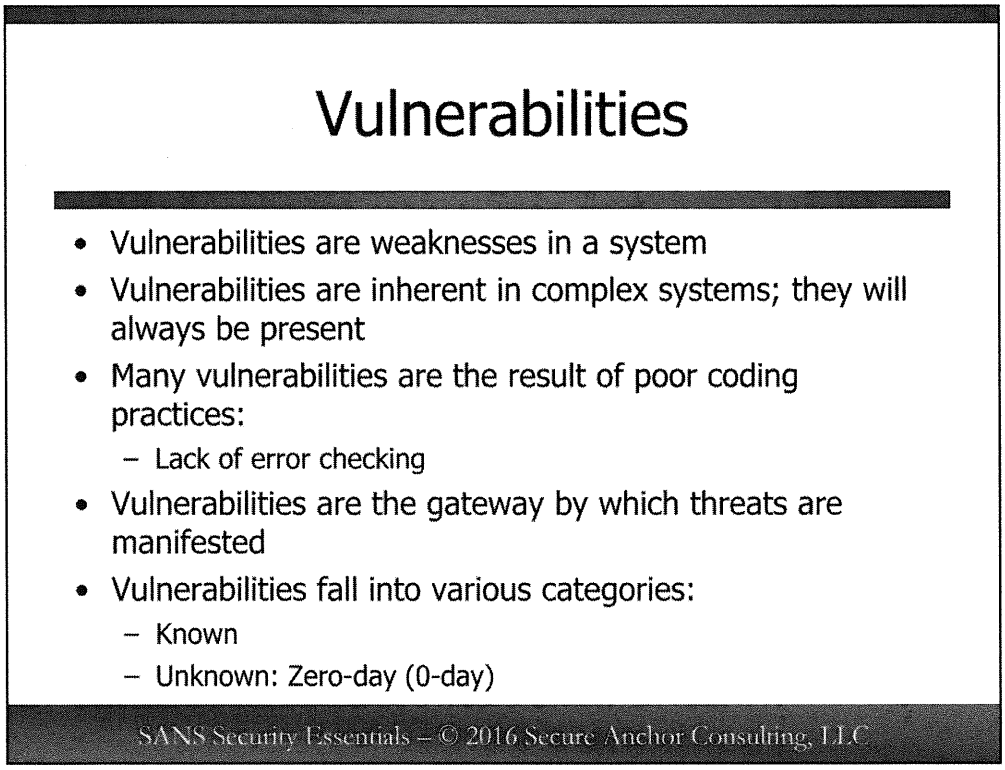
Introduction
Not all the bad things that happen to computer systems are attacks *per se*. There are fires, water damage, mechanical breakdowns, accidental errors by system administrators, and plain-old user error. But all of these are called *threats*. We use *threat models* to describe a given threat and the harm it can do if the system has a vulnerability.

Threats
In security discussions, you hear a lot about threats. Threats, in an information security sense, are any activities that represent possible danger to your information or operation. Danger can be thought of as anything that would negatively affect the confidentiality, integrity, or availability of your systems or services. Thus, if risk is the potential for loss or harm, threats can be thought of as the *agents of risk*.

Types of Threats
Threats can come in many different forms and from many different sources. There are physical threats, like fires, floods, terrorist activities, and random acts of violence. And there are electronic threats, such as hackers, vandals, and viruses. Your particular set of threats depend heavily on your situation: what business you are in; who your partners and adversaries are; how valuable your information is; how it is stored, maintained, and secured; who has access to it; and a host of other factors.

The point is that there are too many variables to ever protect against all the possible threats to your information. To do so would cost too much money and take too much time and effort. So, you need to pick and choose against what threats you will protect your systems. Security is as much risk management as anything. You start by identifying those threats that are most likely to occur or most worrisome to your organization.

Summary
Although there are many threats, the primary threats in most organizations are malware, insider, natural disaster, and terrorism. Insider threat is there to remind us that nation states target companies and systematically acquire their intellectual property.



Vulnerabilities

Introduction
In security terms, a *vulnerability* is a weakness in your systems or processes that could be exploited by a threat. However, simply having a vulnerability by itself is not necessarily a bad thing. It is only when the vulnerability is coupled with a threat that the danger starts to set in.

Example
Suppose you like to leave the doors and windows to your house unlocked at night. If you live in the middle of the woods, far away from anyone else, this might not be a bad thing. In this case, the vulnerability of having no locks is present, but there really isn't any threat to take advantage of that vulnerability.

Now suppose you move to a big city full of crime. In fact, this city has the highest burglary rate of any city in the country. If you continue your practice of leaving the doors and windows unlocked, you have the same vulnerability as you had before. However, in the city the threat is much higher. Thus, your overall danger and risk is much greater. Similar vulnerabilities exist on your computer systems.

Vulnerabilities
Vulnerabilities are the gateways by which threats are manifested. Without vulnerabilities, threats do not pose a risk to the organization. Of course, vulnerabilities do not have to exist solely in *software flaws*. Vulnerabilities can be incorrect configurations, poor physical security, poor hiring practices, and so on. When we couple vulnerabilities with threats, we introduce risks to an organization. A zero-day (0-day) attack is an exploit that is not publicly available and the vendor is usually not aware of the flaw. As you can imagine, these are the most dangerous.

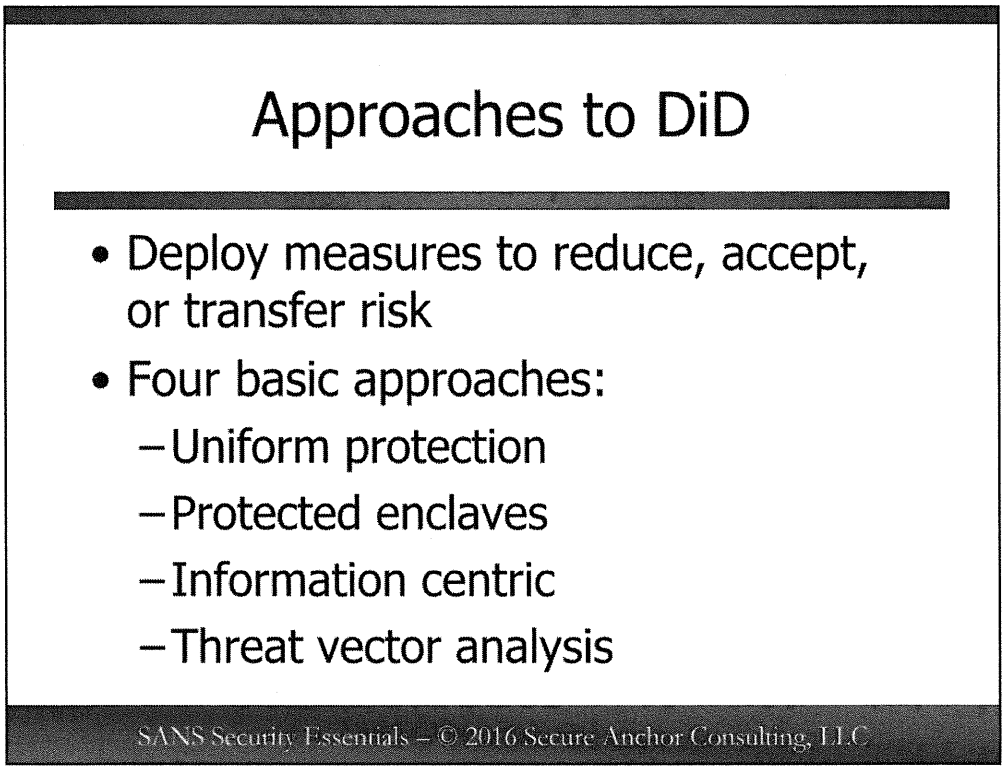
Types of Vulnerabilities

When we talk about identifying vulnerabilities, it is easy to focus on software vulnerabilities, and the difficulty of implementing all-encompassing patch-management systems. However, we are introduced to vulnerabilities in a much broader scale, including electronic vulnerabilities from misconfigured software and problems introduced from the rapid deployment of software patches. We are also asked to manage vulnerabilities of the human type—accidental and intentional attacks against information and its storage, for example. When assessing safety, we are also concerned about vulnerabilities in our physical structures including fire, water, temperature extremes, toxins, and electrical vulnerabilities (loss of power).

Assessing vulnerabilities can be a difficult task to complete—it is easy to assess obvious vulnerabilities, but a thorough assessment can take considerably more time. Only with a comprehensive vulnerability assessment can we accurately calculate our overall risk.

Summary

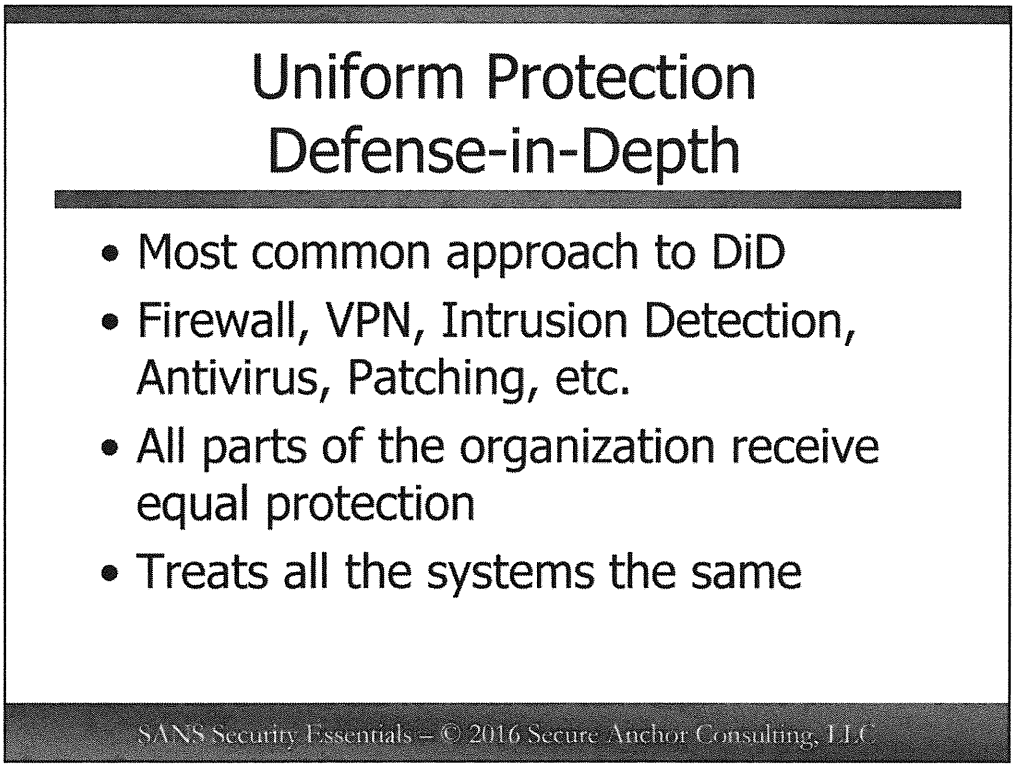
Vulnerabilities can be reduced or even prevented, provided, of course, that you know about them. The problem is that many vulnerabilities lay hidden, undiscovered until somebody finds out about them. Unfortunately, the *somebody* is usually an attacker. The attacker always seem to find out about vulnerabilities long before the organization that needs to protect against it.



Approaches to DiD

The concept behind defense-in-depth (DiD) is simple. The picture we have painted so far is that a good security architecture, one that can withstand an attack, has many aspects and dimensions. We need to be certain that if one countermeasure fails, there are more behind it. If they all fail, we need to be ready to detect that something has occurred, clean up the mess expeditiously and completely, and then tune our defenses to keep it from happening to us again.

We examine four approaches to defense-in-depth.

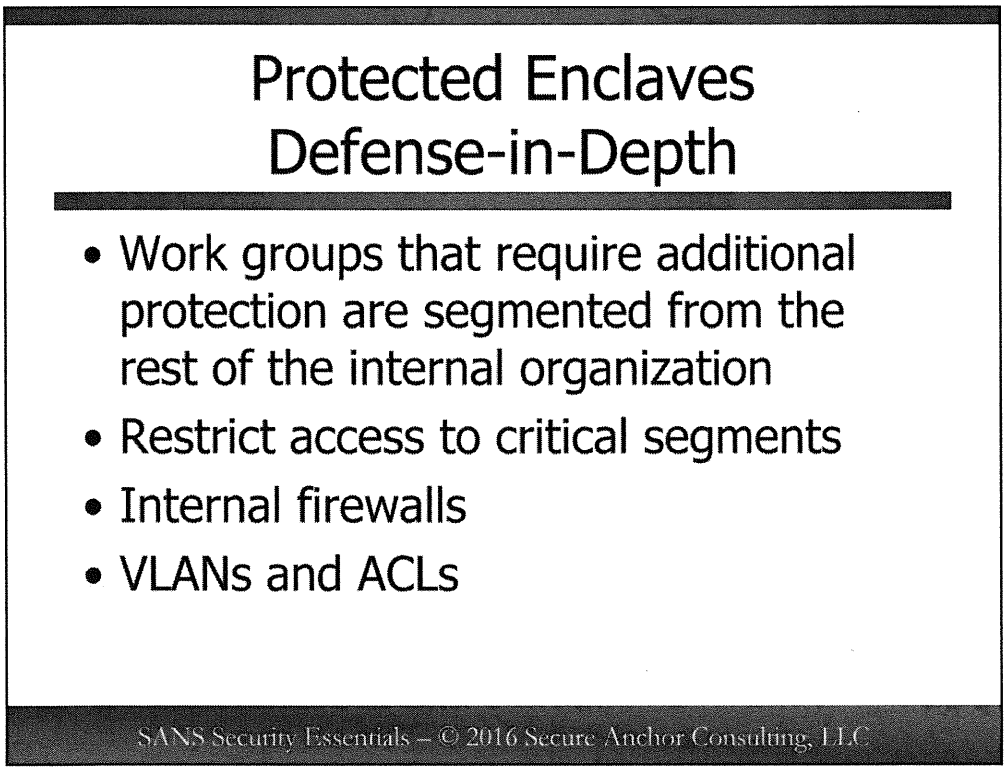


Uniform Protection Defense-in-Depth

Uniform protection treats all systems as equally important. No special consideration, or protection, is given to the critical intellectual property of an organization. As a result, this approach can be more vulnerable to malicious insiders, because the systems are not separated or categorized within the network.

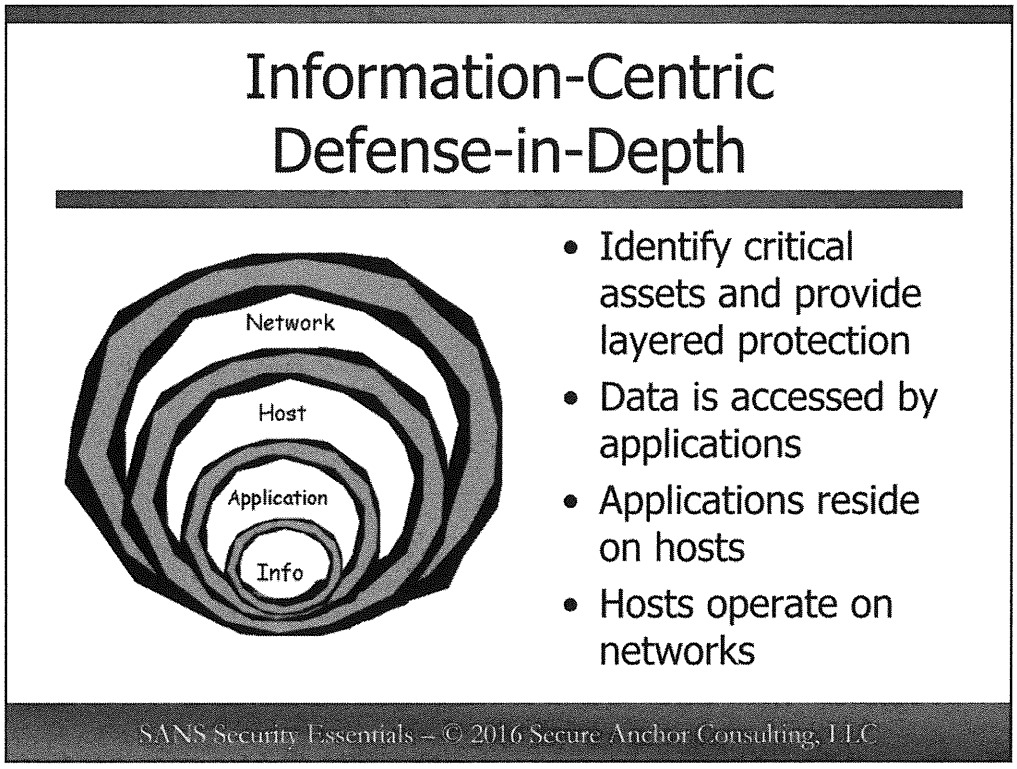
The majority of attacks succeed because they take advantage of well-publicized vulnerabilities for which exploits have been created. The best answer is to patch the systems, but this takes time. Of all the approaches to defense-in-depth, this one can be the weakest, unless you have a good uniform protection design.

This is also by far the most common approach and usually the starting point for most organizations.



Protected Enclaves Defense-in-Depth

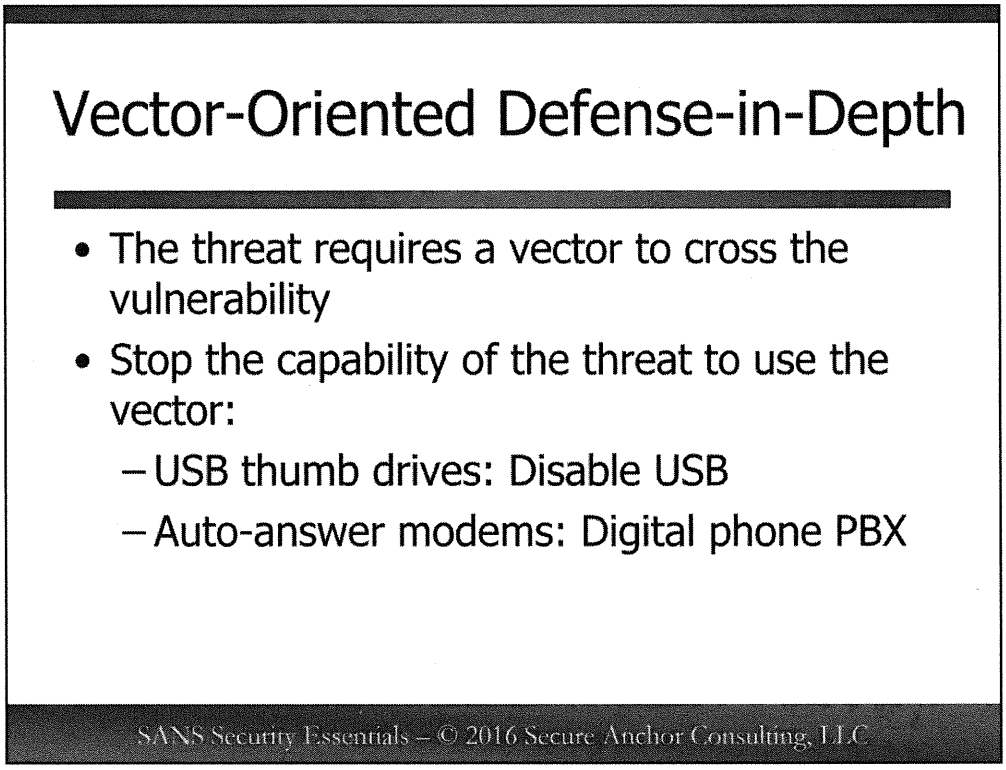
Protected enclaves involve segmenting your network. This can be done by implementing many VPNs across a single network, VLAN segmentation of switches, or firewalls to separate out the network. This is a simple, yet effective, technique. Reducing the exposure or visibility of a system can greatly reduce the impact malicious code can have. For example, if you have 5,000 systems on a network and a virus infection breaks out, it could spread to all systems. However if you create separate segments with 100 systems per segment, the virus would now impact only a small percent of your systems and cleanup and damage would be minimal.



Information-Centric Defense-in-Depth

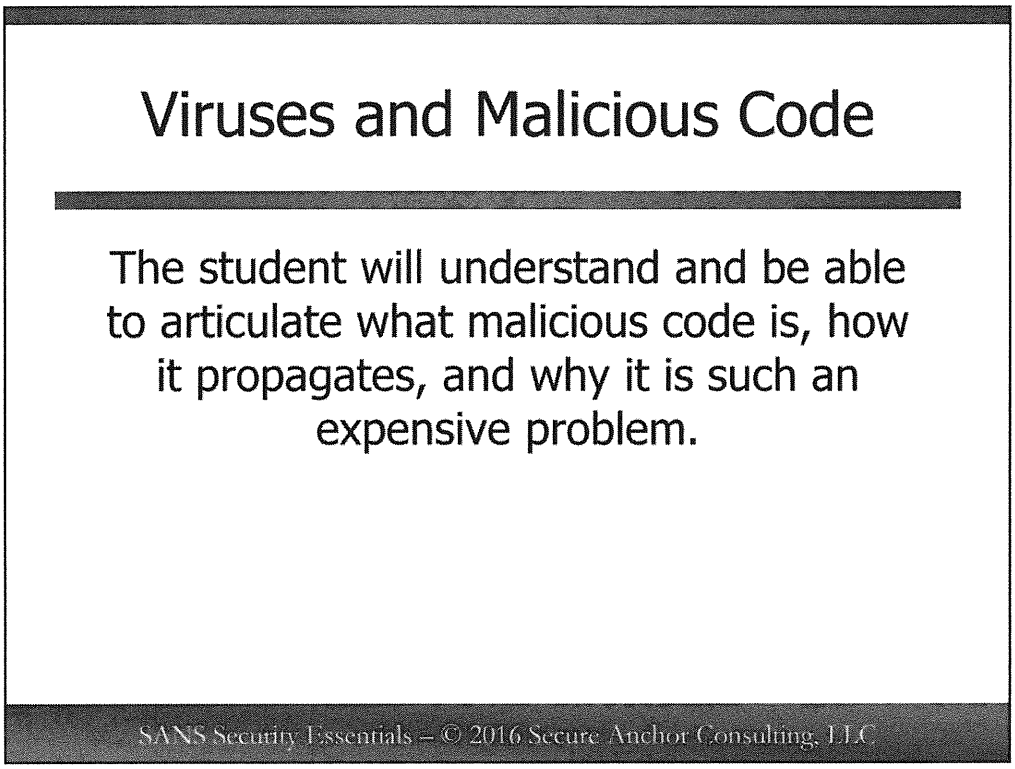
This slide shows another way to think of the defense-in-depth concept. At the center of the diagram is your information. However, the center can be anything you value, or the answer to the question, "What are you trying to protect?" Around that center, you build successive layers of protection. In the diagram, the protection layers are shown as blue rings. In this example, your information is protected by your application. The application is protected by the security of the host it resides on, and so on. To successfully get your information, an attacker would have to penetrate through your network, your host, your application, and finally your information protection layers.

Information-centric defense starts with an awareness of the value of information within an organization. Identify the most valuable information and implement controls to prevent non-authorized employees from accessing it. A good starting point is to identify your organization's intellectual property, restrict it to a single section of the network, assign a single group of system administrators to it, mark the data, and thoroughly check for this level of data leaving your network.



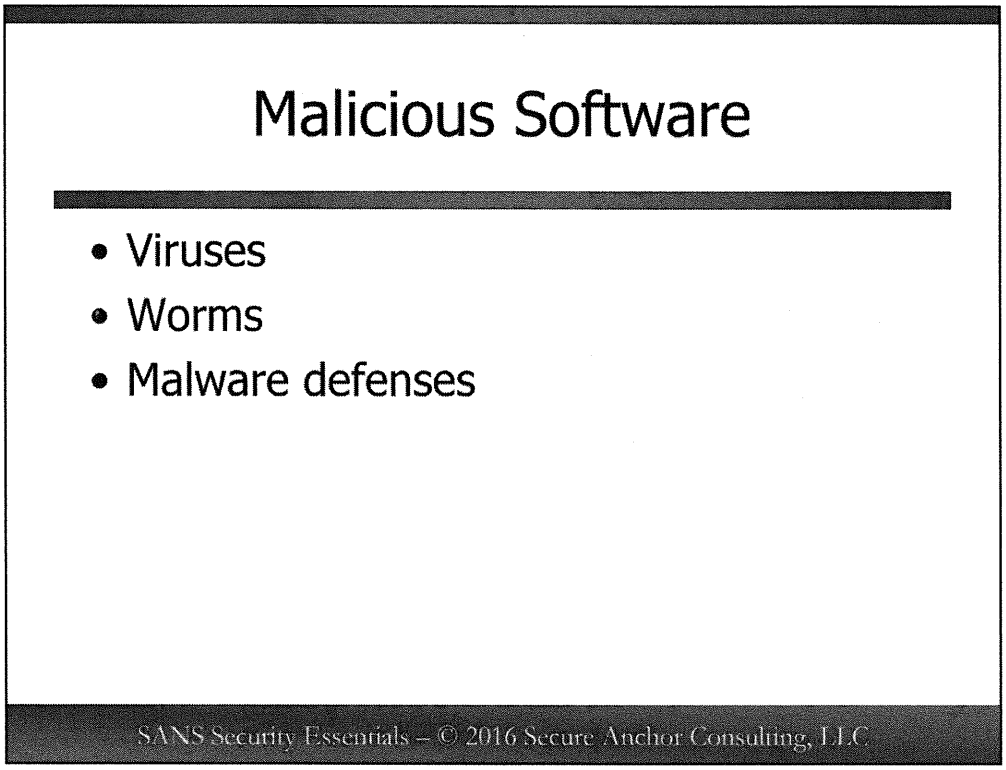
Vector-Oriented Defense-in-Depth

Vector-oriented defense-in-depth involves identifying various vectors by which threats can become manifested and providing security mechanisms to shut down the vector—for example, disabling USB thumb drives and floppy drives. In an ideal case, you want to remove the vulnerability so the attack has no chance of success. However, in many cases, the vulnerability cannot be removed and there is an active threat. When you have a high risk and you cannot remove the vulnerability, you would mitigate it by removing the vector or avenue the threat would have to use to compromise your system.



Viruses and Malicious Code

This section intentionally left blank.



Malicious Software

In this section, we look at malware, the threats it presents, the types of malware being propagated (including hybrid threats), and the lessons we can learn from historical attacks. Key topics include:

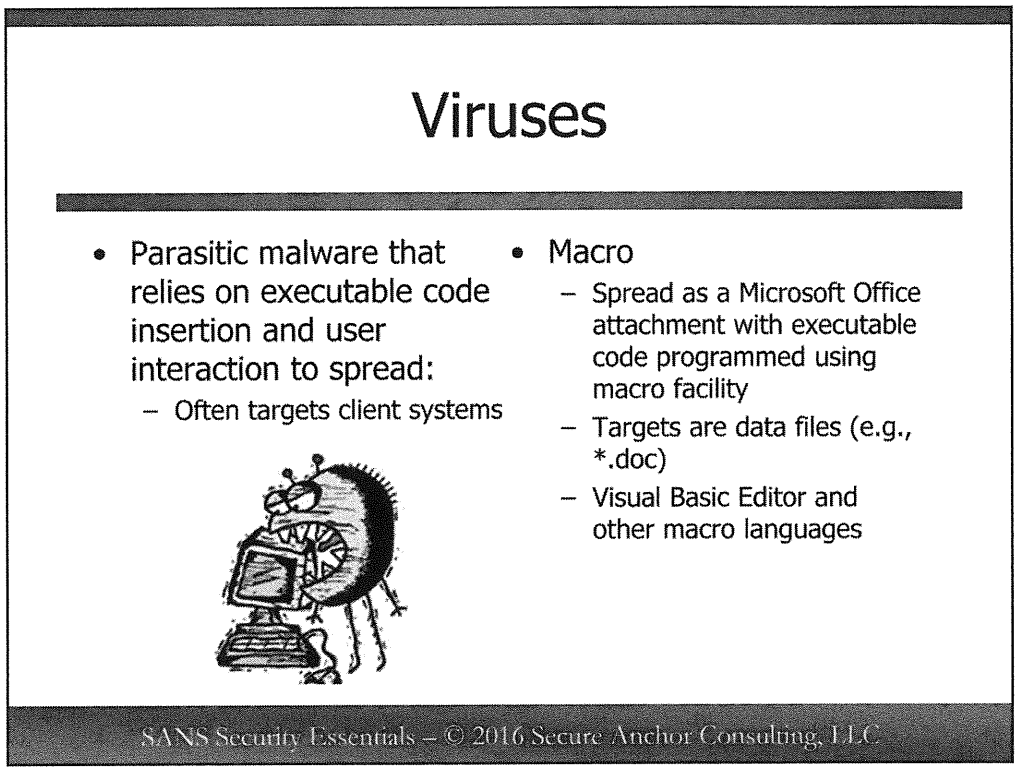
- Malicious software taxonomy
- Viruses and how they work
- Worms
- Malware defenses

We begin with the discussion of malware taxonomy by describing popular categories of malicious code.

Malware is a generic term that refers to software that was written with malicious intent and performs its actions without the user's permission.

The type of malware we discuss is sometimes referred to as *malicious mobile code*, because the defining feature of these types of malware is that they are meant to replicate from computer to computer, most of the time as a function of their own coding. Other types of malware, like ping of death tools, packet-capturing programs, or password crackers, are malicious; but their intent does not include replication.

To identify threats associated with malware, let us explore differences and similarities between categories of malicious code. Understanding this taxonomy will help you devise appropriate defenses against malware attacks.



Viruses

Introduction
A *virus* is a malware specimen that has the ability to replicate and possesses parasitic properties. A virus is a parasite because it cannot exist by itself; instead, it must attach itself to another executable program, data file, or area of coding (like a boot sector).

Virus Payload
The *payload* of the virus executes when the user or computer launches the code to which the virus is attached. Instructions in the payload allow the virus to replicate and give it the opportunity to do its author's bidding. Some viruses do little more than spread and serve as a nuisance. Others can do serious damage, such as destroy data or degrade system performance. Even if a virus isn't intentionally designed to cause damage, merely infecting other program code is damaging by itself. Infections can cause errors, lockups, and operational problems, and at the very least the virus is taking up CPU cycles.

Virus Types
Another way to have malicious code execute when the machine starts up is to place it in the boot sector of the computer's disk. Every disk has a boot sector, regardless of whether or not it is actually bootable. When a PC is powered up, it looks for boot information in the order dictated by the machine's BIOS. If any of the media in the drives specified by the BIOS has an infected boot sector, the infection will get transferred to the boot drive. After the infection is complete, malicious code will get loaded into memory at startup. A malware specimen that places malicious code into the boot sector is called a *boot record infector*.

Historically, the vast majority of boot record infectors have been viruses. That is why you are much more likely to hear about some "boot record virus," and will rarely (if ever) hear the term "boot record worm."

Macro Viruses

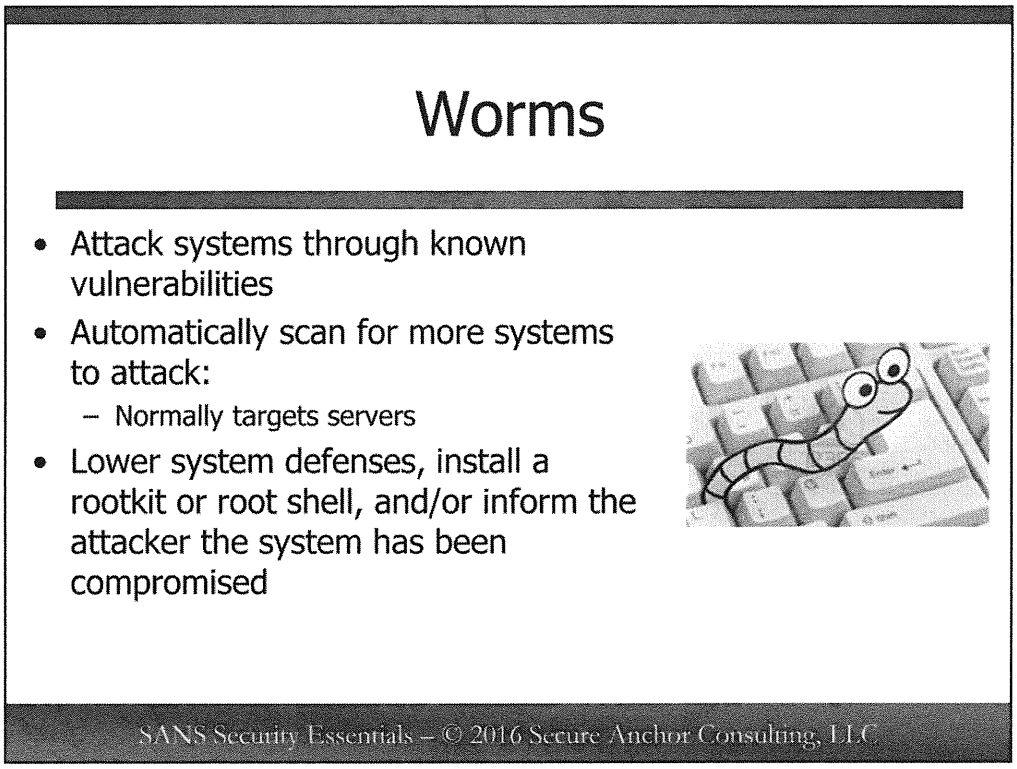
A macro virus (or a *macro worm*, for that matter) is implemented using instructions that can be interpreted within applications, such as a word processor and a spreadsheet. Unlike program viruses, which target executables, macro viruses usually target application files. Capabilities of a macro infector are limited only by the macro language used to implement it. Microsoft Office macros written in Visual Basic are very powerful and can access all features of the host application and numerous features of the underlying OS. Many MS-Office viruses are able to infect multiple applications (for example, Word and Excel), and are cross-platform. Some MS-Word macro viruses can infect both IBM-compatible and Macintosh computers.

Program Infectors

Malicious code that infects by attaching itself to existing program files is called a *program infector* (sometimes also known as a *file infector*). A program infector is activated when the executable that hosts malicious code is launched. The malware specimen is then loaded into memory and is ready to perform its author's bidding. Program infectors are usually attached to files with .com or .exe extensions, but can infect other types of program executables (such as SYS, OVL, SYS, SCR, and so on) and interpretable files as well.

Summary

Viruses will continue to evolve and be a threat because it is a primary vector of attacking desktop systems. It is important to note that many viruses target Windows systems because that is the majority of deployed desktop operating systems. However there are viruses that target other operating systems.



Worms

Introduction

A worm is a self-contained malware program that has the ability to copy itself without parasitically infecting other host code. It might rely on the presence of other code to help it get launched, but it doesn't modify other host code in order to replicate and execute. If you recall, the payload of a virus was triggered when the victim launched the infected host code. A worm, on the other hand, can be much more autonomous. Some worms can activate themselves without requiring action on the user's part; other worms still require a user to do something like open an e-mail. Also, unlike a virus, a worm is able to function without having to attach itself to another program. The self-sufficient nature of a worm often allows it to infect a larger number of systems. This trait makes worms very effective attack tools, especially when they are able to propagate over the network. As a result, worms now seem to have surpassed viruses in popularity.

How Worms Work

You can think of a worm as a virus on autopilot—it typically doesn't need a user to do anything to spread. Instead, most worms use some variation on the following algorithm:

- The worm scans a large number of systems for one or more vulnerabilities. After it has found a system that has a vulnerability it recognizes, it attacks the remote system with a tool written to exploit that hole.
- After breaking in, the worm tells the remote system to download a fresh copy of the worm code (either from the attacking system itself or from a web server) and tells it to run some commands that perform some actions on the attacked system.
- Step 1 is repeated—that is, the newly infected system starts scanning for even more systems to attack.

Worms spread exponentially as long as there are many vulnerable machines left to compromise. Say that each scan resulted in just five newly infected systems. Only one system would be infected in the beginning (the worm has to start somewhere). After the first round of scanning, five more systems would be infected. Each of those systems would then infect five other systems, meaning that 5 * 5 = 25 more systems would be compromised in the second round.

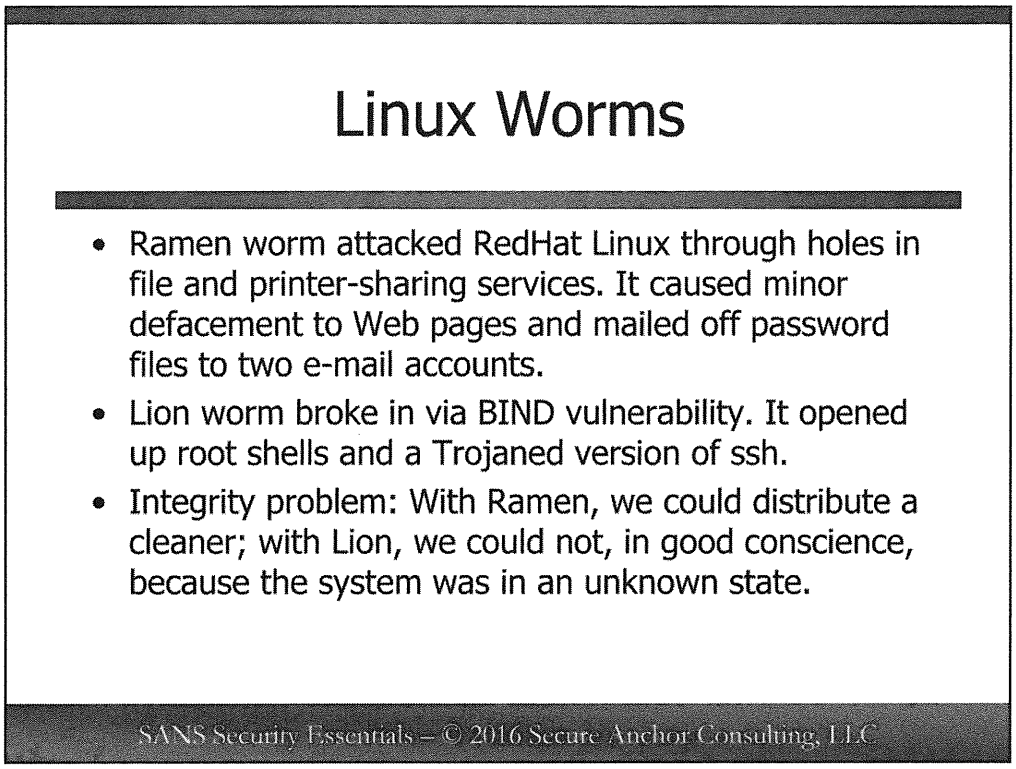
The worm can easily claim thousands of systems within five or ten minutes of the original infection. In extreme cases, a worm can claim hundreds of thousands of systems in as little as a few minutes.

Impact of a Worm
The worm might commonly do any of the following on the attacked system:

- Let the original attacker know about this newly infected system by sending her an e-mail message with this system's address and a copy of the system password files for cracking later.
- Open up backdoors for easy access.
- Deface Web pages on the system.
- Replace system binaries such as netstat, ls, and ps, so the administrator cannot tell whether the system is infected.
- Install kernel modules that modify the operating system's kernel or add services, such as hiding of malicious processes from common tools such as ps.

Worms can do almost anything to the system—even delete all the files on the system. They rarely get that destructive, because they do not want to tip off the administrator that something's wrong. Besides, a working compromised machine is useful to the attacker; why destroy it?

Summary
Worms generally reduce the availability of a system or network. The act of scanning for additional systems to infect can completely saturate the victim's Internet connection. If the worm infects and re-infects a system, it might tie up the processor or fill the drive. It also might shut down needed services. Although we usually think of viruses and worms as attacking Windows systems, they can affect Unix systems as well.



Linux Worms

Introduction
Worms can impact any type of operating system including Linux systems. In addition to worms impacting many types of operating systems, new worms are much more stealthy than earlier worms. Previously, worms would have an obvious component such as defacing a web site, so they were easier to identify. However, today, it is much harder to detect a worm on a system.

Ramen
Ramen looks for three specific vulnerabilities in certain services bundled with RedHat Linux 6.2 and 7.0. The vulnerable services are WU-FTPD, LPRng print spooling, and rpc.statd (part of NFS). Note that these services were *known* to be vulnerable at the time Ramen was released. Patches for each of them had been available for at least three months. Systems with these patches applied were not vulnerable to this worm.

After Ramen breaks into a system, it:

- Replaces all the Web pages on the system with one that says, "Hackers looooooooooooove noodles," and displays an image of a package of Ramen noodles
- Mails the password files to two e-mail accounts, presumably owned by the attacker
- Replaces ps and netstat with versions that hide the existence of Ramen
- Installs and runs a Stacheldraht distributed denial of service agent
- Closes the holes it used to break in

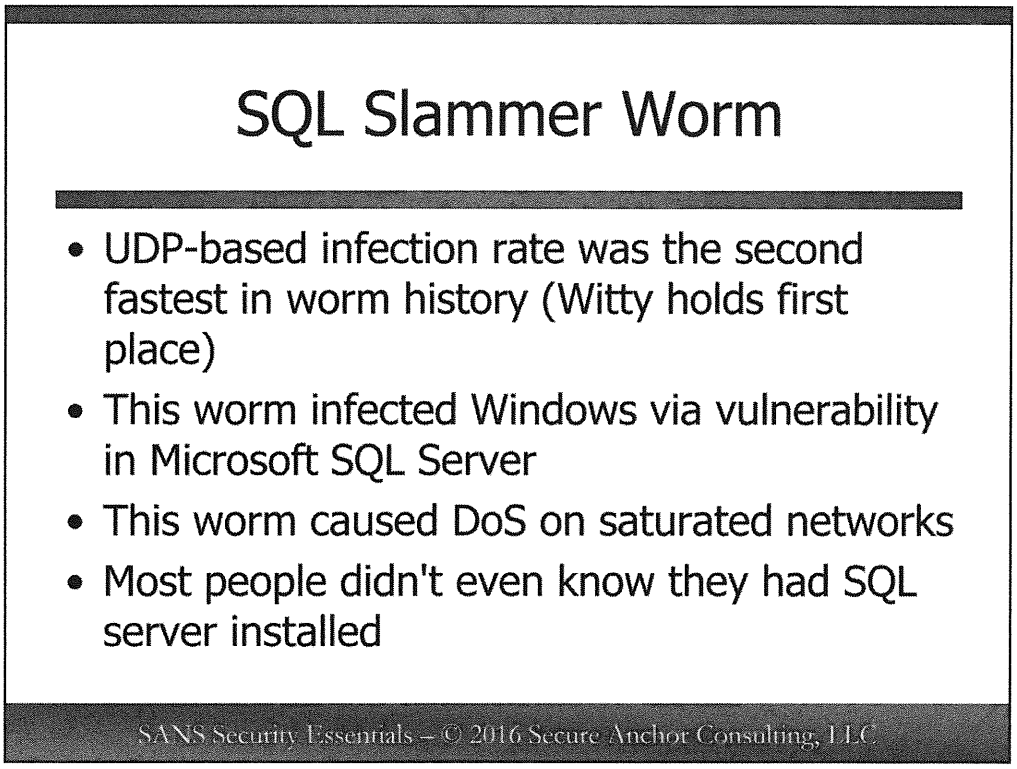
Protecting the System
Closing the hole might seem strange—why would a worm want to *close* a hole on the attacked system? There are two main reasons. The most important is that a worm needs some way to stop itself from infecting a given system more than once. If it didn't, the worm would go on forever, infecting and re-infecting systems, eventually chewing up all the resources of the given systems and their networks. The original Morris Internet worm failed to correctly check whether it had already infected a system and thus effectively crippled the Internet. Closing the holes is the easiest way to prevent this.

The second reason is that the attacker might not want other attackers to get into the system. The original attacker might want to build up her own collection of compromised systems and not want anyone else to steal any of them.

Lion
The Lion worm, uses the same approach as Ramen. It is a collection of shell scripts and executable programs that attack Linux systems. Lion uses vulnerability in the BIND name server to break in. Then it gets destructive:

- It does not close the hole it used to get in.
- It opens up root shells on two ports and starts a Trojaned version of ssh (one with a backdoor password stored in /etc/ttyhash).
- It replaces 12 system tools with versions that hide its presence.
- Depending on the version, it might install the t0rn root kit or the Tribe Flood Network 2000 (TFN2K) distributed denial of service agent.

Summary
With the Ramen worm, because it did not open a back door, SANS was able to create and distribute a removal tool. Lion left a back door and so we were faced with the ethical problem of whether to release a cleaner for a system in an unknown state.

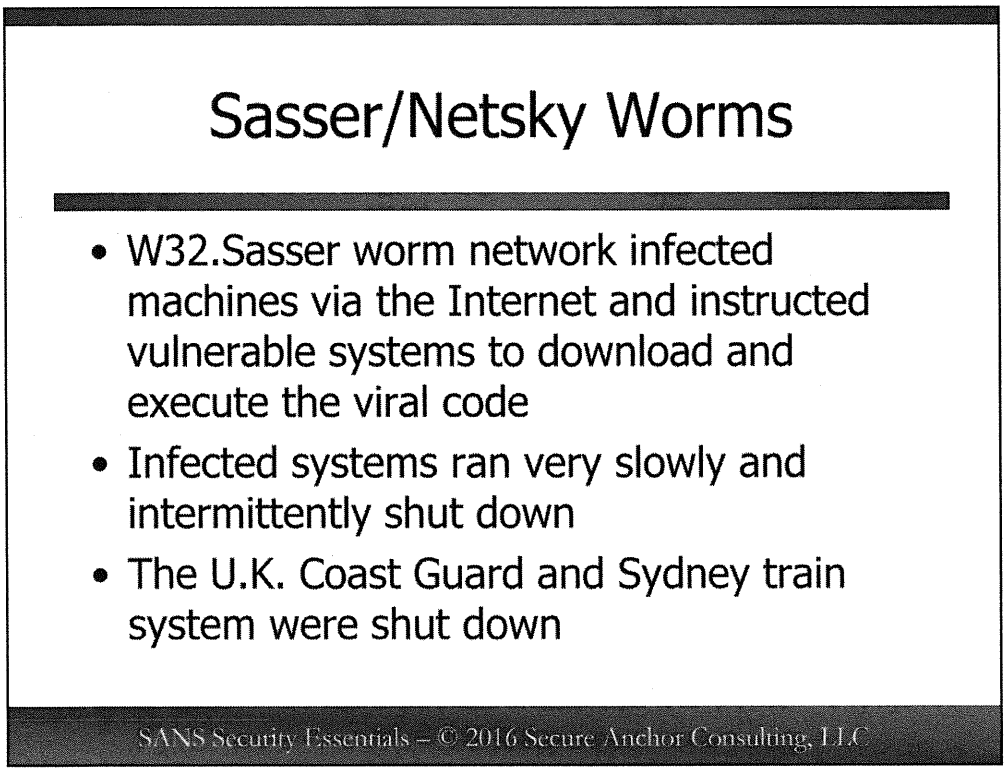


SQL Slammer Worm

The SQL Slammer worm targeted UDP port 1434 to exploit a buffer overflow vulnerability in the Microsoft SQL Server and Microsoft Desktop Engine (MSDE). The worm has an aggressive propagation mechanism and caused a DoS of local networks.

The most interesting thing about SQL Slammer is that most infected machines were not SQL servers. The Microsoft Desktop Engine (MSDE) is built in to a large number of Microsoft products, such as Microsoft FrontPage, Microsoft .NET Framework, Microsoft Office 2000 and XP, as well as several non-Microsoft products.

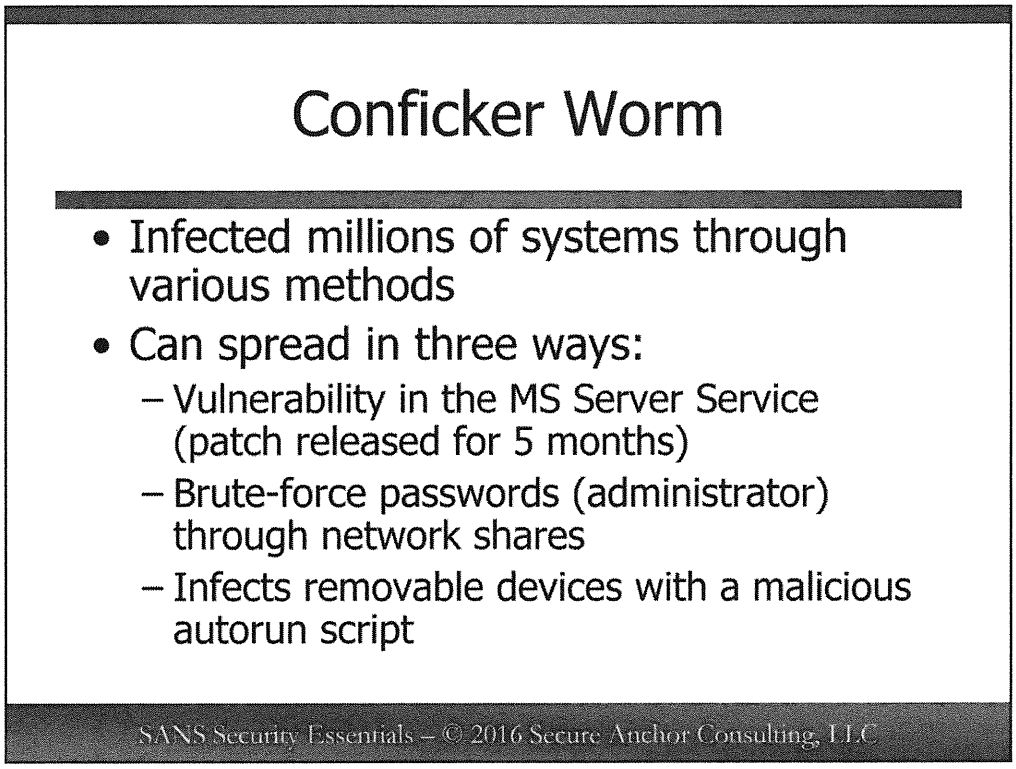
Again, good patch management practices would have substantially reduced the impact of this worm.



Sasser/Netsky Worms

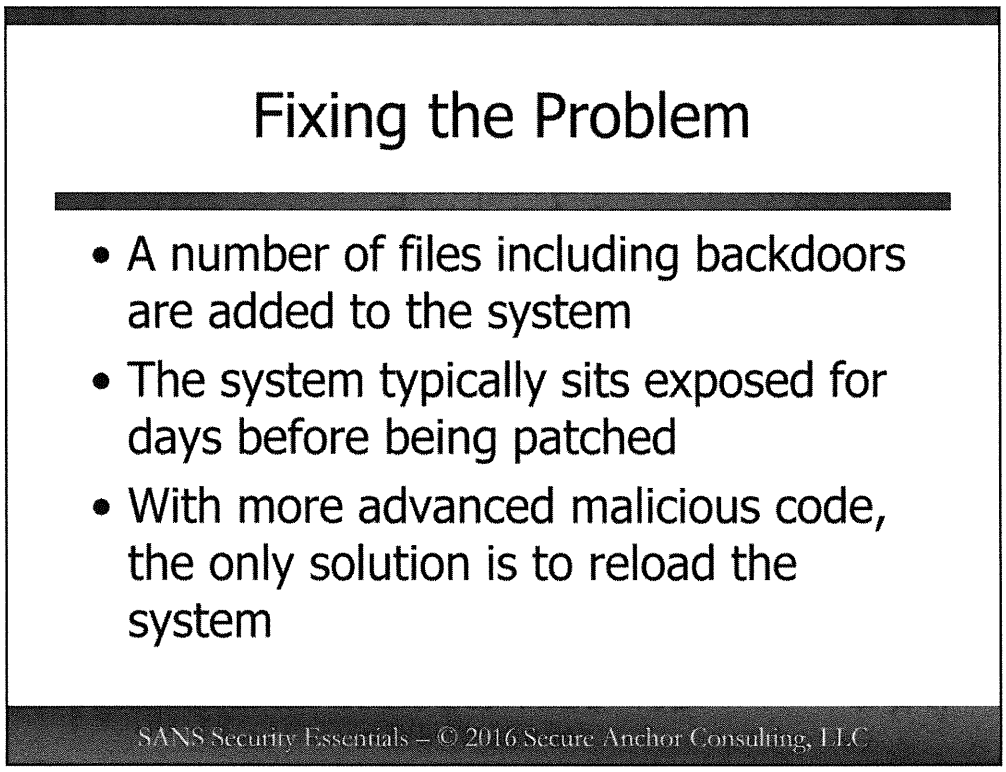
Sasser shows you that if an attacker performs careful planning, the impacts of a worm can be devastating in a very short period of time.

Called a network telescope, Sasser infected over 500,000 machines in three days. More than 300,000 commuters were stranded when trains in Sydney did not run on time; not only did their systems fail, because they did not install the patch, but their backup systems failed as well.



Conficker Worm

The Conficker worm shows us that attackers continue to take advantage of systems that are not properly secured. According to the Internet Storm Center, it can spread in the following ways: "First, it attacks a vulnerability in the Microsoft Server service. Computers without the October patch can be remotely attacked and taken over. Second, Conficker can attempt to guess or brute-force Administrator passwords used by local networks and spread through network shares. And third, the worm infects removable devices and network shares with an autorun file that executes as soon as a USB drive or other infected device is connected to a victim PC."



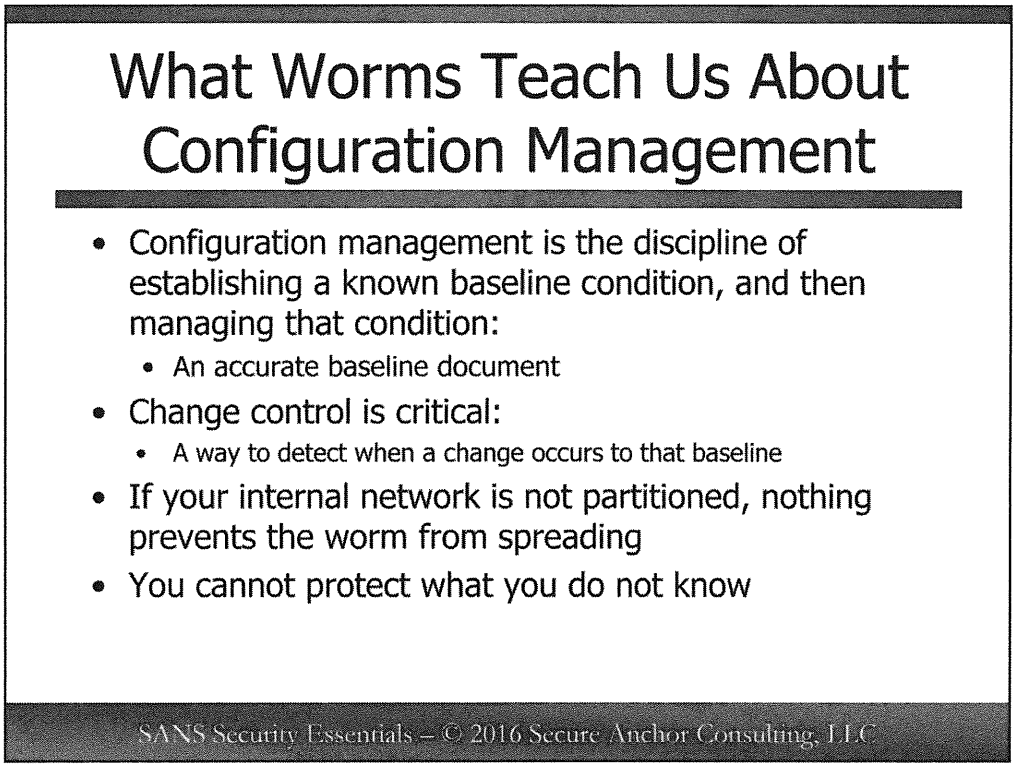
Fixing the Problem

After a system has been infected with a worm, we are left with a *giant integrity problem*. A number of files, including backdoors, are added to the system. The system typically sits exposed for days before being patched.

What Have We Learned?

Most worms are so successful because of the prevalence of undefended perimeters, operating systems left unchanged and certainly unpatched since their default installation, and one application automatically installing another. These were failures to practice defense-in-depth against known "vulnerabilities."

The point isn't how the worm worked, but why it was successful; and, more importantly, what measures could have lessened its impact or stopped it altogether. How might this have been different if defense-in-depth were being practiced? Systems could have been properly configured and kept patched, and services could have been separated onto different systems. A baseline could have been established so as to detect any alterations, instead of being blind to what might have transpired through the backdoor. Are you taking these steps today?



What Worms Teach Us About Configuration Management

The primary attacker strategy is to scan, looking for a vulnerable system, and then establish a beachhead or foothold by compromising that system. Then the vulnerable host that got compromised can be used to attack other systems, either in the same facility or in other organizations. This is one reason for saying that "risk assumed by one is shared by all."

In the early stages of protecting a site, a perimeter defense, such as a firewall, is about the only reasonable thing you can do. Although chokepoint defenses such as firewalls can yield some protection to internal systems, they can be circumvented in a number of ways, so the organization turns its focus to identifying and fixing vulnerabilities—what we call *hardening* systems. It takes a lot of energy to get to a known, reasonable configuration. How do you maintain that state?

Configuration management is the discipline of establishing a known baseline condition, and then managing that condition. Now, of course, change is inevitable, and change generally is thought of in two major categories: repairs and improvements. Although vulnerabilities might occur while fixing something, they are far, far more likely to occur when deploying something new. We can label adding software, upgrades, new features, and new systems as "new construction."

Before you can do new construction, you need a building permit, and part of the building permit process is a design review and an inspection. The building permit process gives the organization an opportunity to ensure that the new construction introduces no new vulnerabilities into an organization. Of course, it's not foolproof, but it sure is better than not doing anything at all. And it has a lot of benefits! Perhaps the most significant is that the earlier in the development lifecycle you identify a problem, the cheaper it is to fix it. The rich question is this: Can you detect that new construction is taking place (regardless of whether or not a building permit has been issued)? Said differently, can you detect a change to the network (or computer) infrastructure if no one has indicated that the change is (or will be) taking place? If you can't detect the change, how can you inspect it? If you can't inspect it, how can you assess and manage the risk? You can't. To manage your configuration, you need two things:

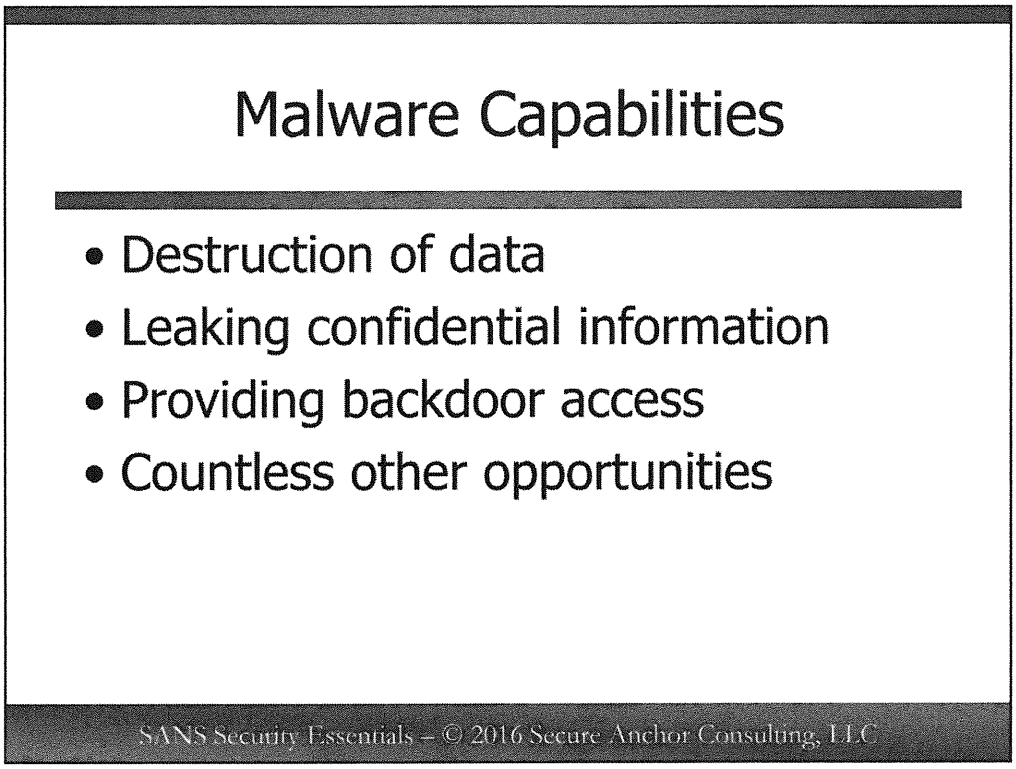
- An accurate baseline document
- A way to detect when a change occurs to that baseline

If either is missing, you may well have a tough time managing the configuration.

What is involved in developing and maintaining the baseline document? Typical components include mapping the network and conducting vulnerability assessments against the computers on the network.

All improvement starts with one person willing to exert the energy needed to make a difference. If your organization doesn't have configuration management, and doesn't plan to ever implement configuration management, you can still implement configuration management on the things for which you are responsible.

For configuration management to be truly successful, we need instrumentation, such as system scanners, network mapping, and vulnerability scanners to detect unauthorized change—and we need to use those tools on a scheduled basis. Only a facility with an accurate baseline is likely to practice anomaly detection to find attacks for which there are no known signatures.



Malware Capabilities

Introduction
Anything that can impact the confidentiality, integrity, and availability of critical information represents a risk to security. Worms and viruses represent a threat because they can access confidential records, and, possibly, retain your records for future use. From a business perspective, there is usually monetary loss associated with losing critical files, revealing sensitive information, or providing unauthorized access to internal systems.

Destroying Data
Destroying data is one of the most insidious actions that a malware specimen can take after infecting a system. For example, the CIH virus had a particularly destructive payload. CIH was programmed to activate every year on April 26, at which point it overwrote data on the computer's hard drive. Additionally, the virus attempted to overwrite the flash-BIOS of the infected system, often rendering the computer unusable. CIH is also known as the "Chernobyl virus," because April 26 marks the anniversary of the nuclear plant disaster that occurred in Chernobyl, Ukraine, in 1986.

If you lose data as a result of a malware infection, your most practical means of recovery is to retrieve files from backup. If backups are not available and lost data was very valuable, you might be able to restore it via low-level forensic recovery techniques. Such methods, however, tend to be time-consuming and expensive. Unfortunately, destruction of data is only one danger associated with a malware infection.

Leaking Information
The possibility that a malware incident led to information leaking to unauthorized parties can be as devastating as the destruction of data. You might recall that the Melissa virus, often resulted in sensitive Word documents being sent to recipients listed in the victim's e-mail address book. Of course, a document is only one type of information whose confidentiality can be compromised by malware. The Caligula virus was programmed to locate the victim's

Pretty Good Privacy (PGP) private key file and transmit it to the creator of the virus via FTP. The Marker virus, discovered about half a year later, used a similar technique to obtain information about the infected user from the system's Registry and transferred the data to the author's FTP site. This capability allowed Marker to maintain a trail of infected users, empowering its creator to study relationships between members of the targeted organization.

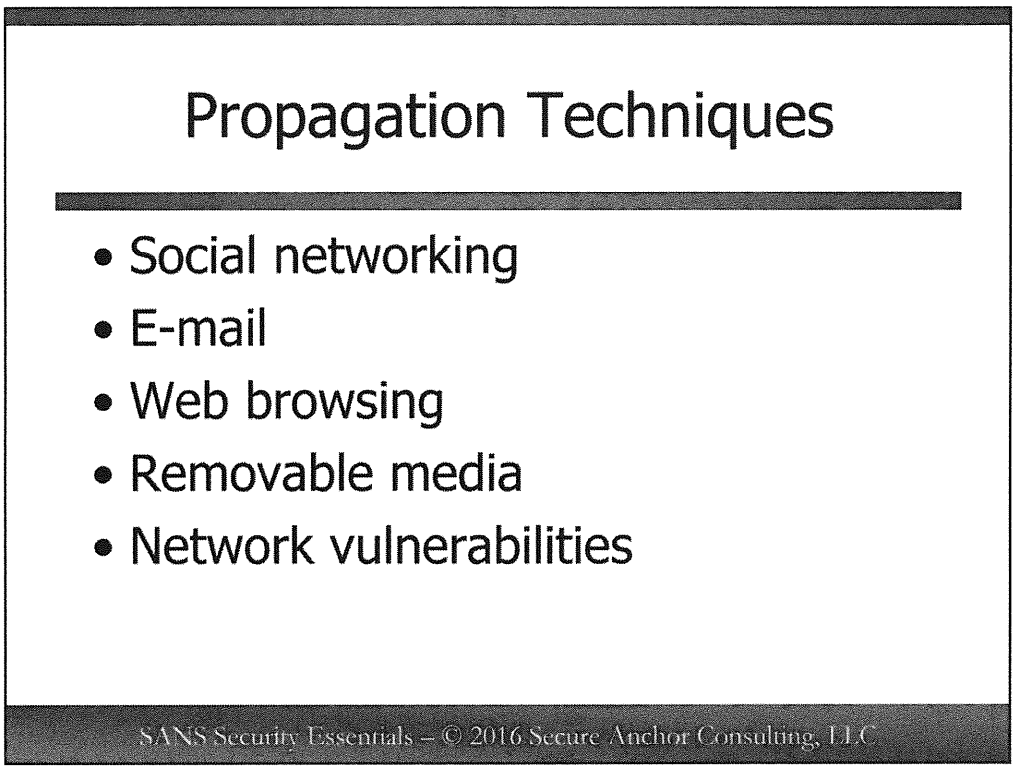
Providing Backdoor Access

Attackers use backdoors to ensure that they retain access to the system after it was compromised. An elaborate example of using a backdoor can be found in the functionality built into the Leaves worm. The Leaves worm spread by scanning for hosts that were already infected with the SubSeven Trojan. When such a system was located, the worm attempted to authenticate to the Trojan using a master password that was known to work with some versions of SubSeven. After Leaves gained access to the computer through this backdoor, it removed the pre-existing Trojan, presumably to prevent anyone else from getting into the system through such means.

As the next step, Leaves acted to provide its author with a backdoor of his own, by connecting to a channel on a remote Internet Relay Chat (IRC) server. As the worm spread, infected computers logged into the IRC channel and awaited additional instructions from the worm's creator. This gave the attacker the ability to authenticate to all instances of the worm simultaneously and issue commands for launching programs, manipulating files, and obtaining system information.

Summary

Malware writers have numerous options for creating malicious code that meets their objectives by incorporating different malware types and attack capabilities. The trick is to prevent your systems from being infected before a worm hits.



Propagation Techniques

Introduction
Worms, viruses, and Trojans that find their way onto our systems and networks are becoming increasingly complex. Authors of malicious code have numerous choices when it comes to designing mechanisms for malware propagation, infection, and cloaking.

Propagation Techniques
Although we might not be able to account for all possible ways in which malicious code penetrates defenses, some propagation techniques are employed much more often than others. The following list presents distribution mechanisms used most frequently:

- Social networking
- E-mail
- Web browsing
- Removable media
- Network vulnerabilities

Removable Media
One of the fundamental properties of mobile malware is the ability to replicate. Within the confines of a single computer, replication involves modifying the system's environment. However, the possibility for additional infection within the same system can be quickly exhausted. Malware specimens benefit from the ability to spread to other computers. Removable media, USB drives, and CDs have provided a time-tested method for malicious code to travel between machines.

Now that floppy disks are no longer used, high-capacity removable media, such as USB memory drives, CDs, and DVDs, continue to be used to distribute files, and therefore to spread malware. If you save infected files onto a removable storage device and later insert it into another computer, you might inadvertently assist in spreading the virus. Official media distributors are as capable of spreading malware as users of Zip disks and home-burned CDs are.

Hint: If you are creating custom floppies, CDs, or DVDs for distribution to others, such as customers or internal users, be sure to first scan the contents of the master disk with an up-to-date antivirus program. Additionally, take advantage of free tools such as md5sum to record cryptographic signatures of all files in the original package. Use the signatures to spot-check duplicated disks to make sure their contents are identical to the original.

When designing network defenses, do not forget to account for the possibility that malicious code might be brought into your environment by foot through the use of removable media. Furthermore, you should not assume that a vendor-installed computer system is malware-free.

E-Mail

It's no surprise that e-mail offers a convenient alternative to using removable media for distributing files. The same malware that attaches to legitimate files and spreads via floppies can be disseminated through files that are sent as e-mail attachments. Because e-mail messages are usually allowed to leave and enter the organization's network, malware often uses e-mail as a propagation mechanism. Malware might use the infected host's e-mail client and server to spread, or contain, its own SMTP engine. The latter method is becoming more popular. If you suddenly get a lot of port 25 (SMTP) blocks on your firewall coming from end-user workstations, it might indicate the presence of an e-mail malware threat.

Web Browsing

Much like e-mail, web browsing is an activity that takes place at most, if not all, organizations connected to the Internet. A web browser provides malware with another way of entering the system. We already mentioned malicious code based on Java applets, JavaScript, ActiveX, and other types of malicious HTML content that could be embedded into a Web page. This approach allows malware to propagate via the web when a potential victim connects to the rogue web site. In another example, the backdoor built into OpenSSH, was disseminated when people downloaded a tainted version of the program from ftp.openssh.org and its mirror sites.

Network Vulnerabilities

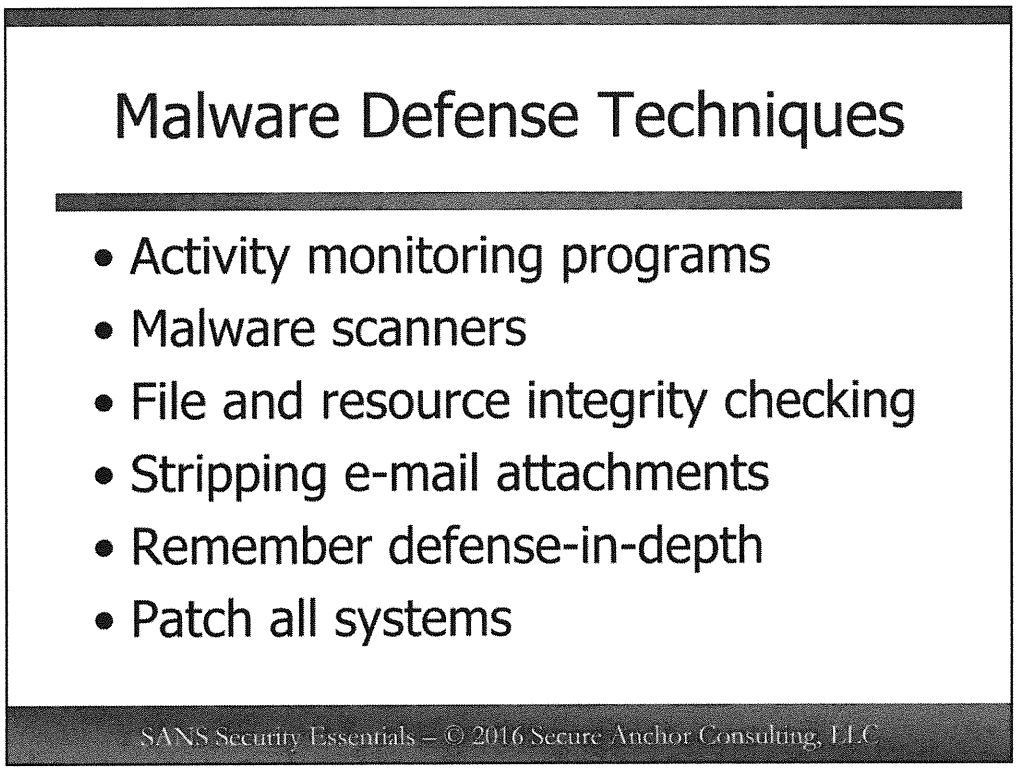
Yet another way for malware to propagate involves actively probing systems for holes that can be exploited over the network.

A malware specimen does not have to limit itself to a single operating system when propagating through the use of network vulnerabilities. For example, the sadmind/IIS worm spread, by targeting unpatched Solaris and Windows systems. The worm took advantage of a two-year-old buffer overflow vulnerability in the sadmind program to compromise Solaris systems. After infecting a Solaris machine, it used a seven-month-old vulnerability to compromise Windows IIS servers.

Taking advantage of overly permissive trust relationships is another way in which malware can exploit network vulnerabilities. This approach dates as far back as 1988, when the Morris worm, also known as the "Internet Worm," disrupted most of the major U.S. research centers in a matter of hours. One of the worm's propagation techniques involved examining lists of host names of which an infected system was aware and attempting to connect to them in hopes that the infected machine is trusted to execute commands on the remote machines. The Morris worm also attempted to exploit known vulnerabilities in several commonly used network services.

Summary

Any technology that is in wide-spread use for legitimate purposes will also be exploited by malicious code. Anything that can be used for good can be used for evil. Therefore, we have to understand the popular channels of distributing information and make sure it is legitimate and not attack traffic.



Malware Defense Techniques

You should now be familiar with the capabilities of several types of malicious code, their propagation and infection mechanisms, and techniques that they can employ in an attempt to thwart detection. If you are feeling powerless against the forces of malware, don't worry; there are practical steps you can take to improve the resilience of your defenses. Historically, methods of preventing infections were heavily focused on the use of antivirus software on user workstations. Such products are still critical components of a defense infrastructure, but they should not be employed without additional security layers. As we discuss in this section, an effective malware defense strategy should incorporate the following items:

- Antivirus software at multiple locations
- Up-to-date virus signature files
- A practice of reviewing and installing security patches
- Lock-down of system configuration and dangerous application features
- Blocking file attachments (the number one technique to stop e-mail viruses)

Capabilities of Antivirus Software

Antivirus software is one of the most popular measures for fighting malware infections, particularly on Windows systems. When deployed in conjunction with other security mechanisms, it can be reasonably effective in blocking and detecting common malware specimens. There are three primary types of defensive techniques that may be incorporated into an antivirus product:

- Scanners
- Activity monitors
- Integrity checkers

Scanners

Scanners are the best-known form of antivirus defenses. They operate by searching files and disk boot areas for content attributable to known malware specimens (signatures). If a scanner detects a file that matches a malware signature, it may be able to block access to the file. One of the biggest challenges of this approach is that it detects only malware for which the antivirus vendor has already provided a signature. Another problem with scanners is that the signature might not be able to match all variants of a particular malware specimen, especially if it has polymorphic properties. To counteract these problems, some antivirus products are able to search files looking for malicious-looking code routines, a process called "heuristics." Heuristic search routines allow antivirus software to identify malicious programs without having a predefined signature. Unfortunately, *heuristic* scanners are overly susceptible to false-positives and significantly slow down scanning.

Activity Monitors

Activity monitors, also known as *behavior blockers*, aim to prevent infection by monitoring for malicious activity and blocking such activity when possible. Activity monitors observe programs that run on the system and look for attempts to perform actions, such as low-level formatting of the hard drive, writing to boot records, and modifying program files. Activity monitors may be able to detect malware specimens that they have not encountered before; however, they are not considered a particularly strong form of defense if deployed alone. Some of the ways of bypassing their restrictions involve directly accessing interrupt handlers on hardware controllers, performing actions that are not being monitored, or disabling the monitoring process altogether.

Integrity Checkers

Integrity checkers compute checksums or hash values of original files on the system and store the results in a baseline database. During subsequent runs, the defense program compares the current state of the filesystem with the baseline, warning the administrator if any of the monitored files have changed. Examples of integrity verification software include Tripwire and AIDE. These programs might be considered generic detectors because they have the ability to detect activity from previously unknown malware specimens. However, integrity checkers are rarely able to block malicious code from executing—they are usually implemented as infection detectors, not preventers.

No single malware defense technique is effective on its own. It is a good idea to use antivirus software that employs several approaches discussed previously. Luckily, modern antivirus software usually combines activity monitors and scanners into a single product. For more robust protection, consider also deploying an integrity checker on your critical systems, even if they are already running antivirus software.

As we discussed in the "Propagation Techniques" section, malware can spread using a variety of methods, such as CD-ROMs, e-mail, web browsing, and network exploits. It is prudent to install antivirus software at multiple places in your infrastructure so that it watches as many entry points as possible. Classic locations for antivirus products include:

- Workstations
- File and print servers
- Mail servers
- Internet gateways

Leading antivirus products provide software that is able to operate at these locations. You might be tempted to install antivirus products from different vendors on the same system to improve the likelihood that a malware infection will be blocked. This is usually a bad idea, because it will rarely result in a stable configuration. Instead, consider using one vendor's product in one location (perhaps for server and workstation scans), and another vendor's product in another location (such as the mail server). Keep in mind that setting up multiple products will increase the solution's cost and will complicate its maintenance.

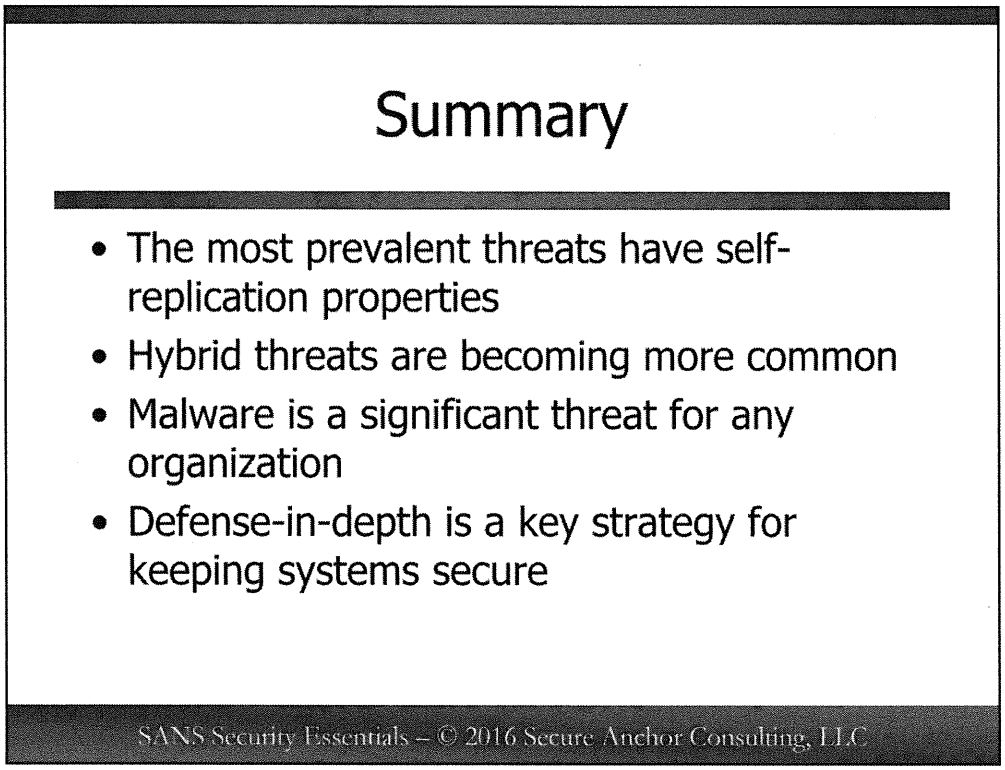
Stripping E-Mail Attachments

The majority of today's threats use e-mail attachments to spread to new targets. Organizations relying solely on antivirus software at e-mail gateways were scrambling to install the latest patches when new threats were discovered, and then recover the systems that became infected. Organizations that stripped e-mail attachments before delivering messages to the recipients waited for their antivirus updates to be delivered in planned update schedules, knowing that their e-mail systems were not vulnerable to the new threats. Which option sounds like a better defense-in-depth technique?

Consider the number of e-mail attachments malware can utilize to propagate to new victims. Some file extensions are obviously executable files: .exe, .com, .bat, for instance. Other attachment types are less obvious, but still represent potentially executable content including: .pif, .jsp, .vbs, .shs and .scr. Microsoft Office files represent a risk of embedded macro code that runs when the document is opened (.doc, .xls, .mdb, and so on), as well as Internet Explorer files that run with a low security restriction when opened from the local filesystem (.htm, .html, .mht, and so on).

There is little challenge in stripping e-mail attachments from a technical perspective—most antivirus gateway software has the ability to identify the type of attachment and safely quarantine the contents before delivering the remainder of the message to the recipient. The challenge for most organizations is political—what is an acceptable trade-off between security and functionality? Although this is a difficult question to answer, organizations need to consider the total cost of permitting e-mail attachments into their organization. If you work for an organization that does not strip e-mail attachments, or allows e-mail attachments that are dangerous, it is important to collect the appropriate information to motivate management to change policies regarding e-mail attachments. This includes the number of man hours spent dealing with virus outbreaks, the amount of time spent managing antivirus updates, costs from unplanned outages while antivirus signature data files are updated to mitigate a new threat, the cost of deleted or leaked information from an infection, and so on.

Alternative technologies are available for use in sharing files among systems instead of using e-mail attachments. Where establishing an FTP site might introduce different risks (clear-text passwords), protocols such as the Secure Shell (SSH) include tools to transfer files securely, encrypting the authentication and contents of files exchanged among systems. Alternatively, an HTML scripting language, such as PHP, includes extensions to accept file uploads from a web browser; organizations could develop an application to accept file uploads from unauthenticated users through a web browser and quarantine the file for analysis by antivirus tools before being made available to the recipient.



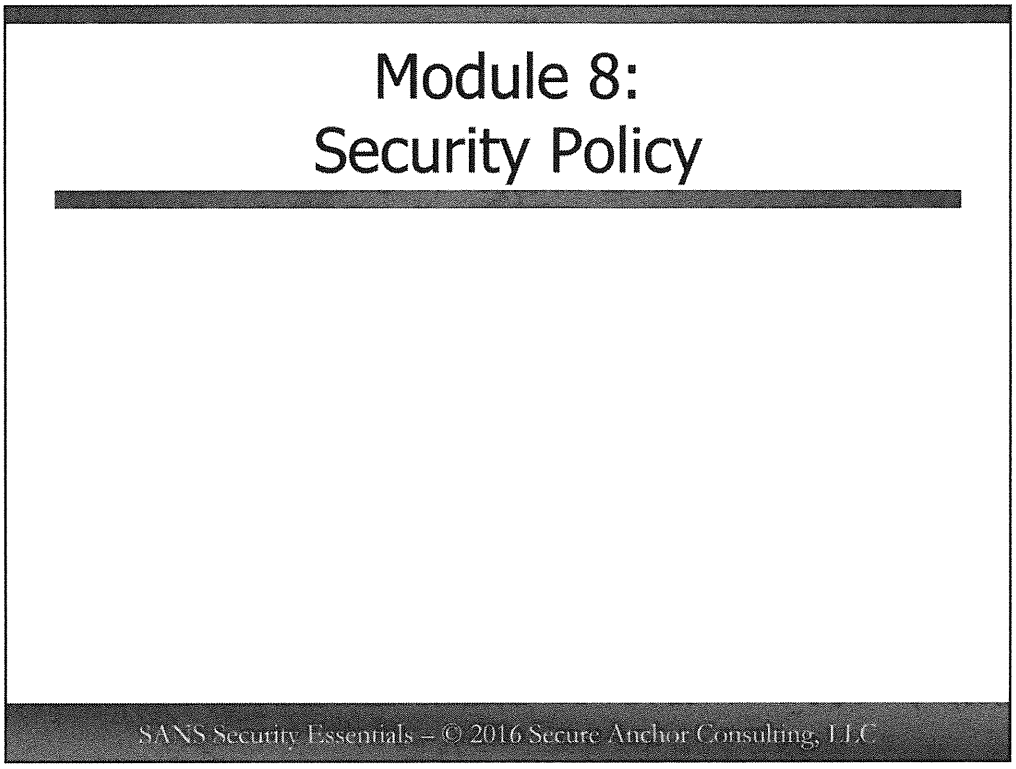
Summary

We've examined several different types of malware in this section. By their nature, self-replicating worms are the most prevalent threat, because they can reach the largest number of hosts in the shortest amount of time. Other malware types include viruses, Trojans, malicious browser content, and hybrid threats. More advanced malware applications are employing hybrid techniques to reach as many systems as possible, using worm, virus, and Trojan properties to quickly spread to numerous systems.

Malicious mobile code is the term used to describe malware that is meant to replicate from computer to computer, most of the time as a function of their own coding. Malware uses any mechanism available to propagate to additional hosts that will allow it to spread. The propagation methods range from traveling on floppy disks to advanced peer-to-peer protocol distribution.

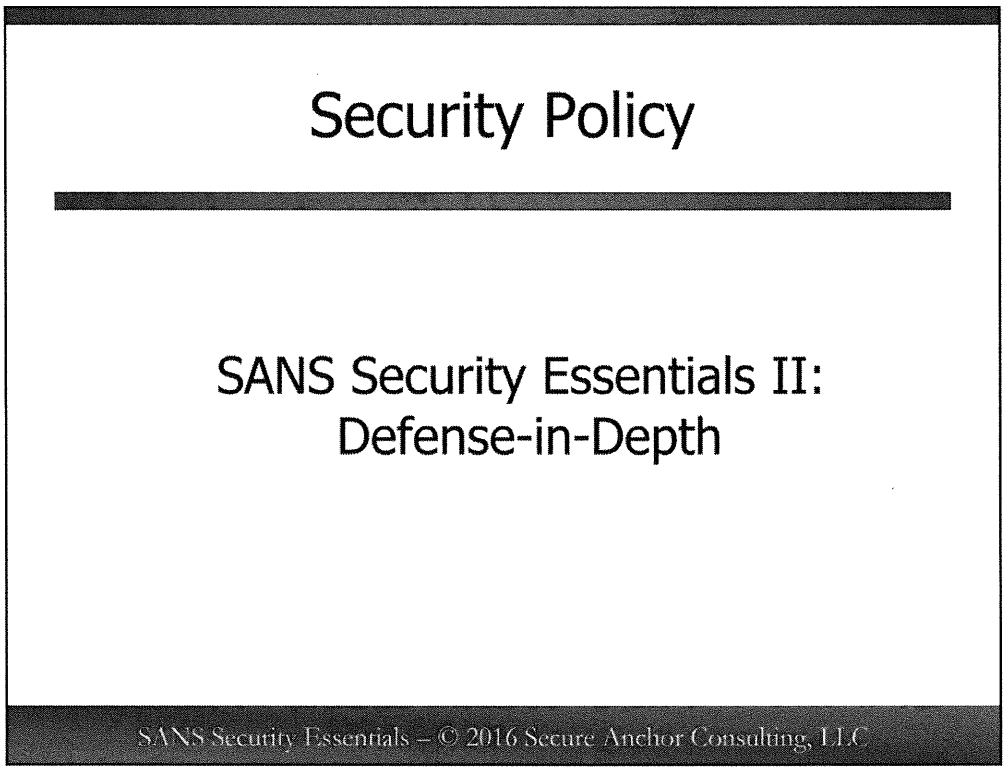
The presence of malware signals the likelihood of a significant number of compromised systems. Comprehensive patch management and antivirus tools are a necessary defense mechanism to protect systems from becoming compromised.

We also learned in the beginning of the section the different strategies for implementing defense-in-depth and that it is one of the best ways to keep your organization secure.



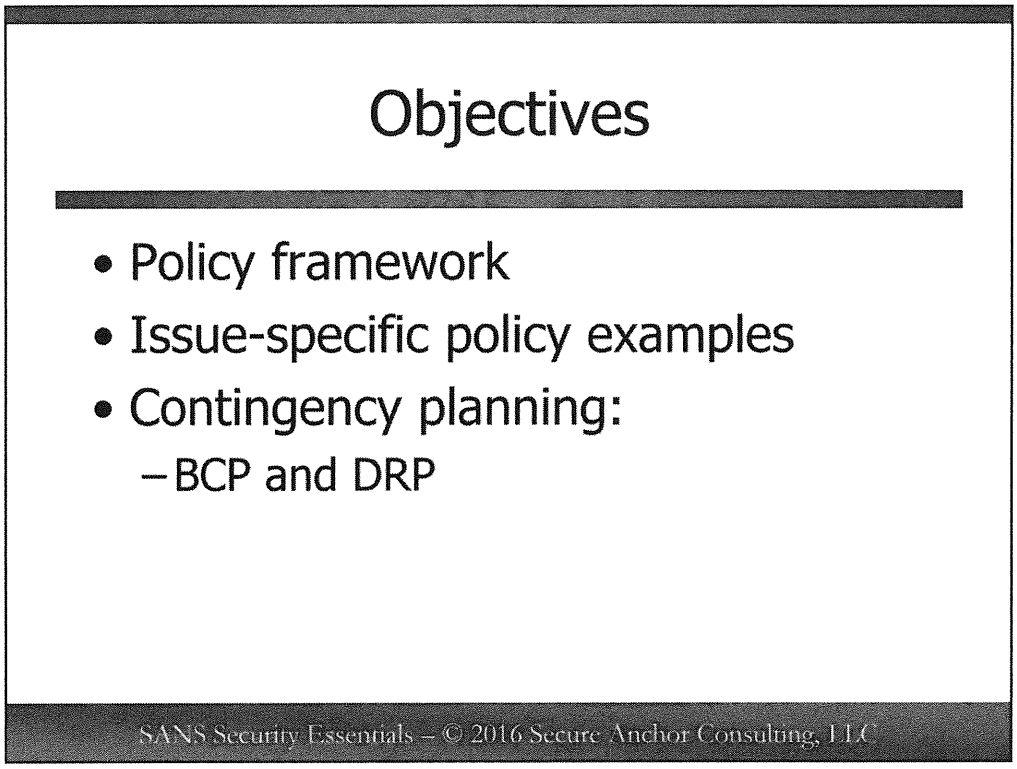
Module 8: Security Policy

This section intentionally left blank.



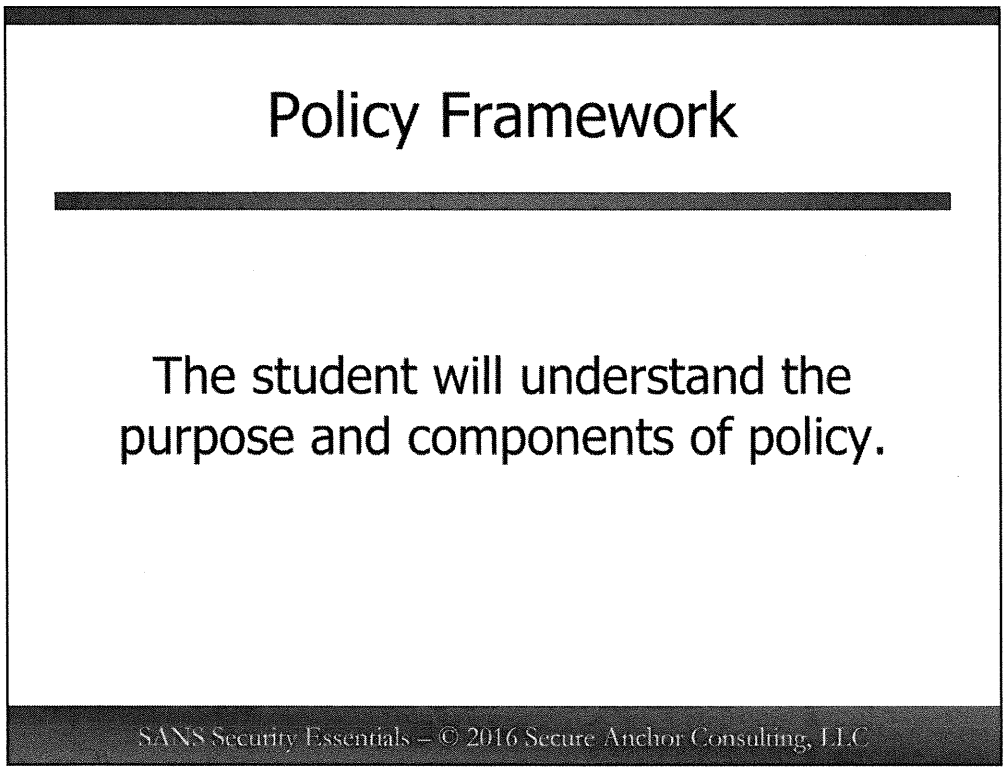
Introduction: Security Policy

In this module, we equip you to write and evaluate security policies, whether you focus on an entire organization or on your individual job. We also cover contingency planning, such as for when your organization faces a disaster.



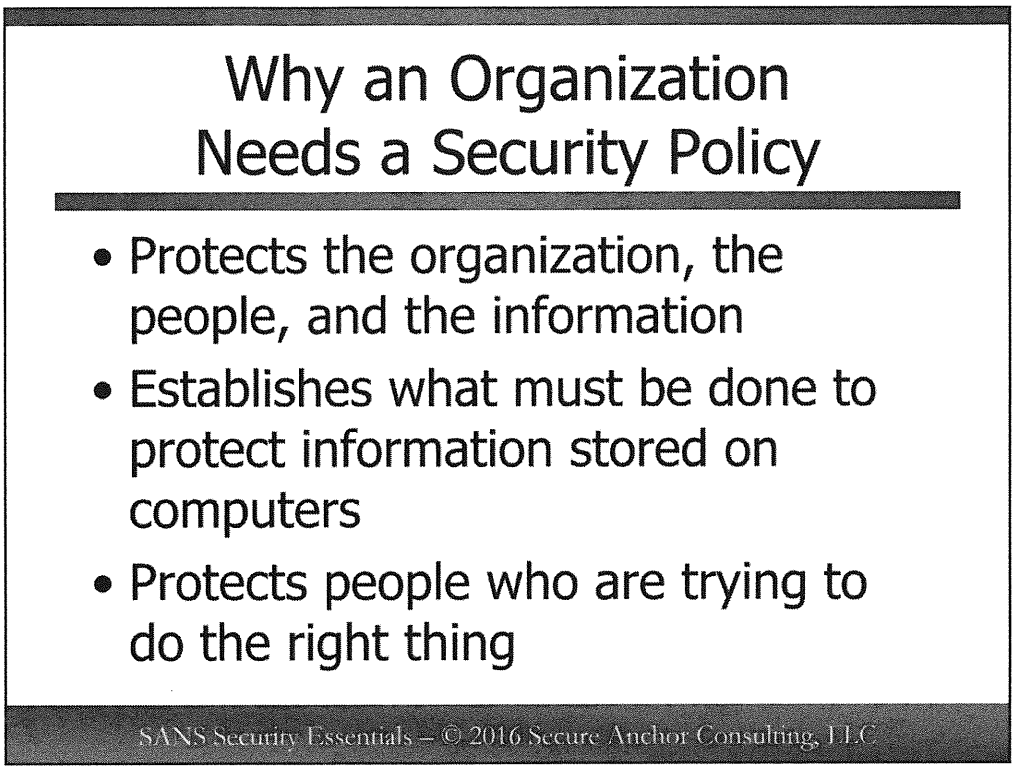
Objectives

In this module, you learn how to assess a policy by establishing a baseline framework to work within, and by establishing a mission statement that defines your policies. You learn how to assess and repair critical policies, and then how to repair others, one at a time. Involved in policy assessment is the need to understand how policies work throughout the organization and the need to understand unwritten or missing policies. In assessing policies, you should understand that there are roadblocks. Delays should be expected and overcome.



Policy Framework

This section intentionally left blank.



Why an Organization Needs a Security Policy

Consider This Scenario
NSWC had a policy and procedure for vulnerability scanning that was approved by the Captain of the base. As long as the person doing the scanning followed the policy and procedures, he was protected. The communications on the model were the same as the one on the real ship. When its networking hardware received a packet on a certain port, it died. Its Fiber Distributed Data Interface (FDDI) ring came to a complete stop.

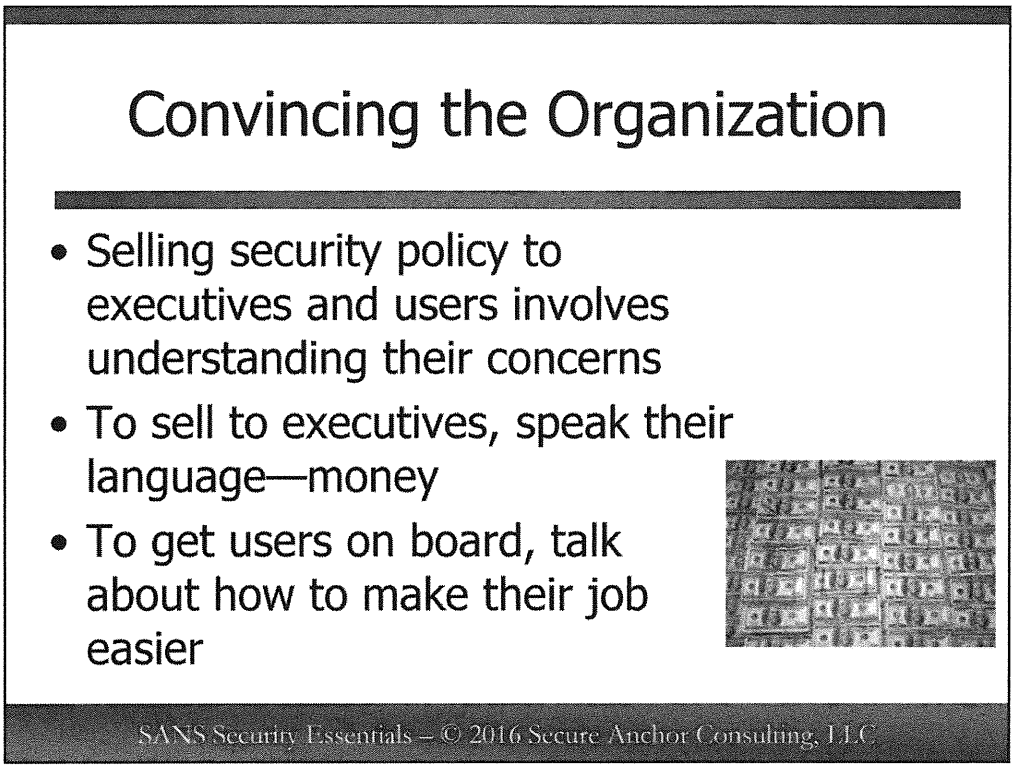
The people in this lab were furious when the person performing the scan did this. They formed an investigative panel and called him in. He could see grim looks all around the table, and the sparks flew. Finally, someone asked whether the incident could happen in real life. The answer is, "Yes." The next question then is, "Then shouldn't we get it fixed?"

The policy point is that the network scan dropped the network during a test, and a lot of money was lost. This made people angry enough that the scanner's job could have been in jeopardy if he had not been protected by policy. Remember the key rules of vulnerability assessment are to make sure you always get permission in writing and to put out the word ahead of time so everyone knows what you are doing. These policies form the groundwork for a pretty airtight policy. The procedures had stated that the security office was supposed to put out a notice in advance of the scan, use an approved tool with specific settings, and be available in the office during the scan. All of these procedures had been followed.

Safeguarding Information and People
Safeguarding information is a challenge when records are created and stored on computers. We live in a world where computers are globally linked and accessible, making digitized information especially vulnerable to theft, manipulation, and destruction. Security breaches are inevitable. Crucial decisions and defensive action must be prompt and precise.

A security policy establishes what you must do to protect information stored on computers. A well-written policy contains a sufficient definition of "what" to do so you can identify, measure, or evaluate the "how."

An effective security policy also protects people. Anyone who makes decisions or takes action in a situation where information is at risk incurs personal risk as well. A security policy allows people to take necessary actions without fear of reprisal. A security policy compels the safeguarding of information while it eliminates, or at least reduces, personal liability for employees.

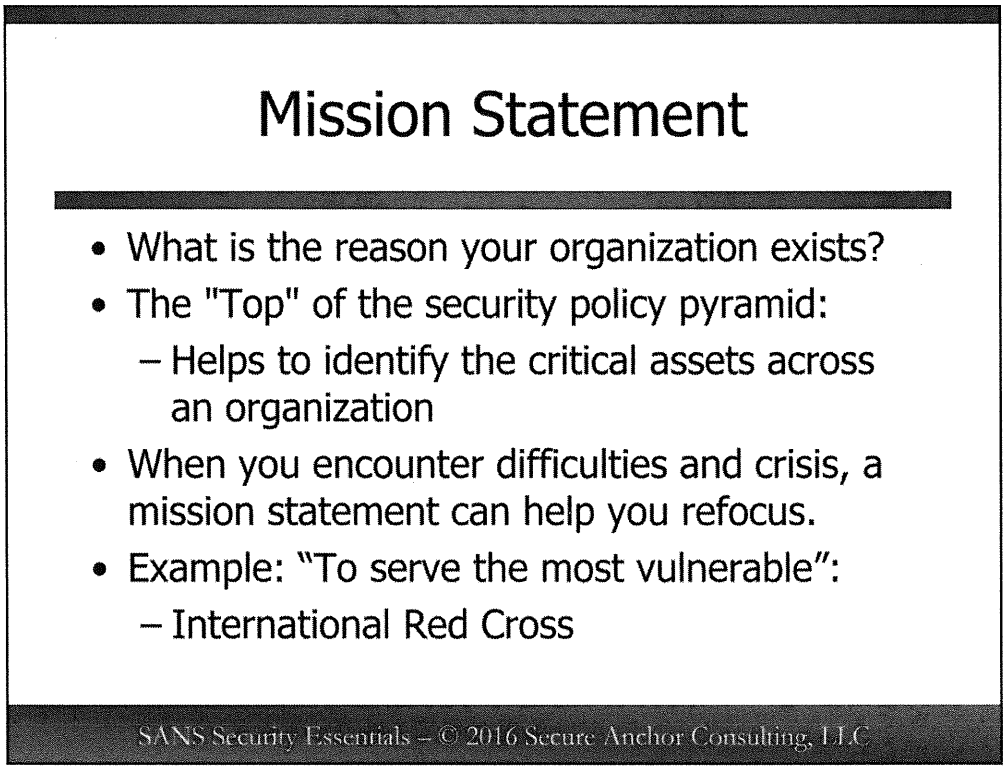


Convincing the Organization

When it comes to selling the organization on the importance of a security policy, the more you understand the organization's concerns, the better you are able to sell your ideas. Knowing your company's overall security posture is vital. Understanding any specific areas of concern is also important. Perhaps intellectual property or web applications have recently become an area of interest. Try to understand what the organization's primary concerns are and then look for ways to address those concerns.

On a more detailed level, people in different groups within an organization have different levels of concerns. Generally speaking, senior executives are concerned about finances. Understand this basic truth and you will start packaging your ideas in a way that provides senior management with information that speaks their language-saving money.

Users generally are interested in things that make their jobs easier. When selling security policy to user groups, try to identify with how the policy can help them perform their jobs in less time or with less effort.



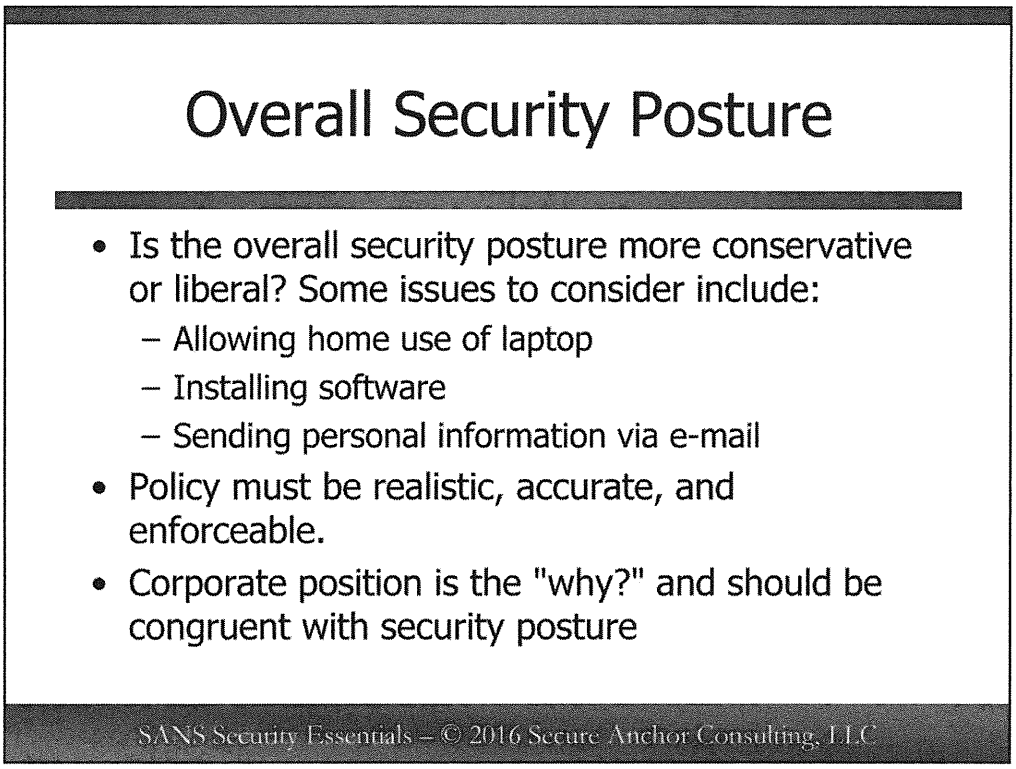
Mission Statement

A mission statement is the idea behind a brand. It is a statement to your customers and suppliers of what they can expect from you. Peter Drucker defines it in this way:

"A mission statement has to be operational, otherwise it's just good intentions."

A mission statement embodies your organization's reason for being, your purpose. If your organization does not have a mission statement, you should attempt to develop one and get it approved. You need an approved mission statement as we move forward to evaluate policy.

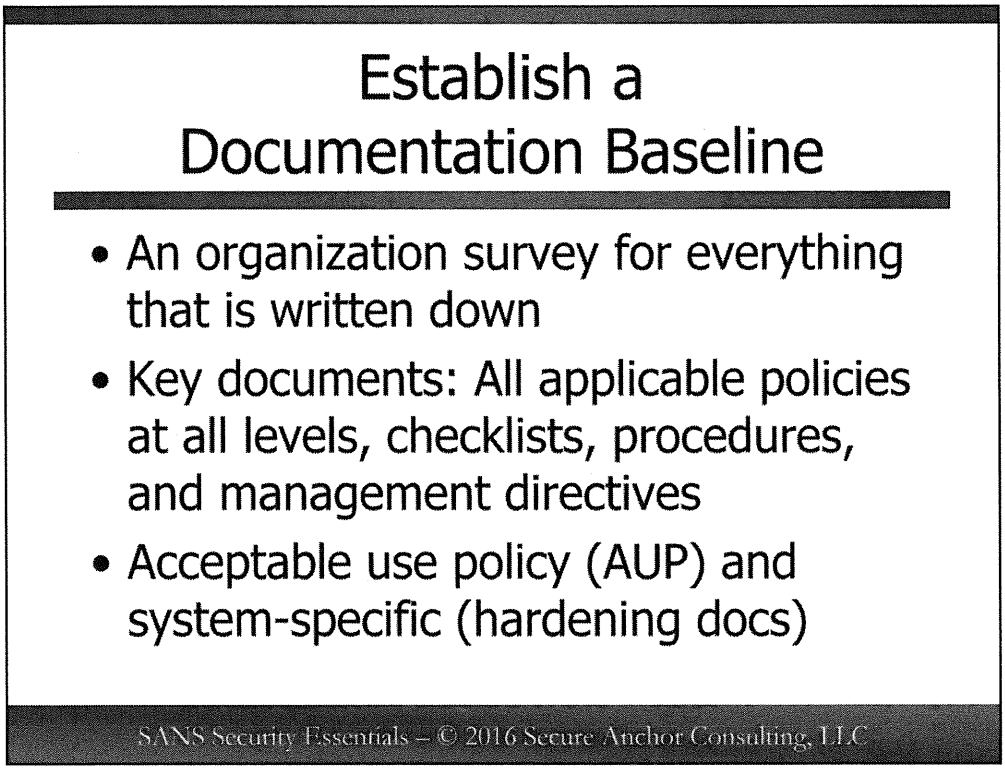
What does a mission statement have to do with information security? One of the biggest criticisms of security workers is that they are not sensitive to the needs of the business. So, we start with the heart and the soul of an organization, its mission. There are three major styles of good mission statements and one major style of bad mission statements.



Overall Security Posture

Many times, a mission statement points to what the expected overall security posture of an organization will probably look like. We discuss a number of "hot button" points, and you can probably add some that you've observed in your career to the list. One of the things to be sensitive to are "hot buttons" —the organization reacts differently to these than their normal approach to business. For instance, if a company was normally relatively liberal and informal, but required an incredible amount of documentation for sales, that might be worth discussing with management. We once worked with a high-tech company that ran seminars, and for every item it sold, it had to staple the cash money, check, or credit card receipt to the receipt for the purchase. The accountants spent hours unstapling all the receipts, depositing the money, and managing the records. After discussing this with management and explaining that it ran completely contrary to the corporate culture, they changed this policy.

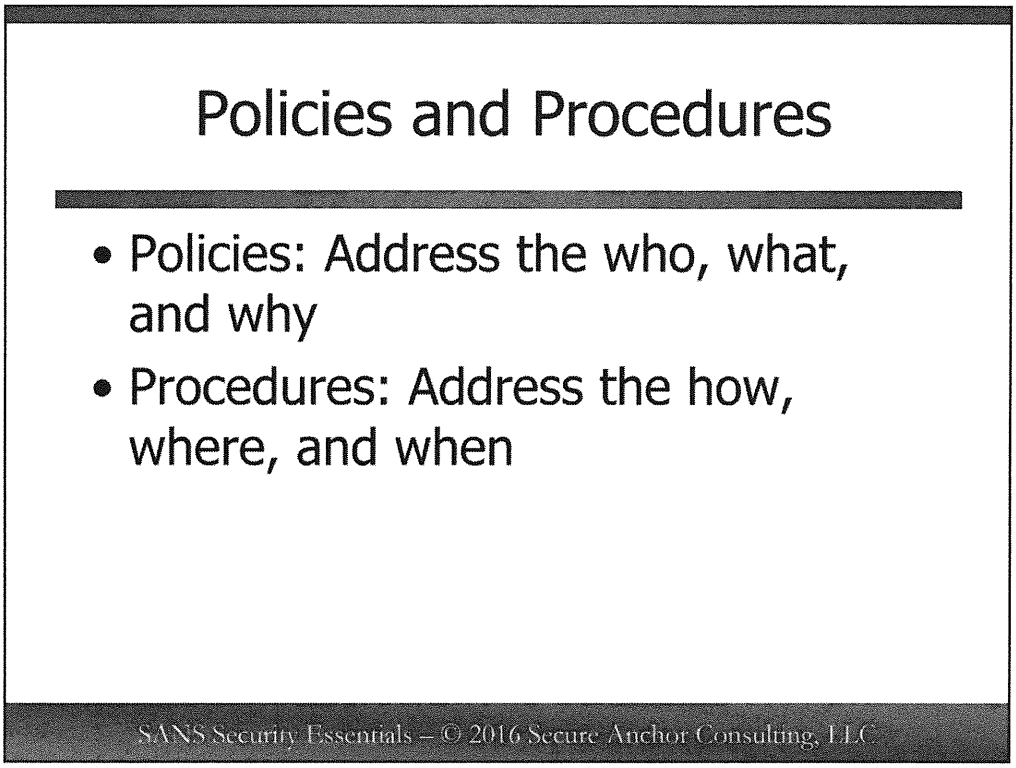
There is no right or wrong security posture. A conservative approach might offer security benefits to the company by placing more controls on what an employee can do. However, this approach might take more security and IT staff to implement and monitor. A liberal approach might provide a more informal work environment, but might put the company assets at a higher level of risk.



Establish a Documentation Baseline

A baseline is our foundation for evaluating policy and it is made up of several components. We have the mission statement that defines what customers, suppliers, and employees should be able to expect from the organization. We have the assessment of the organization's security posture, which is a bit like looking in a mirror. Our mission statement is the way we hope people view us; the security posture is what we actually look like. Now, we can begin to evaluate policy. As you know, policy exists on several levels. Hopefully, everything is organized and up to date; if not, you need to begin the search for written guidance.

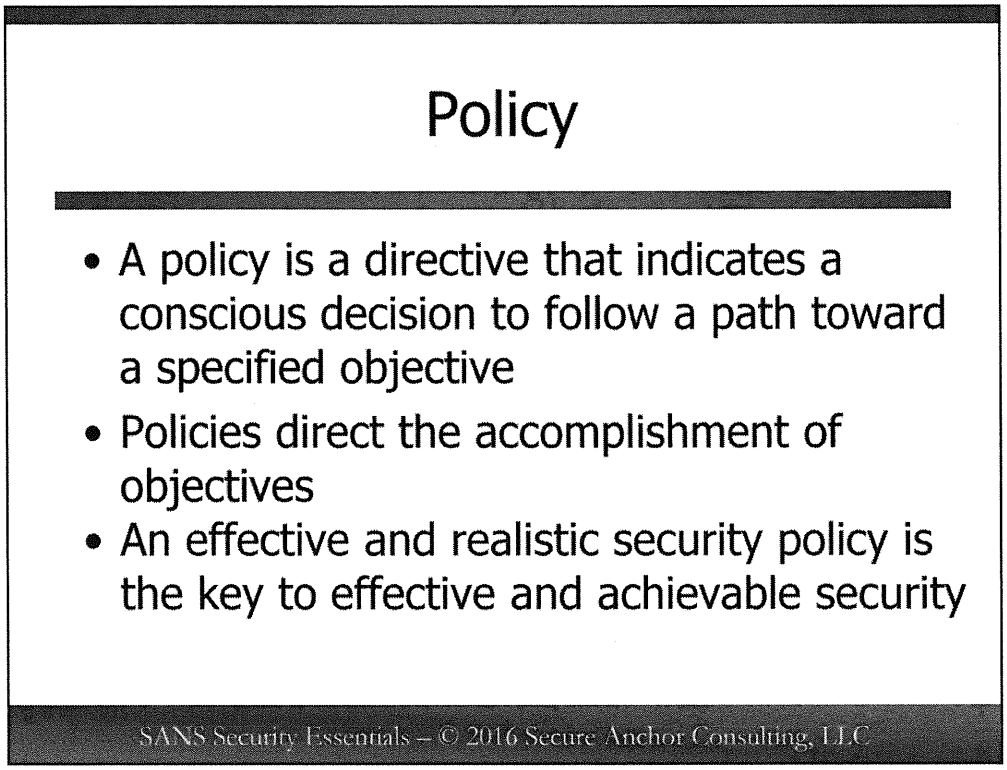
Enterprise-wide or corporate policy is the highest level of policy and consists of a high-level document that provides a direction or thrust to be implemented at lower levels in the enterprise. The ISO 17799 approach to this for information security is a letter of endorsement from senior management. This policy must exist to properly assess lower-level policy. If this policy does not exist, begin work to create this policy document and get it approved before attempting to assess lower-level policy. This enterprise- or corporate-level security policy is the demonstration of management's intent and commitment for the information security in the organization. This should be based on facts about the criticality of information for business as identified during our assessment and evaluation of security posture. The security policy statement should strongly reflect the management's belief that if information is not secure, the business will suffer.



Policies and Procedures

What do you do when some work does not seem to be covered by an organizational policy? Procedures are derived from policies; if you can characterize the procedures you follow (and you should be able to do that easily), then you can derive the parent policy. This is true even if it has not yet been written and signed. By walking through the who, what, when, where, and why, the parent policy is derived from an understanding of the procedure.

In your organization, what procedures can you list for which you need to document the policy? Make notes on the who, what, when, where, and why of your procedures. You will be able to derive any missing policies based on these notes.

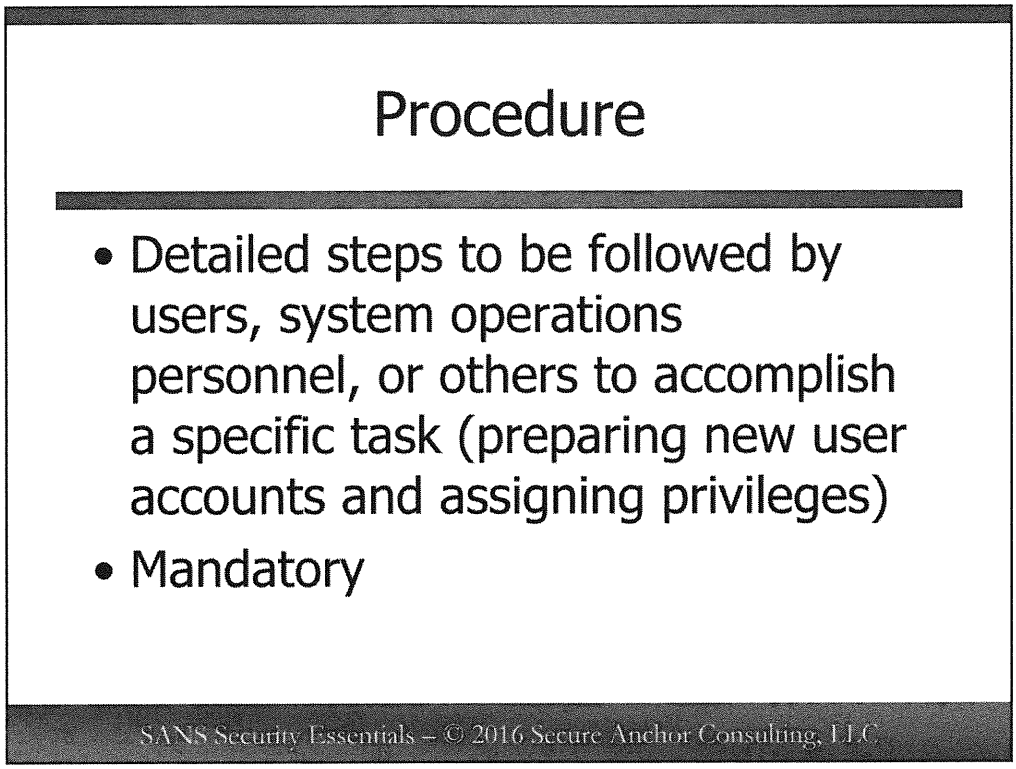


Policy

Defining a Policy

A *policy* is a *directive* that indicates a conscious decision to follow a path toward a specified objective. Often a policy can initiate the institution and empowerment of resources, or direct action by providing procedures or actions to be carried out. The policy itself should be effective and realistic and have achievable security goals.

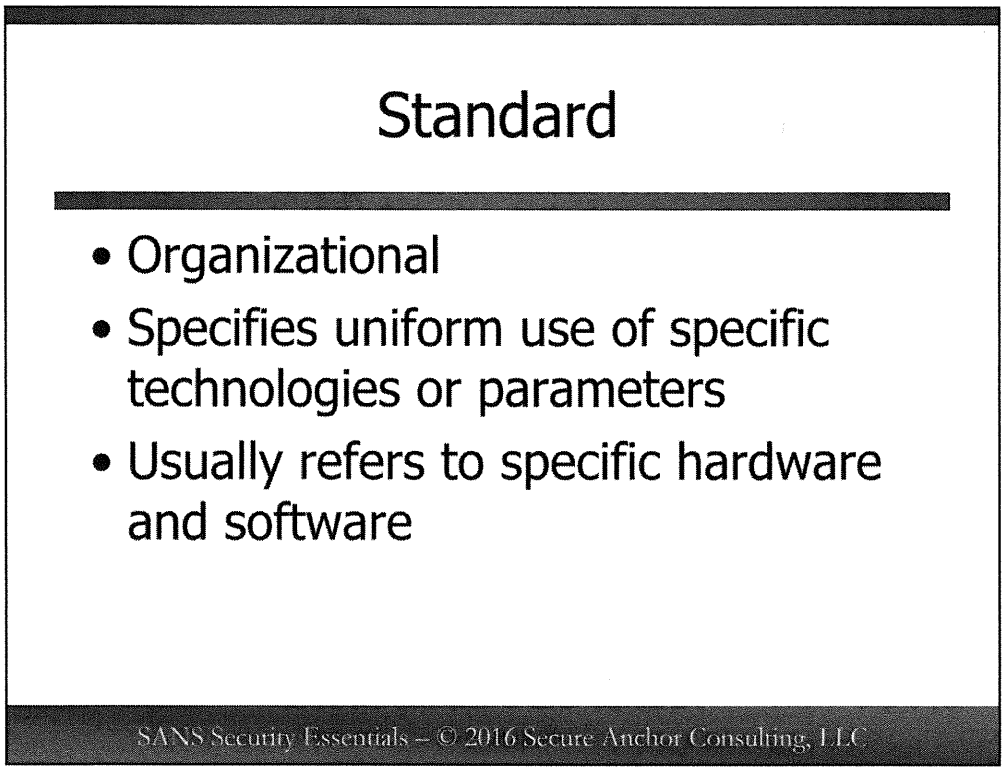
It is critical to write down in a clear and concise manner what is expected of everyone in the organization when it comes to security. It is also helpful to inform people about what is expected of them, what the organization does, and what others in various roles within the organization do.



Procedure

Definitions and Issues

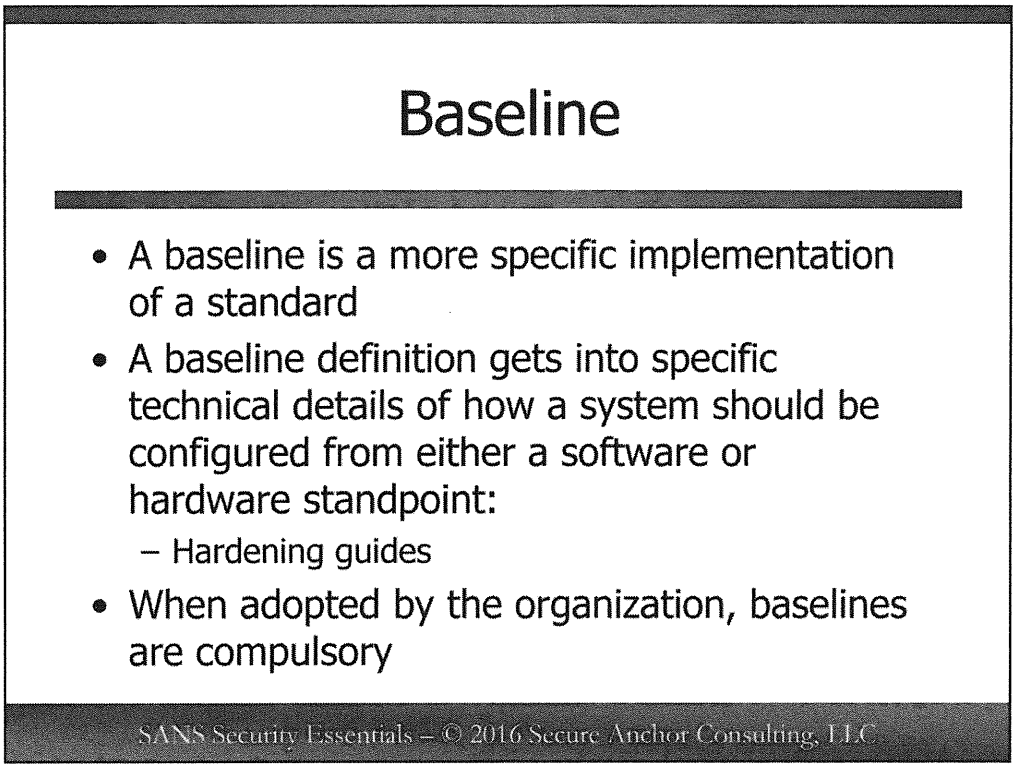
A procedure is a step-by-step document that is used for operations. Procedures can be daily operations, such as nightly backups or infrequent operations, such as recovering from a disaster. These steps must be clear, complete, and concise. They should also be reviewed on a regular basis to ensure they are accurate and that no changes have occurred.



Standard

Definitions and Issues

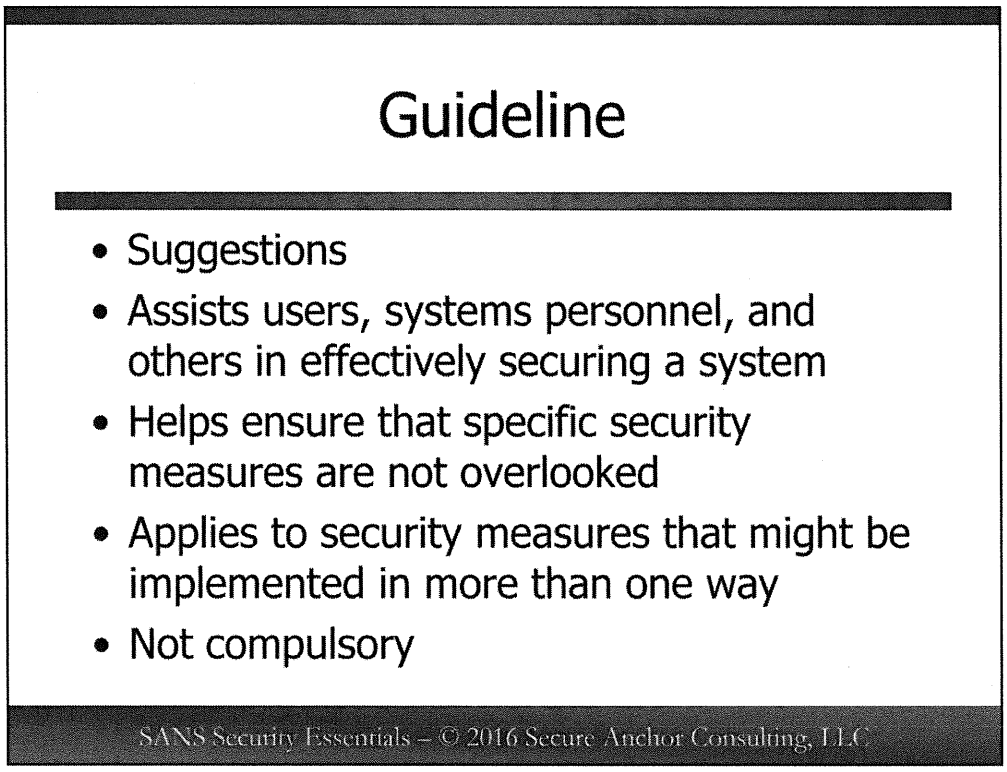
Standards are applied to the organization as a whole. As with policies, these are mandatory. Standards are more specific than the overarching policies. They provide additional definition to the policies and tailor them to specific technologies. Unlike a policy, a standard does not state what is expected of a user from an organizational security stance. Instead, a standard specifies a certain way something should be done or a certain brand or type of equipment that must be used. A simple example of a standard is that all computers purchased must be a certain model and from a certain vendor.



Baseline

Definitions and Issues

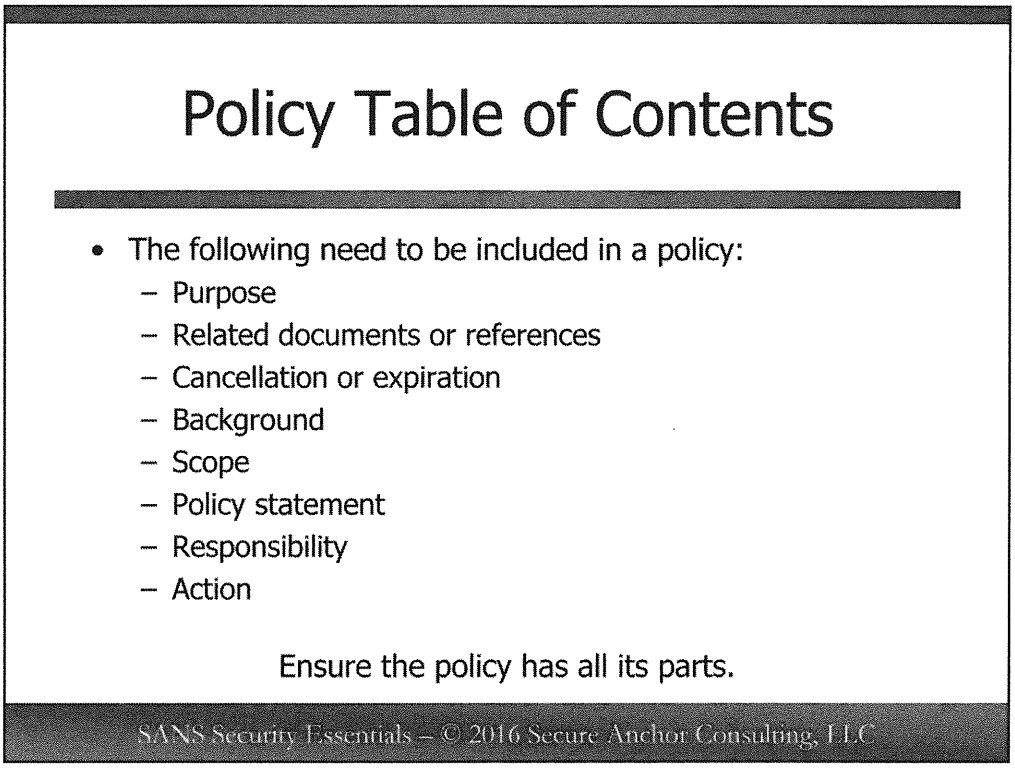
A baseline definition is essentially a more specific implementation of a standard. A baseline definition usually gets into specific technical details of how a system should be configured from either a software or hardware standpoint. After these documents have been thoroughly tested, they become mandatory for someone to enforce. Usually a baseline starts off as a guideline until it has been properly modified to meet the needs of the organization. Hardening rules for setting up a new server is an example of something that starts off as a guideline and quickly turns into a baseline.



Guideline

Definitions and Issues

Guidelines, unlike standards and policies, are not mandatory. Best practices are examples of guidelines that many organizations try to achieve; however, there is not a penalty if guidelines are not met. A guideline is more like a recommendation of the way that something should be done; however, people can choose whether they want to follow it or not. A best practice might start off as a guideline, and if analysis shows that there is a great benefit to following this guideline from either a security or efficiency standpoint, the guideline might become a standard, which would then make the guideline mandatory to follow.

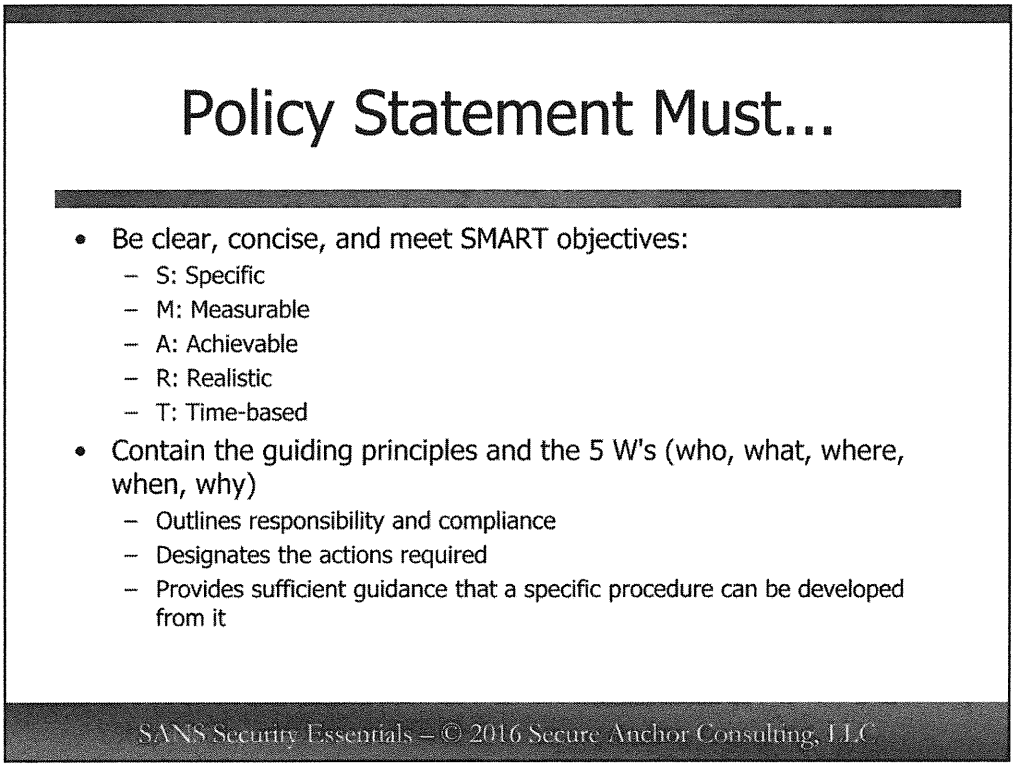


Policy Table of Contents

It is good to follow an approach where if the policy you need does not exist, or is badly out of date, you create, or update it, and then have it approved. The final step in establishing a framework is to assess critical policies. Even though you just wrote or updated them and you know they are correct, policies should be checked. It is better if you do not check your own work. We can spot only our own errors a small percentage of the time. Any detail-oriented person can check to see that the format of the policies is correct. In fact, a person that is not familiar with the subject the policy covers is ideal. People who are subject matter experts might not notice omissions in procedures because they have the procedure memorized. When you ask a person to help you by verifying policy contents, ask that she include the most common elements in her analysis. These include:

- **Purpose:** The security policy usually contains a statement, often at the beginning, describing the reason the policy is being established and any associated goals.
- **Related documents:** This is often entitled "References" and usually cites higher-level policy or implementation guidance.
- **Cancellation:** A new or updated policy might supersede existing (perhaps outdated) policy. This section identifies those policies and clarifies what is actually in effect.
- **Background:** This optional section provides information amplifying the need for the policy. It might also provide historical information relevant to the subject.
- **Scope:** This section identifies the depth and breadth of coverage (to whom or what the policy applies). Is it for one element of the organization, or will it also apply to contractor agencies who work for your organization?
- **Policy statement:** Identifies the actual guiding principles or what is to be done. The statement(s) are designed to influence and determine decisions and actions within the scope of coverage. The statements should define actions that are prudent, expedient, or advantageous to the organization.

- **Responsibility:** The security policy document states who is responsible for what. Typical positions that might be addressed include the head of the corporation, the CIO, people in the legal department or in human resources, system administrators, and information security officers. Subsections might identify how additional detailed guidance will be developed and provided, as well as the frequency of policy review. Methods or techniques for measuring compliance may also be included in this section (as well as identifying parties responsible for the audit).
- **Action:** This section specifies what actions are necessary and when they are to be accomplished. It may identify the time frame in which additional guidance (mentioned previously) will be forthcoming. Hopefully the policy meets the criteria stated previously, but there might be a need for a waiver process. This is one logical place to identify the process as well as the time frame for policy review (and by whom).



Policy Statement Must...

Policy is not detail-oriented; it is a high-level focus of who has to do what. It should not have step-by-step information that is provided in the procedures. The rule of policy assessment is that the policy covers the "who" and "what needs to be done." Procedures cover how to do it. For example, the policy would state that each user must change his or her password every 90 days. The procedures would give you the details of how to change a password on a given operating system. In general, procedures can be updated with far less review than policy. The core characteristics of a policy are:

S: Specific
M: Measurable
A: Achievable
R: Realistic
T: Time-based

Is the Policy...

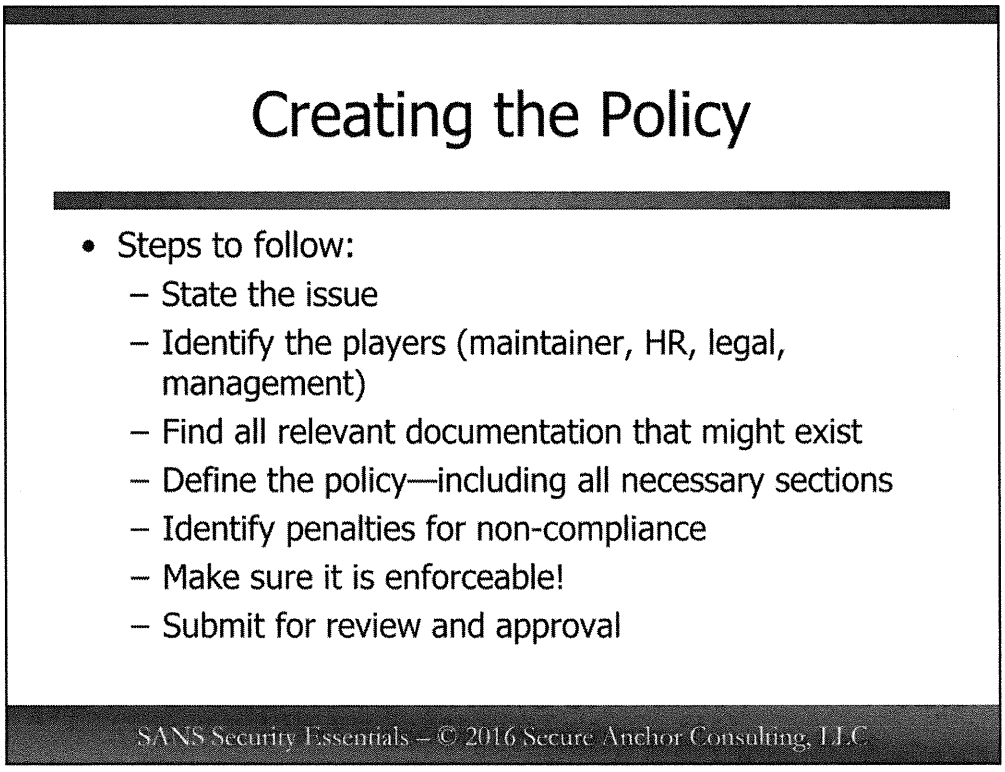
The security policy must also be in accordance with local, state, and federal computer-crime laws. Information Security Management System (ISMS) is a process by which an organization formulates security policy based on ISO17799 Standards. Considering the information in need of securing and the level at which it must be secured, examine the policy to see whether it is consistent with the mission statement and with the following other policies:

- **Program policy:** This high-level policy sets the overall tone of an organization's security approach. Typically, guidance is provided with this policy to enact the other types of policies and who is responsible. This policy may provide direction for compliance with industry standards such as ISO, QS, BS, AS, and so on.
- **Issue-specific policy:** These policies are intended to address specific needs within an organization. This may include password procedures, Internet usage guidelines, and so on. This is not as broad a policy category as the program policy; however, it is broader than the system-specific policy.
- **System-specific policy:** For a given organization, there might be several systems that perform various functions, where the use of one policy governing all of them might not be appropriate. It might be necessary to develop a policy directed toward each system individually. This is a system-specific policy.

If you discover any discrepancies, note them because you will need to resolve them for the policy to be meaningful... Again, note any contradictions you discover so you can get the document corrected.

Examine the policy for provisions to keep it current. Security policy should be reviewed regularly. Revisions in implementation should reflect lessons learned from recent incidents and new threats to the organization's security.

Check whether the security policy is readily available. The policy development guide might provide information regarding the responsibility for publishing and making available specific policy documents. Make sure security policy is incorporated in employee handbooks and posted for reference. It must be required reading as part of the employee-orientation process. We recommend that security officers consider building an intranet Web page.

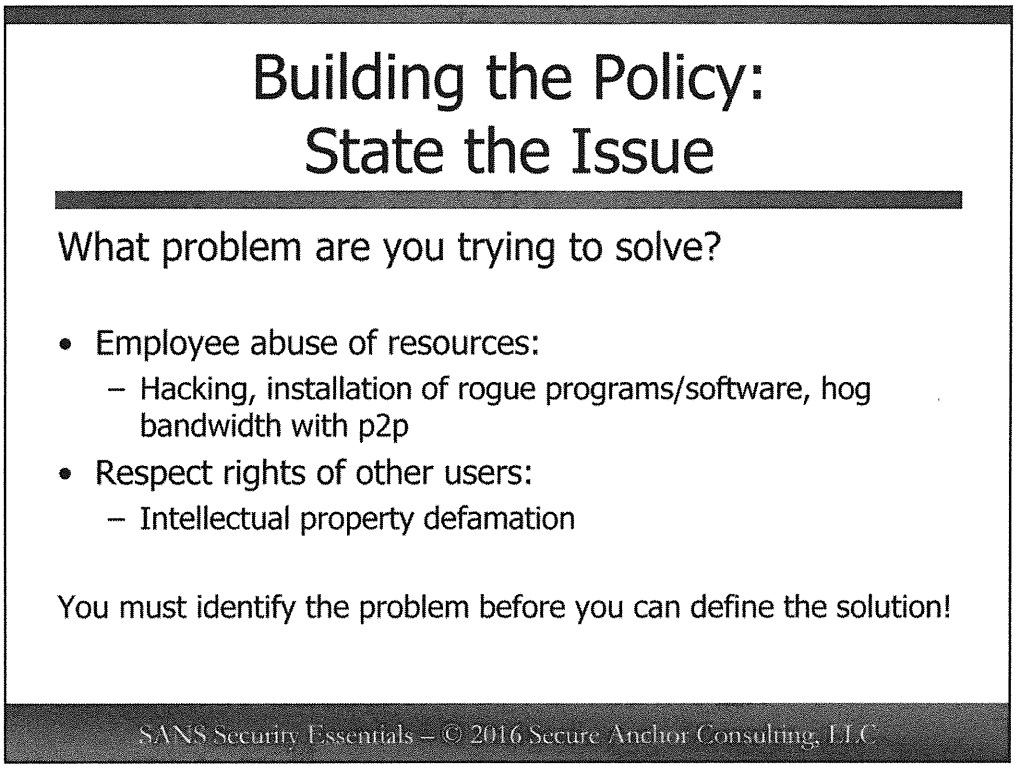


Creating the Policy

You have the basic information that you need to get started. As the policy is developed, you can continue to research the technical controls. As you see each section, one of the important questions to ask yourself is, "Does this policy set the correct tone or posture for my organization?" Obviously, a university will have a different tone to its AUP from that of the Internal Revenue Service or a Department of Energy contractor. However, in every case, if the employees cannot easily read and understand it, the policy will fail to protect information and people, and it might not even prove to be enforceable.

The steps to follow when creating policy are:

- State the issue. What problem are you trying to solve?
- Identify the players. Who needs to be involved in the creation of the policy?
- Find all relevant documentation that might exist. What is already out there that addresses this issue?
- Define the policy. Include all necessary sections, such as background, scope, purpose, responsibility, expiration, action, and related documents.
- Identify penalties for non-compliance. What happens if the policy is not followed?
- Make sure the policy is enforceable. Can it be applied fairly to everyone? Are there any groups that might make enforcement difficult?
- Submit the policy for review and approval. Who has final say on the policy?

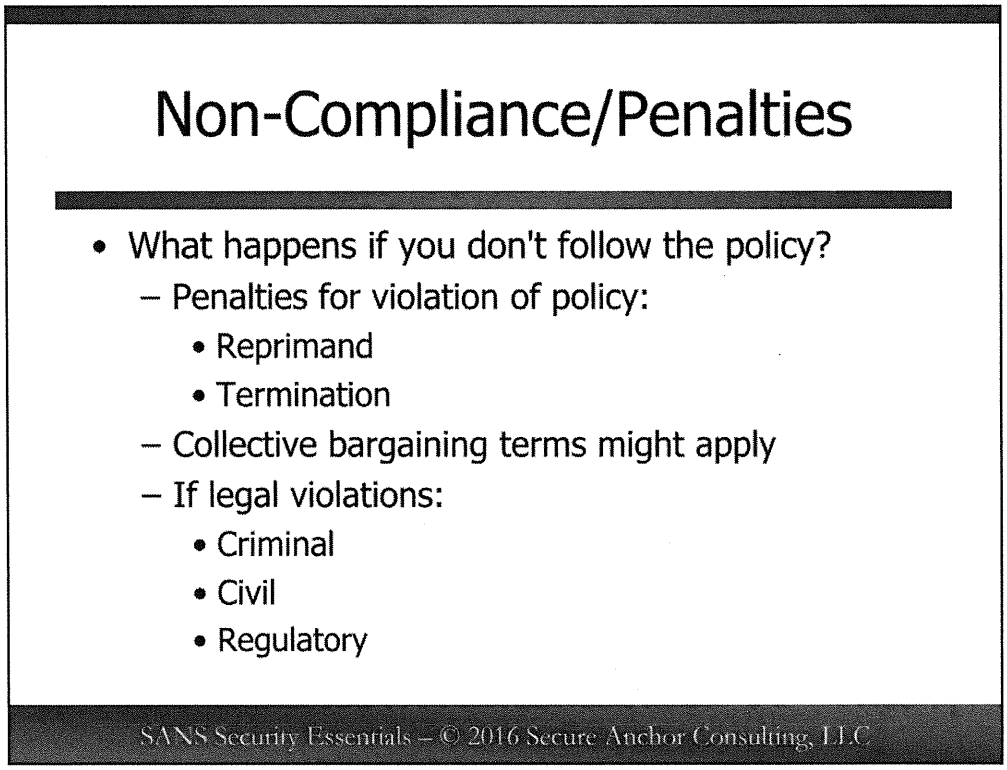


Building the Policy: State the Issue

You must know what the problem is before you can attempt to solve it. Do not begin writing a policy until you have a clear understanding of the problem you want to solve. If your organization has a problem with employee abuse of corporate IT resources, an acceptable use policy might be the correct solution. Here is the first section of a sample AUP:

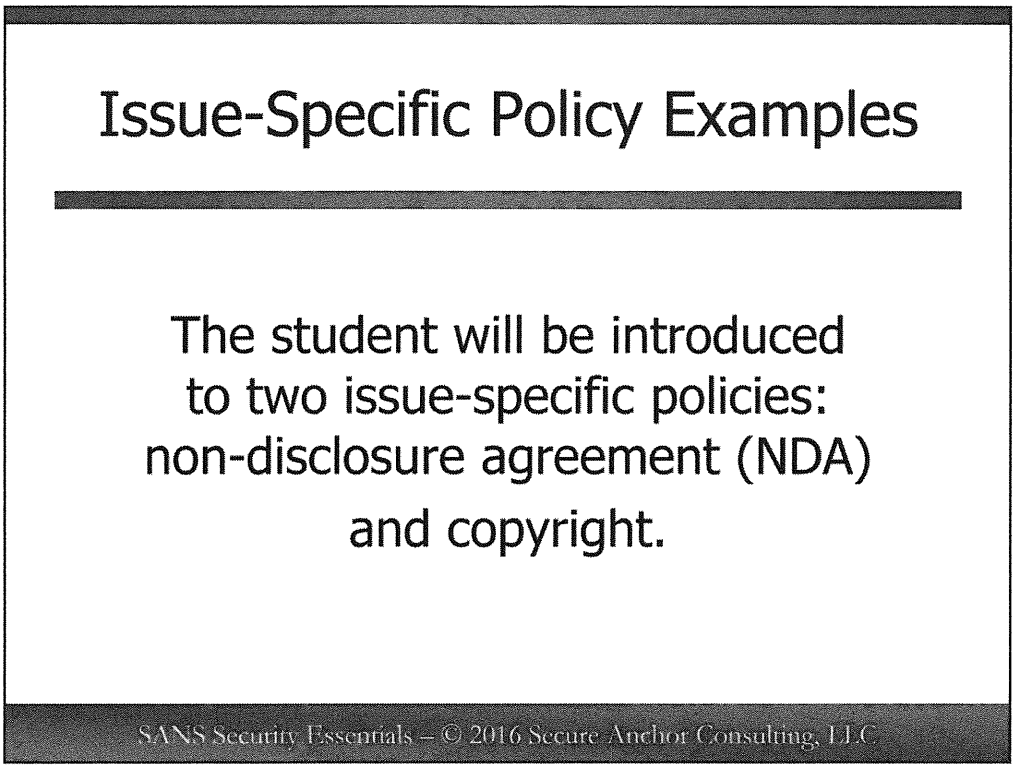
> Information technology (IT) is used daily to create, access, examine, store, and distribute material in multiple media and formats. IT plays an integral part in the fulfillment of *Organization's* research, education, administrative, and other roles. Users of *Organization's* resources have a responsibility not to abuse those resources and to respect the rights of the members of the community. This IT Acceptable Use Policy (the "Policy" or "AUP") provides guidelines for the appropriate use of *Organization's* IT resources.
>
> Both the rights of academic freedom and freedom of expression apply to the use of *Organization's* computing resources. So too, however, do the responsibilities and limitations associated with those rights. However, the use of Corporate computing resources, like the use of other Organizational resources and activities, is subject to the requirements of legal and ethical behavior. Thus, legitimate use of a computer, computer system, or network does not extend to whatever is technically possible.

This text comes from the policy at http://www.cio.ufl.edu/policies/aupolicy.html. The key to understanding this organization's position is to realize that network and computer users do have a presumption of privacy.



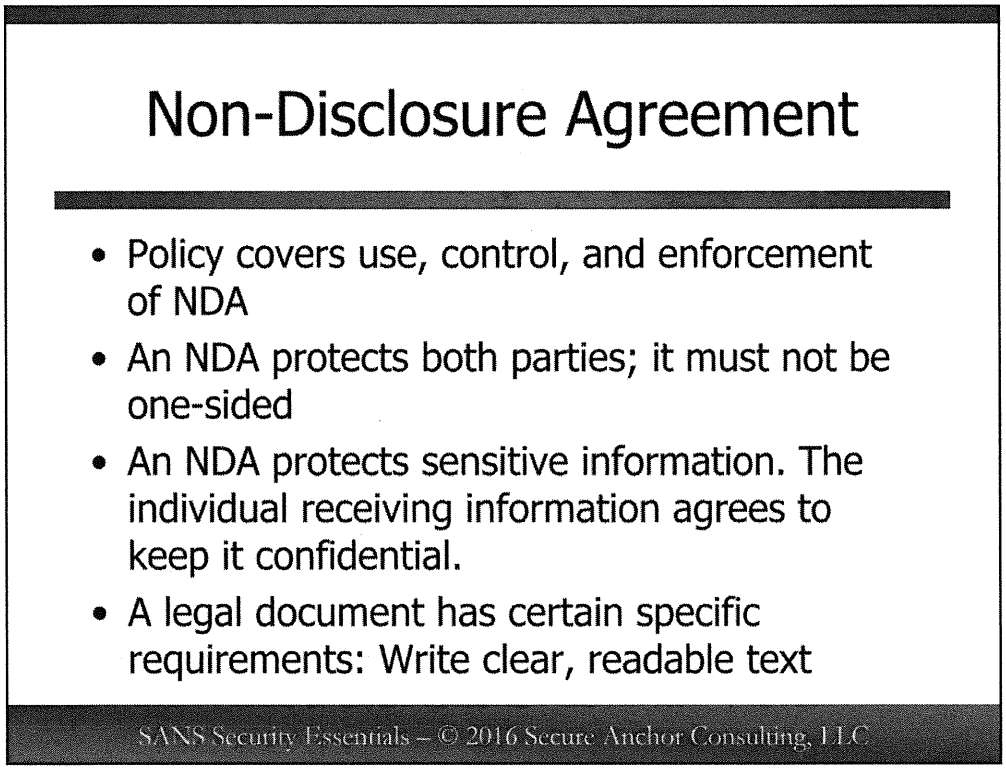
Non-Compliance/Penalties

Penalties for violation of this policy will vary depending on the nature and degree of the specific violation. Penalties range from reprimand through termination for employees in accordance with the provisions of any collective bargaining agreement, to the extent such agreement applies to the employees. If violations of law are involved, users might incur civil liability to the organization or third parties and might also be subject to prosecution.



Issue-Specific Policy Examples

This section intentionally left blank.



Non-Disclosure Agreement

Let's start with an example using a non-disclosure agreement (NDA). This is an extremely wise tool for the protection of people and information. At SANS, every time courseware is sent to anyone other than the author, an NDA is sent along with it. Everyone who works on the Global Information Assurance Certification (GIAC) certification also signs an NDA.

Essentially, an NDA is a legal document that protects the parties involved in specific circumstances. In the case of SANS, the courseware or GIAC certification are the circumstances in which parties might be working together. The owner of the information is protected, and the individual who signs the NDA is also protected.

The NDA is an agreement between parties. The policy covers the use, control, and enforcement of the NDA. When creating the policy, remember that it must contain all of the components essential to any good policy. The policy should include *who* the NDA applies to, *what* the NDA is, *when* the NDA should be used, *where* the NDA is applicable, and *why* the NDA is important.

NDA Protects Both Parties

The following text is a quote from the SANS NDA:

"INDIVIDUAL agrees to maintain in confidence SANS' Confidential Information. INDIVIDUAL will use SANS' Confidential Information solely to undertake the tasks required under the relationship pertaining to the Business. INDIVIDUAL will not disclose SANS' Confidential Information to any person.

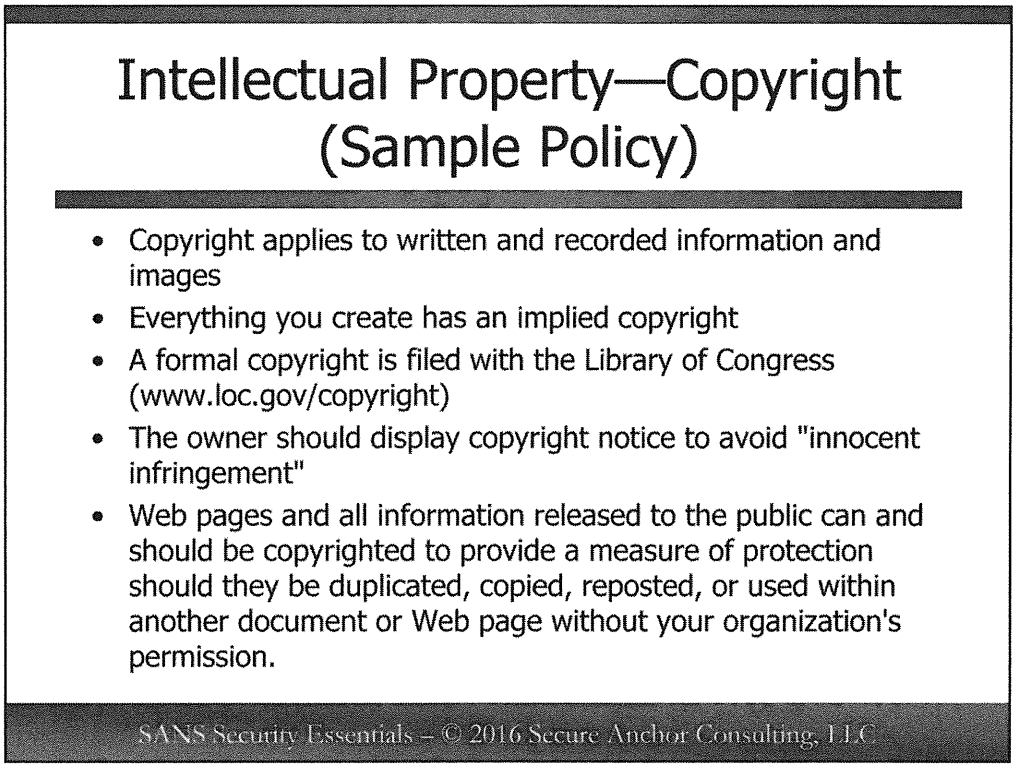
This obligation will not apply to the extent that INDIVIDUAL can demonstrate that:

(a) SANS' Confidential Information at the time of disclosure is part of the public domain;

(b) SANS' Confidential Information became part of the public domain, by publication or otherwise, except by breach of the provisions of this agreement;

(c) SANS' Confidential Information is received from a third party who is not an agent of SANS without similar restrictions and without a breach of this agreement;

(d) SANS' Confidential Information is required to be disclosed by a government agency to further the objectives of this agreement, or by proper court of competent jurisdiction; provided, however, that INDIVIDUAL will use their best efforts to minimize and will consult and assist SANS in obtaining a protective order prior to such disclosure."



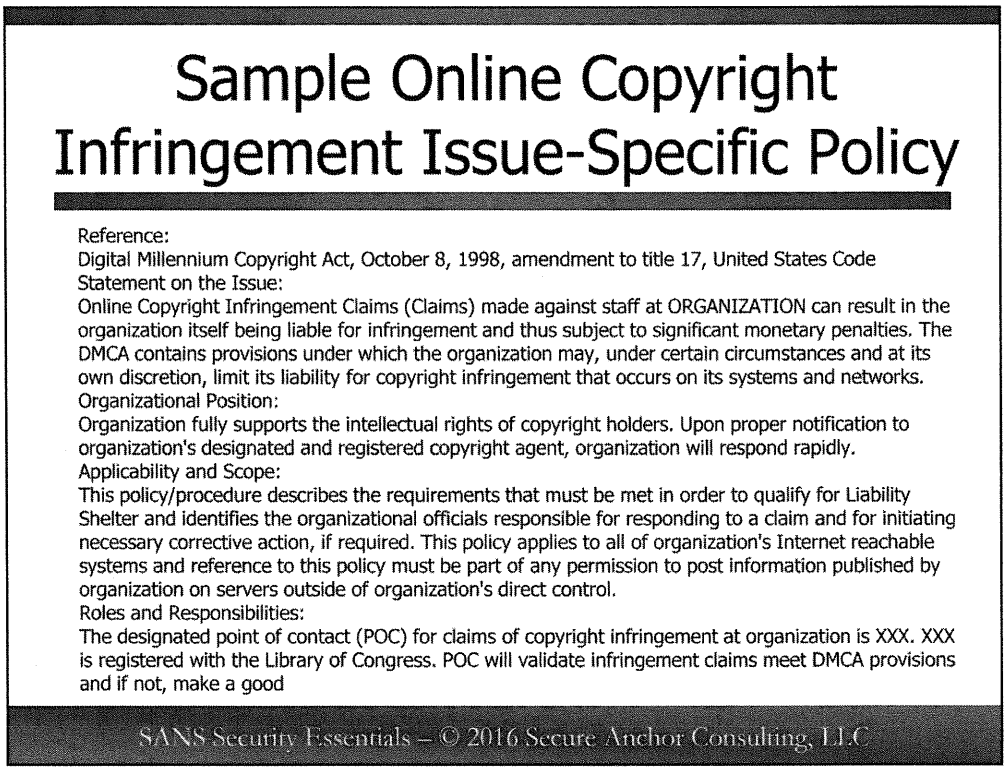
Copyright law is fairly consistent globally based on the work of the Berne Convention, but the information on the preceding slide is based on the US Library of Congress interpretation.

Intellectual Property —Copyright (Sample Policy)

Copyrights primarily protect the author or the owner of a piece of information. If two parties agree that a written piece is work for hire, then the rights become that of the owner. Any organization's document that is published, such as a flyer, manual, or Web page, should have a copyright notice to identify the year of publication and name of the copyright owner. This is not a requirement of U.S. or international law, but it does give additional protection.

Registering a copyright is also not required but gives additional protection. If there is any reasonable chance your organization would legally pursue a violation of your copyrighted material, then you should register that work. Registration is a requirement before you can file an infringement suit in court (for works of U.S. origin).

Registering a copyright is simple, in most cases, and requires sending three elements in the same envelope to the Library of Congress: a properly completed application form; an application fee (currently $35); and a "deposit" (sample copy) of the work.



Sample Online Copyright Infringement Issue-Specific Policy

To receive safe harbor protections under DMCA, two of the most important provisions are that a service provider must appoint a point of contact that is registered with the Library of Congress and post a DMCA policy. Following is an online copyright infringement, issue-specific policy:

Roles and Responsibilities:

POC will validate infringement claims meet DMCA provisions and if not, make a good faith effort to gather such information. POC will serve as liaison between claimant and organization and make a first effort determination as to accuracy of claim. POC will be responsible for record keeping; all communications will be stored for at least three (3) years from time of initial claim.

Compliance:

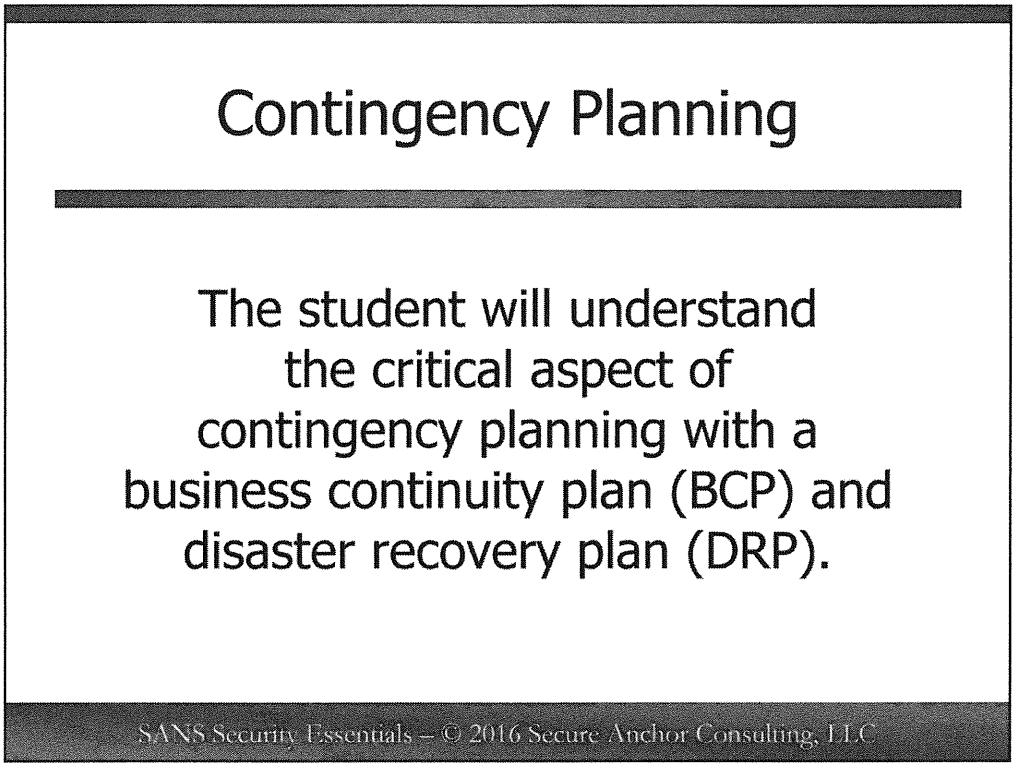
POC will validate the location of the claimed infringing material.
POC will determine authorship of claimed infringing material.
POC will send validated package to appropriate organizational director.

Organizational director will:

Inform author of alleged infringing materials that a Claim has been lodged. At his/her discretion, may inform the publisher that he/she may submit a counter notification if there is reason to believe that the Claim is mistaken.

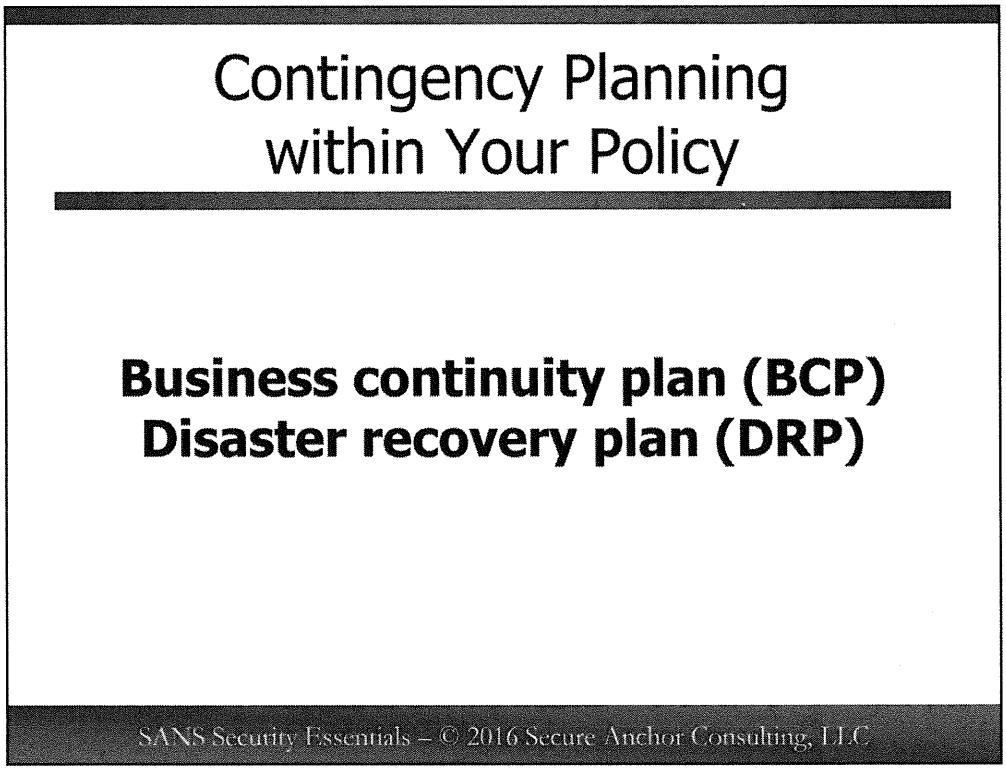
Within two business days, Organization Director, after consultation with appropriate individuals shall decide whether access to the material should be blocked as a stopgap measure and potentially removed. Organization director informs POC of final decision.

POC notifies publisher of infringing materials and claimant of decision in writing (electronic mail or paper) and copies the Designated Agent, and creates an offsite archive copy of the entire transaction to be stored at YYY.



Contingency Planning

This section intentionally left blank.

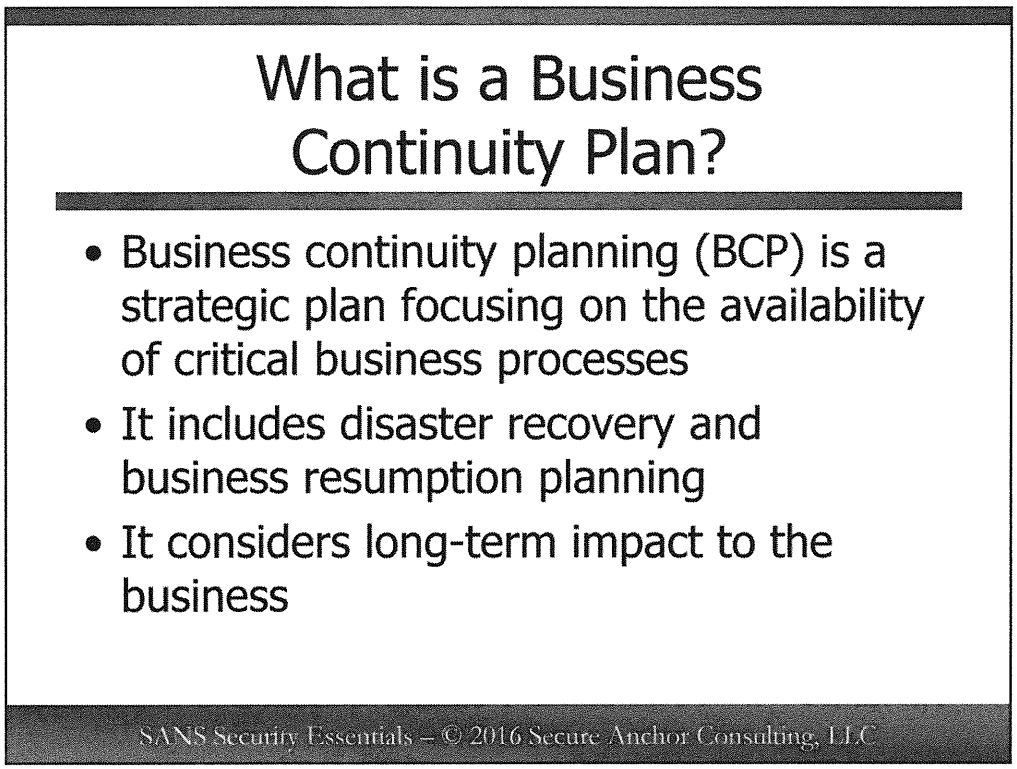


Contingency Planning within Your Policy

A critical aspect of security policy for your organization is planning for contingencies. In this section, we give you an overview of contingency planning—what it is and why you need it—and then we walk you through the contingency planning lifecycle. You will be equipped to create a contingency plan for your organization and to provide references for additional reading.

Overview of Contingency Planning

First, we define what a business continuity plan (BCP) and disaster recover plan (DRP) are, and we explain why an organization needs them. Subsequently, we dive into the process for developing a BCP and DRP.



What is a Business Continuity Plan?

Introduction
A business continuity plan (BCP) is defined as a plan for emergency response, backup operations, and post-disaster recovery maintained by an activity as a part of its security program that will ensure the availability of critical resources and facilitate the continuity of operations in an emergency situation. Business continuity planning (BCP) enables the quick and smooth restoration of business operations after a disaster or disruptive event.

What Is a BCP?
The BCP is an overarching plan that details recovery from a disaster and business resumption planning, as well as a compilation or collection of other plans including:

- Disaster recovery plan
- End-user recovery plan
- Contingency plan
- Emergency response plan
- Crisis management plan
- Other plans as required (for example, a server recovery plan or a phone system recovery plan)

Why You Need a BCP?
A BCP is a business' last line of defense against risks that cannot be controlled or avoided by other risk management practices. In addition to an immediate action plan for recovering the business, the BCP should also consider a long-term plan that keeps the business running. For example, after the company has relocated resources and established operations in a new location, how should the business work to re-establish the disaster site? Long-term planning should also include public relations and possibly marketing, with a plan to maintain the positive, reliable image for the company

following a disaster. The BCP doesn't just define how a company should react to a disaster to keep the business operational—it must also define how the business will restore 100% of the operation including the ability to continue to meet defined business goals.

Example of a BCP

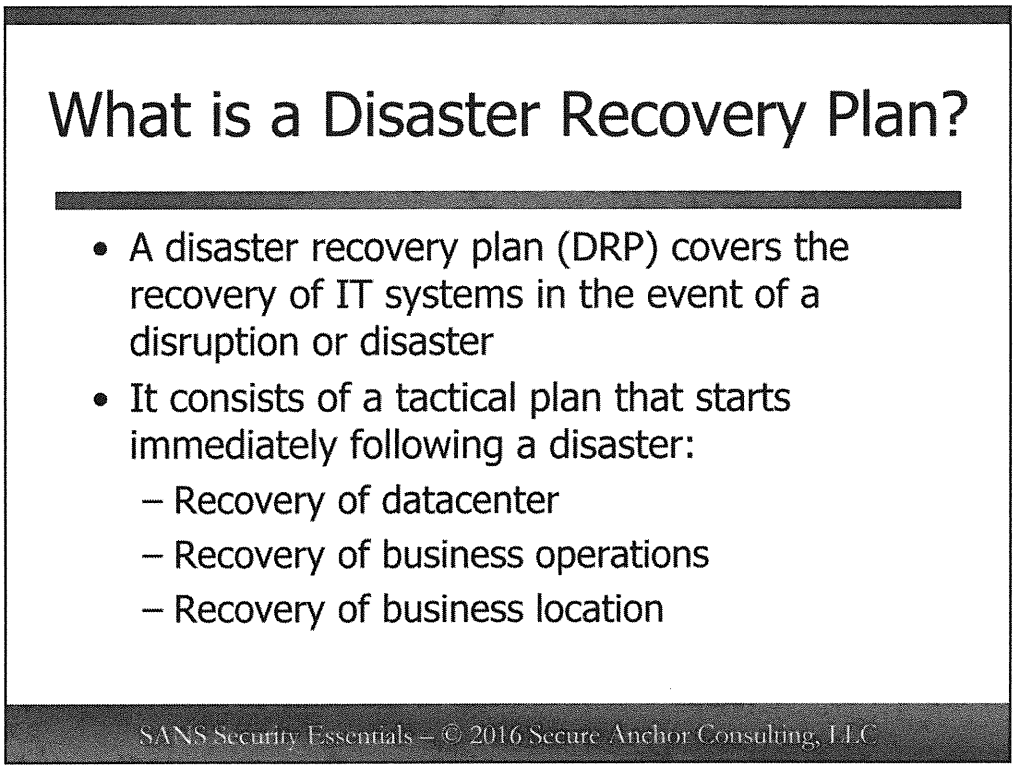
One organization, Morgan Stanley Dean Witter, was blessed with an individual who had the foresight to plan for the attack on the World Trade Center. Rick Rescorla was a unique individual. In 1985 and again in 1993, he identified the World Trade Center as a target for terrorist attacks. Rescorla is a prime example of what one person can do when committed to doing the right thing. Hundreds of people who were in those buildings on September 11 are alive today as a result of Rescorla's planning and because they practiced for such an attack.

Business Resumption Planning

Business resumption planning (BRP) is the generic term used to refer to the actionable plan that coordinates efforts to restore an organization to normal working order. This concept encompasses a wide-scale of topics, from the immediate plans to restore business operations, to long-term business resiliency planning that will help an organization maintain a polished and undeterred image for consumers, even when faced with a disaster.

Summary

Like a business continuity plan, the BRP doesn't just involve IT, it involves all levels of the organization. The best BRPs that I've seen include how the organization will continue to meet and exceed the defined goals for a business following a disaster.



What is a Disaster Recovery Plan?

Introduction
A disaster recovery plan (DRP) covers the tactical recovery of IT systems in the event of a disruption or disaster. It provides the capability to process essential organizational applications, even if they are not operating at 100% efficiency, in addition to the ability to return to normal operations within a reasonable amount of time.

BCP and DRP
The terms *BCP* and *DRP* are often used interchangeably, but are actually two distinct measures that tackle different areas of the recovery process. Business continuity planning deals with the restoration of the business processes—or the continued operation of a business process: organizational processes could operate without computers. For example, checks can be written by hand. With the continuity plan, the company can reduce the impact a disaster could have on the normal business operation. The disaster recovery plan covers the restoration of the critical information systems that support the business processes.

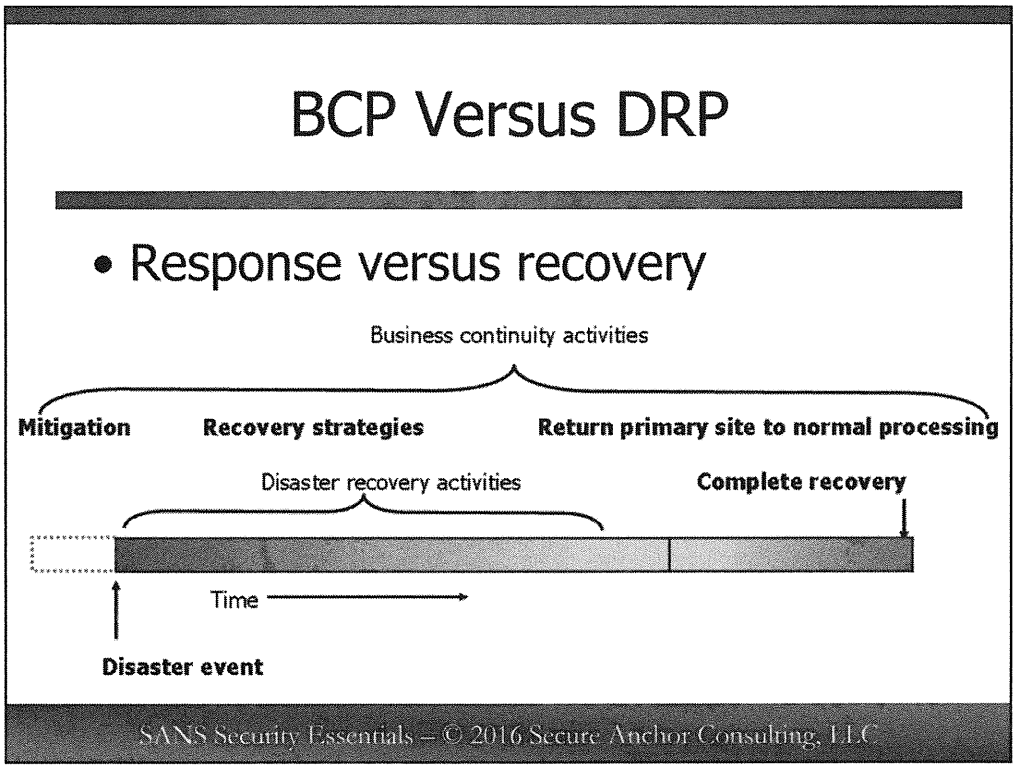
Disaster Recovery Planning
Disaster recovery planning involves the following steps:

1. The recovery of the datacenter. Because the DRP relates to the restoration of the information systems, the datacenter is one of the critical areas the DRP should address in terms of how to bring it back on line.
2. The recovery of business operations. This is sometimes referred to as user contingency planning. If a critical computer system is down, this part of the DRP deals with the alternative methods of continuing with the business operations. For instance, if your main payroll system were inoperable, a contingency plan could be to issue the payroll checks manually.

3. The recovery of the business location. As part of the business resumption plan, this section deals with the steps required to recover the actual physical business location. Often a disaster is partial, and recovery of the premises might consist first of patching together what is left, followed by backfilling what has been lost.
4. The recovery of business processes. Also part of the business resumption plan, this section handles the recovery of all of the various business processes, so that the company can resume normal business operations. This is the paramount step. The whole purpose of the plan is not about computers, networks, and data, but about the timely continuity and restoration of business processes.

Summary

Unlike the BCP, the DRP consists of tactical action items that take place following a disaster. Where a BCP will contain high-level language that is appropriate for assessing the stability and continued operation of the business, the DRP provides clear and concise instructions that will be followed in the event of a disaster. The DRP documentation might even contain checklists for accomplished tasks as a reporting mechanism for the disaster recovery team. This type of documentation is well-suited for the basis of training materials, and should be clearly written and easy to follow and understand such that there is no room for misinterpretation when following the plan. When responding to a disaster, the last thing you want is taking incorrect actions to "fix" the problem, or completing actionable items out of order.



BCP Versus DRP

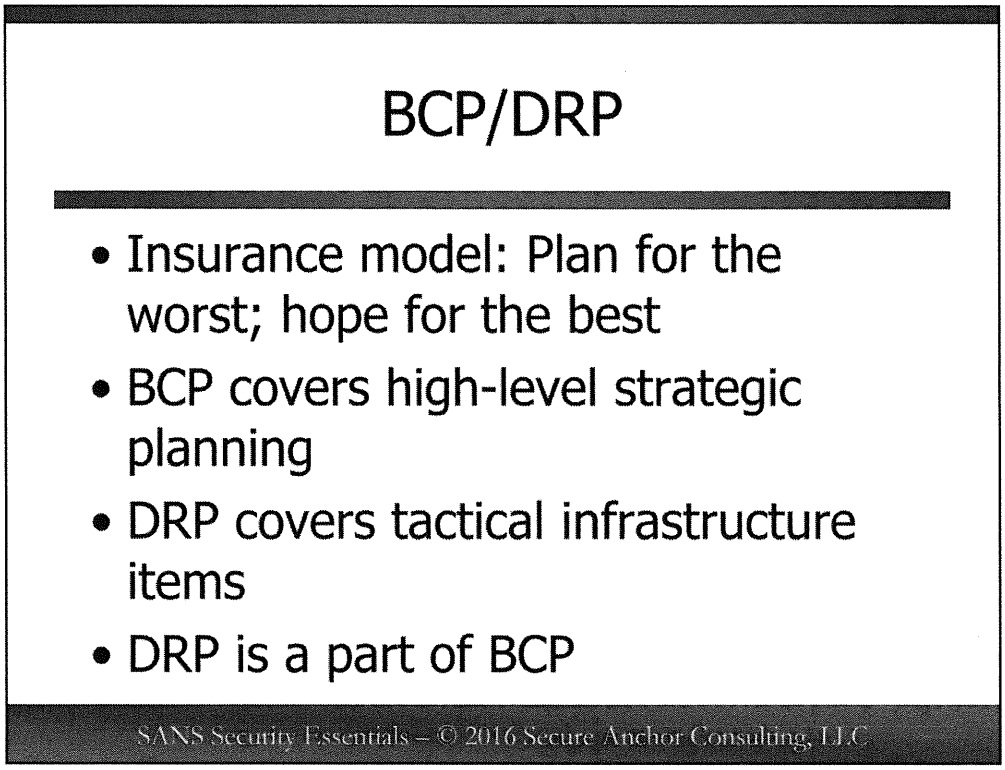
Disaster recovery provides a response to disruption, whereas business continuity planning implements the recovery. The preceding figure shows that the disaster recovery activities have a short time span, but business continuity activities are much more pervasive and long-lasting.

The goal of BCP/DRP is to make the response time to a disruption and the time required for complete recovery as short as possible.

During disaster recovery activities—that is, when a disaster strikes an organization—almost all normal business activities are heavily modified, reduced, or completely suspended. Only critical business processes resume, and usually at an alternate site.

As repairs are completed, normal business activities resume as the business continuity plan dictates. Recovery is complete after all normal business processes return to "business as usual."

Business continuity activities form an umbrella over a crisis situation, while disaster recovery activities are a *subset* of business continuity activities.



BCP/DRP

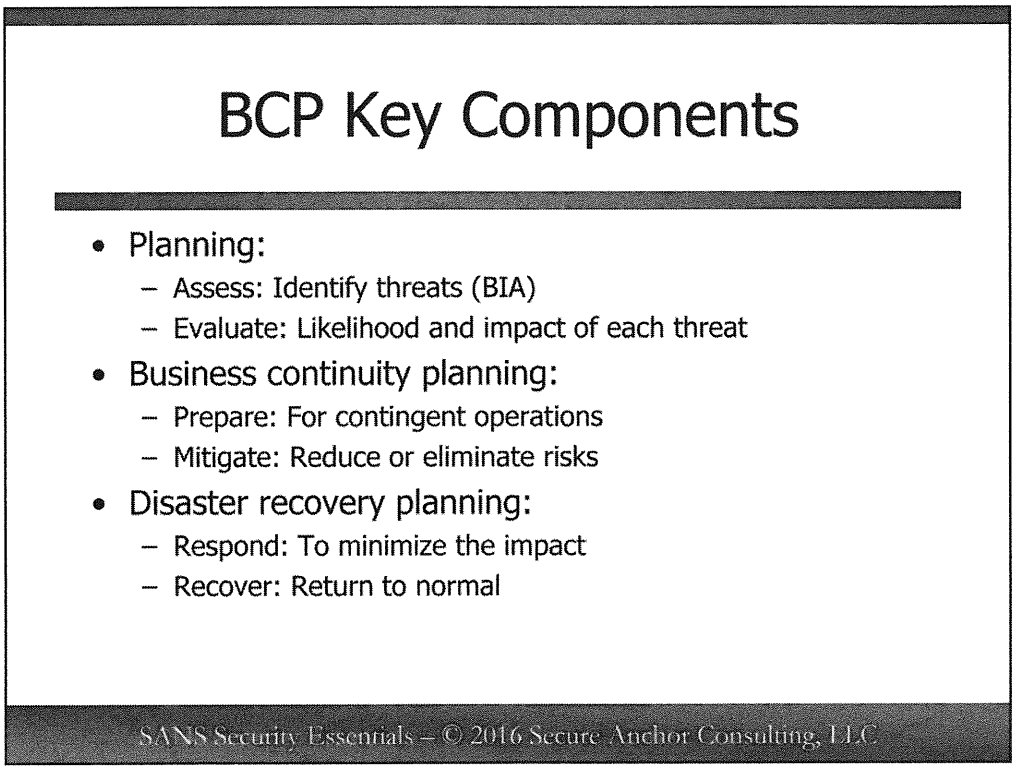
Introduction
Continuity planning might be likened to insurance; it's an expense you consciously make to significantly reduce the impact of something bad that occurs. Although you pay the premium, you hope that nothing bad occurs. Even if it does not, the insurance premium and the expense of continuity planning are not wasted. They purchase certain assurances as a key component of the organization's risk management. As a wise man would say, "Plan for the worst; hope for the best."

Split Operations Model
The key component of a continuity planning is to enable your business to continue to operate. Having a split operations model allows two or more sites to actively cover one another for extended periods of time if needed. This model addresses a lot of the vulnerabilities of the Classic model where all backup materials were on-site, or nearby which doesn't help when in the case of 9/11 backup materials were sometimes in neighboring buildings.

International organizations and nationwide firms can utilize this model on different coasts and even different continents. In this way, routine workloads can be distributed among their locations. Consider two sites, one called "Germany," one called "Australia." Some organizations even do testing for patches and software updates on the backup site (Australia).

If it works for a week or ten days, they make the former backup site (Australia) active; Germany, then, becomes the backup site. If the new active site (Australia) stays stable while in production for a week or ten days, they then move the patches onto the current backup site (Germany). At this point, both sites are completely identical and fully capable of being a backup for one another.

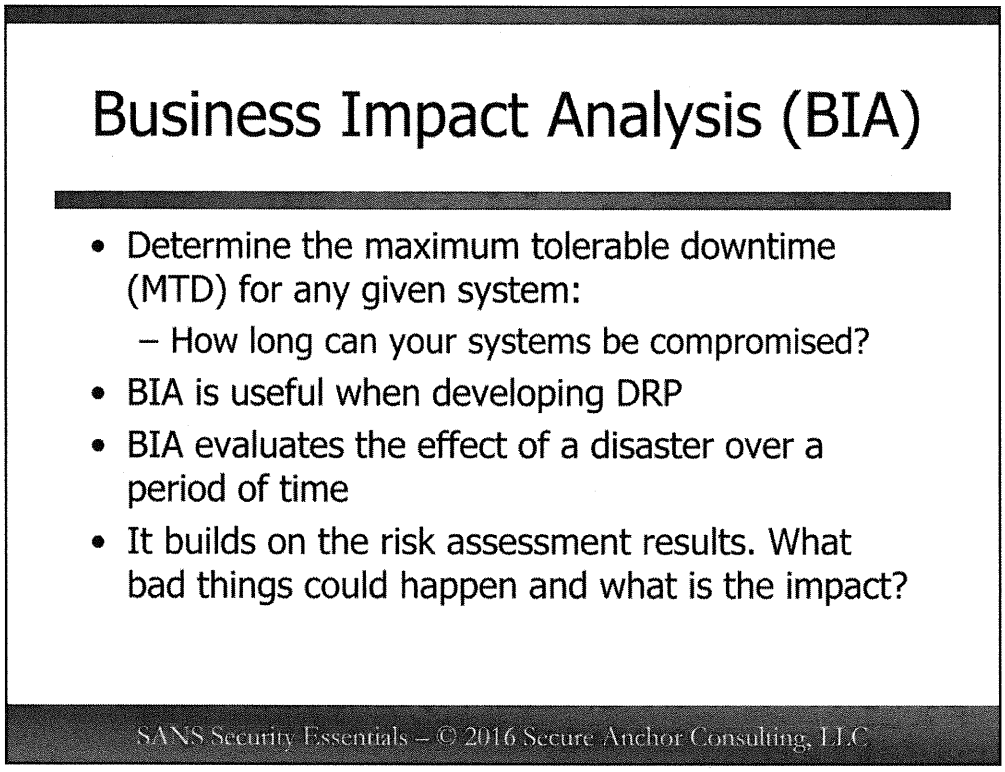
Summary
It is always important to remember that information is the life line of any business. If the information or access to that information goes away, the company would have monetary impact and go out of business. Although redundant sites can be expensive, it is less expensive than going out of business.



BCP Key Components

The key components of a business continuity plan are:

- **Assess:** Identify and triage all threats. This assessment is the beginning of the business impact analysis (BIA). We must understand what the threats are and assess the impact the threat would have on the business if the threat were to become reality.
- **Evaluate:** Assess the likelihood and impact of each threat. Realistically, what is the chance that the threat will happen? Perform the cost-benefit analysis to ensure any investments are justified.
- **Prepare:** Plan for contingent operations to occur within the necessary time frame. This step includes not only preparation of the BCP, but also ongoing management of the plan. Ensure employees are properly trained and all documentation is in order. Perform periodic testing of the plan in accordance with your policy.
- **Mitigate:** Identify actions that might eliminate risks in advance. Are there things we can do that will decrease the likelihood of the threat becoming a reality? Are they cost-justified?
- **Respond:** Take actions necessary to minimize the impact of risks that materialize. When disaster strikes, a quick response can minimize the impact to the business. Organizations that are well-prepared are in a better position to respond quickly than those that have not thoroughly planned for disasters.
- **Recover:** Return to normal as quickly as possible.



Business Impact Analysis (BIA)

Introduction
The business impact analysis (BIA) documents what impact a disruptive event might have on a corporation. The BIA prioritizes business functions versus risks to identify the criticality of functions and the timeframe for which they must recover. Some business functions, if down for only a few seconds, might dramatically and detrimentally impact the business. Other business functions might be interrupted for days or weeks and have no negative effect on the corporation. From a big picture BCP perspective, the BIA helps us focus on those areas of our business that must have priority when recovering from a disaster.

BIA
The primary goal of the BIA is to determine the maximum allowable downtime for any given system, or maximum tolerable downtime (MTD). Understanding the MTD for business processes is mandatory before designing your disaster recovery plan. Without the MTD calculation for systems, you won't know whether the plan meets or exceeds the requirements of the business. Although exceeding business requirements is usually a good thing, the cost of doing so might not be.

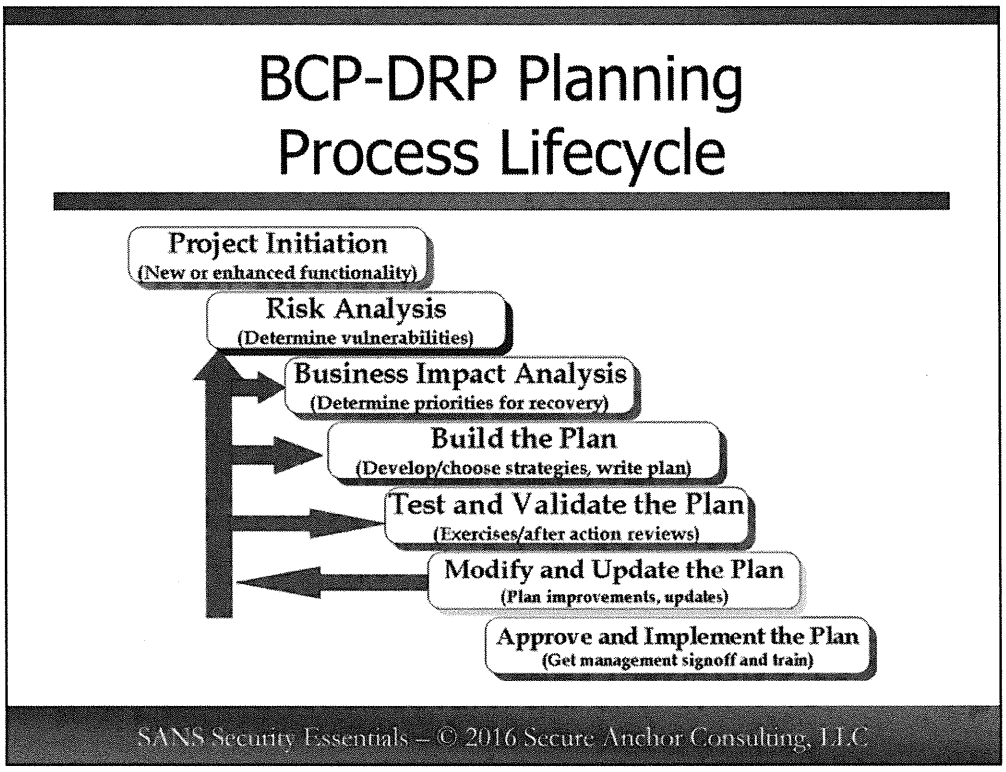
Developing the BIA
The process of developing the BIA typically involves interviewing the various key users of the various computer systems (for example, payroll, accounts payable, and accounting) to get a better understanding of how a disaster could impact the ability to continue operations. Some of the key interview questions might include:

- What would be the impact of an information technology failure on cash flow and revenue generation?
- Would the disaster impact the level of service?

- How long could the outage last before it began to affect your productivity?
- Would there be irretrievable loss of data?
- What are the key resources that are required to be kept operating?
- At what point would those resources need to be in place?
- How does this process/system interact with other processes and systems? What are the dependencies on this process?

Summary

When putting together the BIA, the answers should come from, or be concurred by, executive management. At that level, management understands cost tradeoffs such as between mitigation and loss, and has individual accountability either way. Lower management might err toward too much (for example, too expensive) risk avoidance, whereas upper management might prefer to accept certain risks and redirect mitigation resources elsewhere in the business. This is a common mistake in BCP/DRP planning.

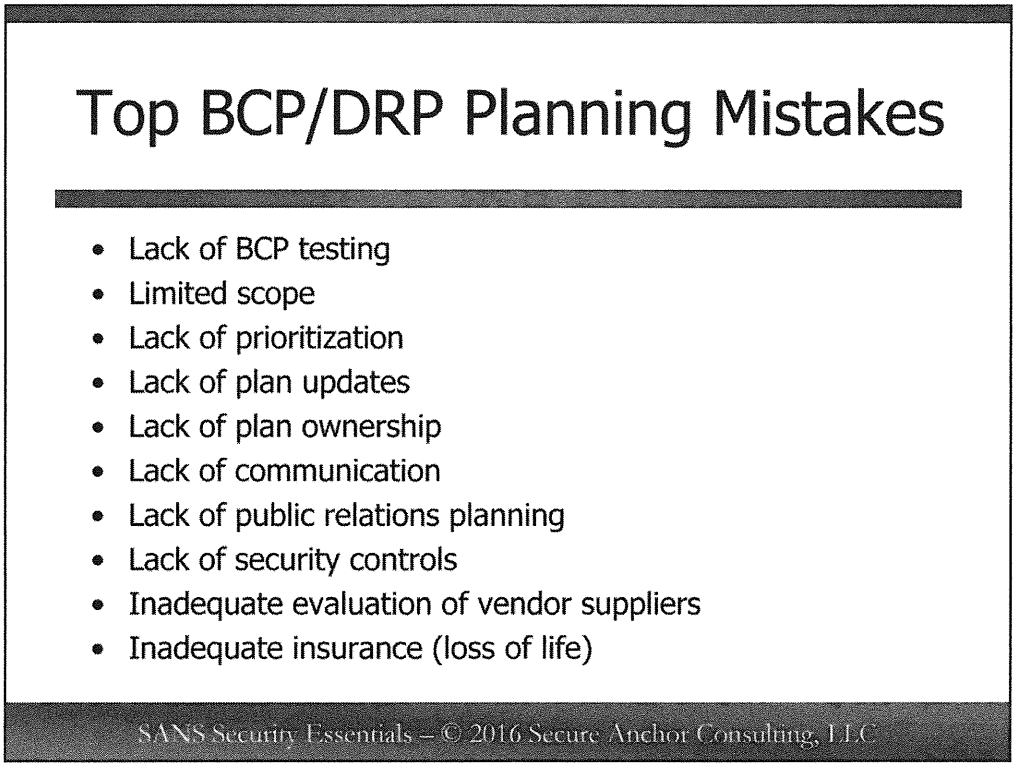


BCP-DRP Planning Process Lifecycle

This slide shows the basic steps that are necessary when developing a BC/DR plan. We start with Project Initiation, for which new or enhanced functionality is required. At this point, you must get management approval to start the project. Management is instrumental in making sure that you have access to the resources that are required to get the job done. The next sequence of steps in the process concerns the company's vulnerabilities, their significance to the company, and what the company is going to do about them.

First, the company determines its vulnerabilities through a Risk Analysis. The company then assesses the impact that each of these vulnerabilities represents for the company by completing a Business Impact Analysis. Realistically, no organization has the resources to deal with every vulnerability. Instead, in this step, the company prioritizes the vulnerabilities based on their likelihood and impact. Those vulnerabilities that represent greater risk to the company are identified so that steps to avoid their occurrence can be planned. In the event that those plans fail, the prioritized vulnerabilities can also be given priority in terms of recovery of affected operations.

Remember, not all losses are directly associated with loss of money (although it will most likely affect the company financially in the long run). Do not forget to include the "intangible" losses, such as customer satisfaction or loss of consumer confidence. For instance, if a major e-commerce shopping site is down for a long time, consumers will become frustrated and will perhaps begin shopping somewhere else. At that point, it does not matter what caused the problem: earthquake, flood in the datacenter, or denial of service attack. The fact is that the site was down. The faster the company is able to recover, the better. Conversely, professional handling of a disaster can actually improve an organization's reputation with its customers and other stakeholders.



Top BCP/DRP Planning Mistakes

A number of other mistakes are commonly —almost predictably—made in contingency planning. These include:

Lack of BCP Testing/Over-Reliance on BCP

Many companies believe that just having the BCP is enough. The document is just a lifeless draft without adequate updating and testing. Organizations that test their BCP consistently find areas needing improvement and often critical flaws. The time to discover these is in advance of a real disruption. "Practice makes perfect." For less expensive testing more frequently than your organization can afford full-fledged, off-site tests, try simulating a disaster, as in a business simulation game. Pretend something has happened, certain resources are no longer available, and have your personnel (who are assumed available) walk through the plan.

Too Limited in Scope

An incomplete BCP will not address all of the organization's needs for recovery. The BCP needs to cover organizational processes and process dependencies, systems recovery, as well as the replacement of key personnel if needed. The organization needs to continue to function throughout a disruption and beyond.

Lack of Prioritization

There is a need to prioritize the key business processes. The risk is to prioritize less-than-critical processes instead of the ones crucial for business survival. This is a time for thoughtful evaluation and decisions.

Lack of Plan Updates

The BCP should be updated periodically, especially when there are significant system or business process or personnel changes.

Lack of Plan Ownership

Someone with the power to lead, influence, prioritize, and organize the BCP is instrumental to the success of the program. This is true during planning, as well as during execution of the plan.

Lack of Communication
There is a need for clear and precise communication with all affected stakeholders of the organization, potentially: employees, contract employees, vendors, business partners, customers, and shareholders. (This relates to Public Relations planning next.)

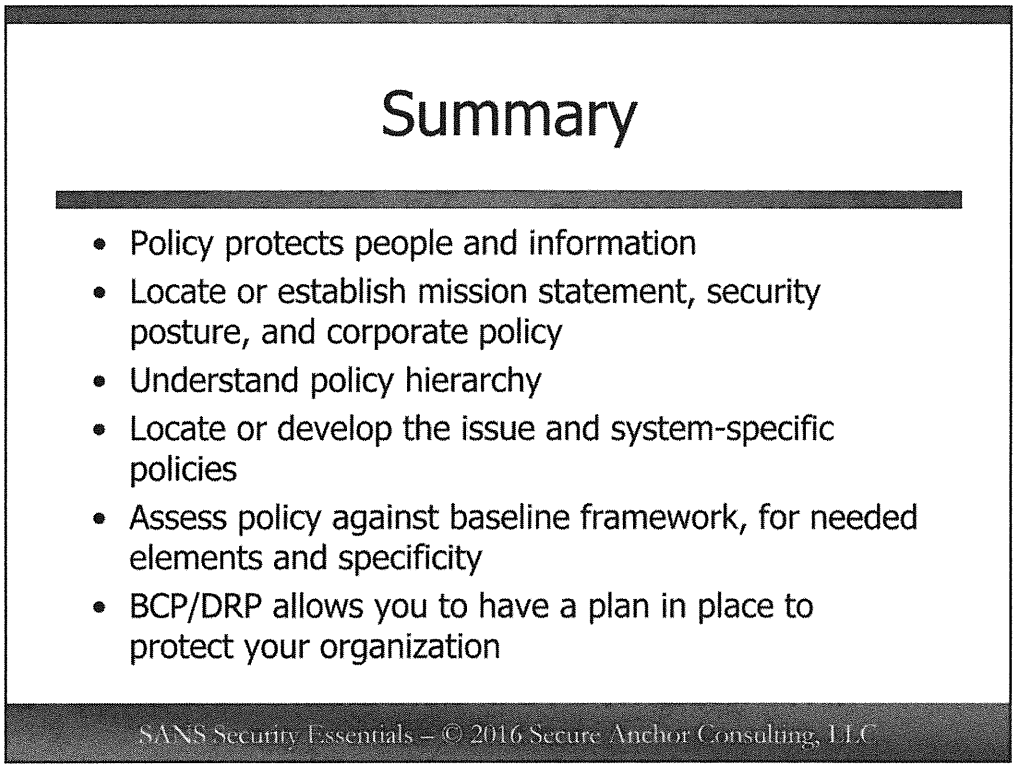
Lack of Public Relations Planning
Organizations often fail to consider public and investor relations, to limit the perceived disaster impact. This can literally make or break the organization. Remember the Tylenol tampering scare some years ago and how the strong PR from that company turned a disaster into a marketing opportunity?

Lack of Security Controls
During the recovery process, sometimes security controls are disregarded, resulting in a greater risk of exposure. Security controls likely might need to be altered and loosened during recovery. But, this should be a matter of a conscious decision and empowerment that are built into the plan. During execution of the plan, there should be strict adherence to the security controls incorporated into the plan.

Inadequate Evaluation of Vendor Suppliers
Many companies poorly evaluate recovery products (hot site, cold site, and planning software), relying on vendor-supplied information. This often leads to a solution that might not adequately address a company's needs.

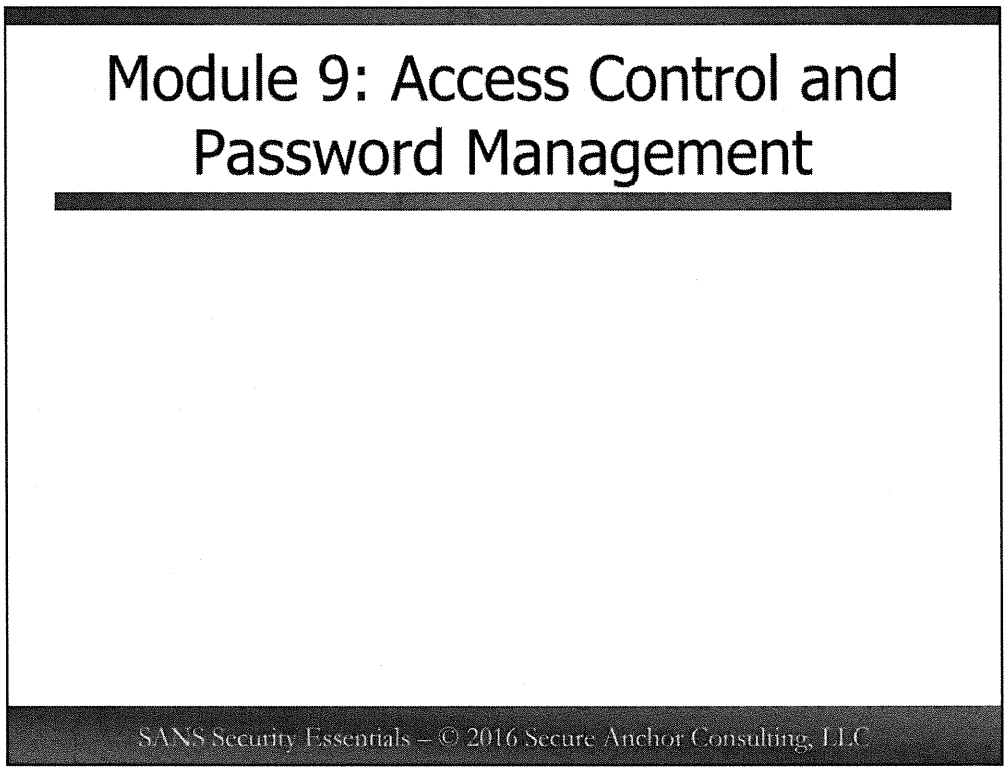
Inadequate Insurance
Some organizations lack adequate insurance coverage and fail to support the filing of insurance claims, and these inadequacies result in delayed or reduced settlements. The plan might lack appropriate processes for capturing losses and recovery costs, without which the organization might realize a loss greater than otherwise necessary.

Summary
Use these examples of common mistakes as a checklist to review your organization's contingency planning—the documentation, testing, integration with organizational processes and personnel, and so on.



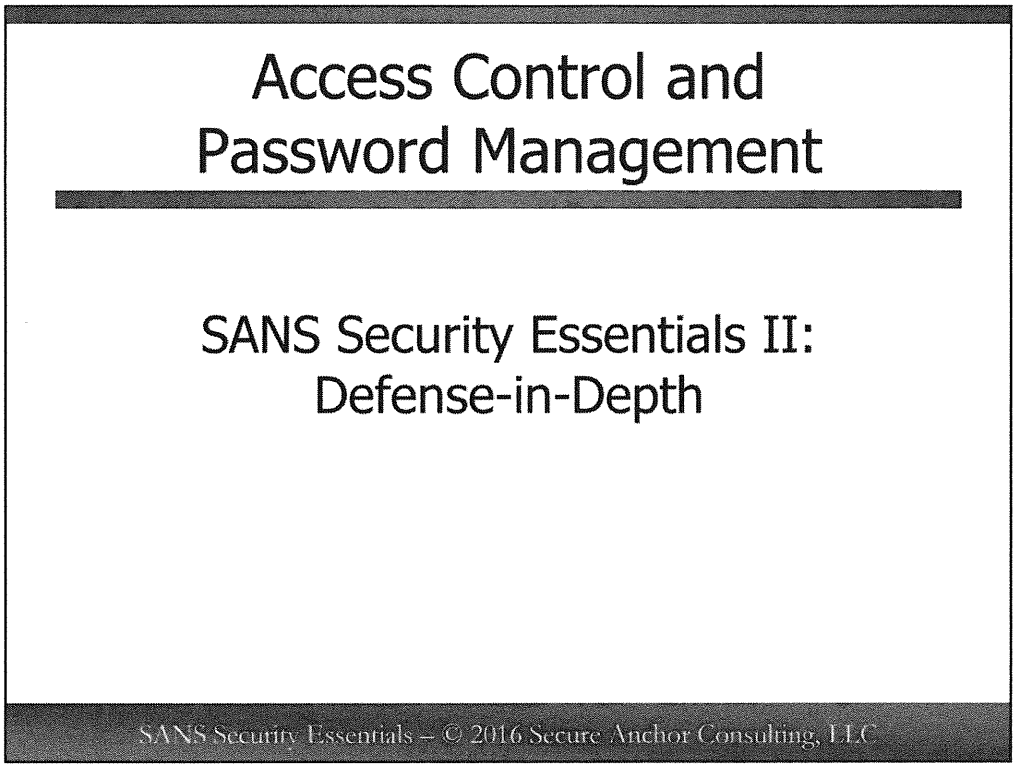
Summary

You now have a big-picture approach for evaluating policy. When you get back to your organization, we hope that you will put what you have learned into practice! Take the steps in order and the framework will work for you. Revisit your mission statement and ask questions to see whether your organization is living up to its mission. Start with a corporate policy, ensure you have the support of senior leadership, and help them state that good security is good business in a manner that cannot be misunderstood. Make sure you have required policies, that they have the required elements and that they are clear, concise, and SMART. Use a detail-oriented person to help you assess the specificity of the policy. Someone familiar with the topic might "autocorrect" or "fill in" any errors or omissions. If you wrote the policy, employ someone that knows the organization and fundamentals of information security to review the content. Remember that the threat level can change and policy should be reviewed in the light of major changes.



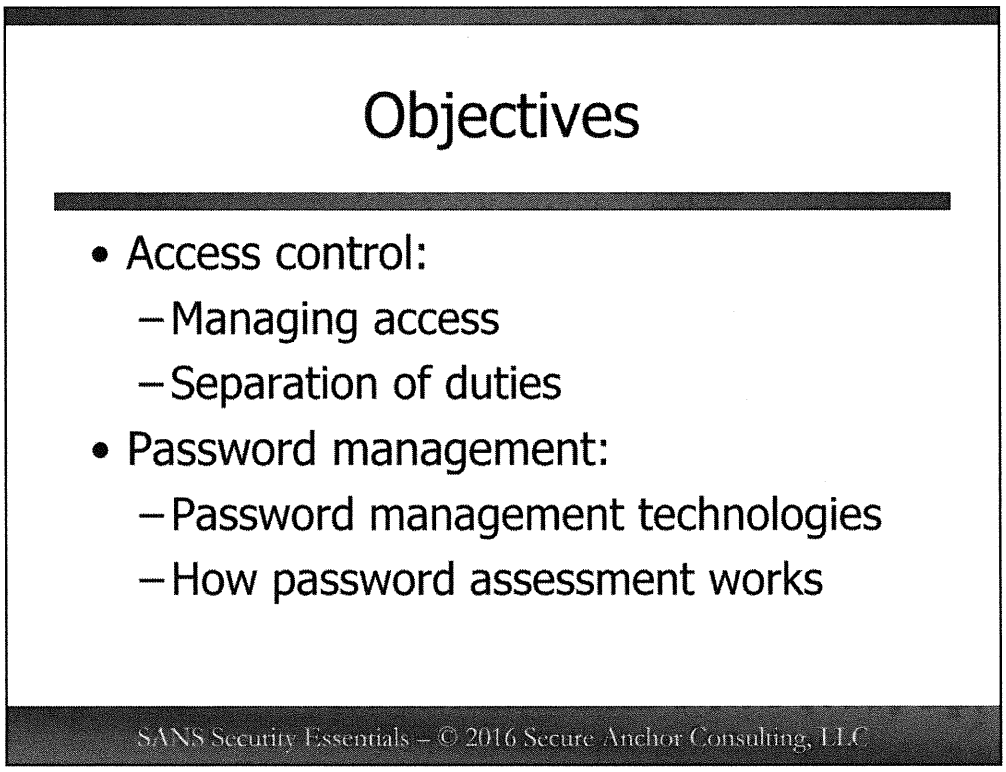
Module 9: Access Control and Password Management

This section intentionally left blank.



Access Control and Password Management

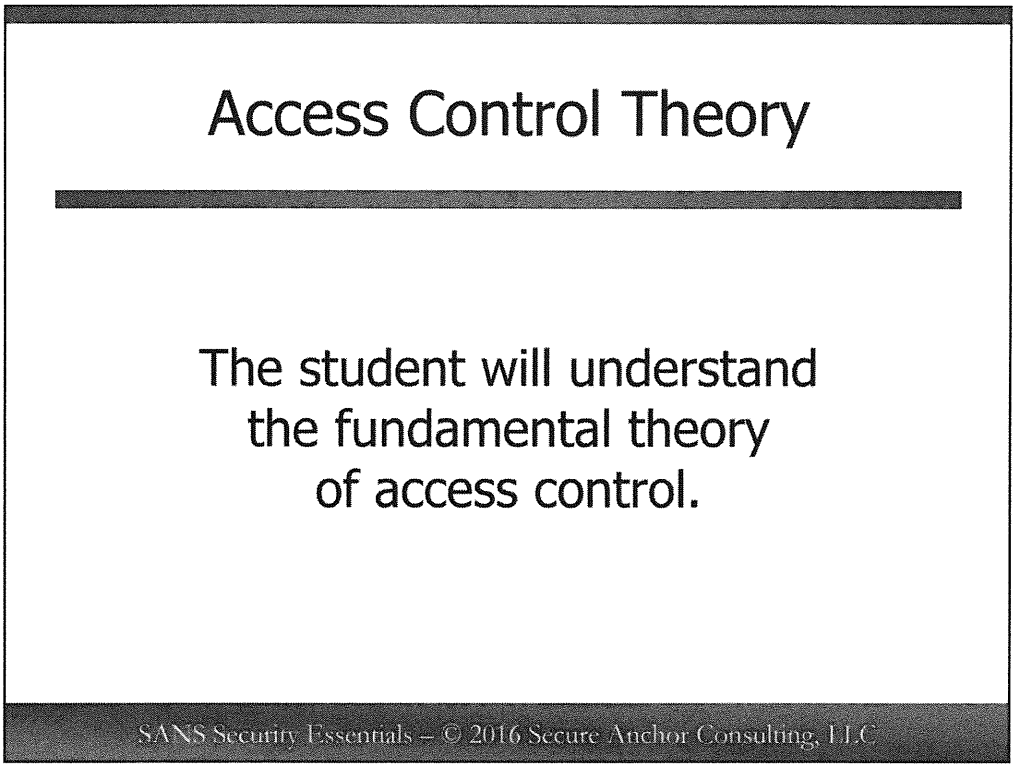
This section intentionally left blank.



Objectives

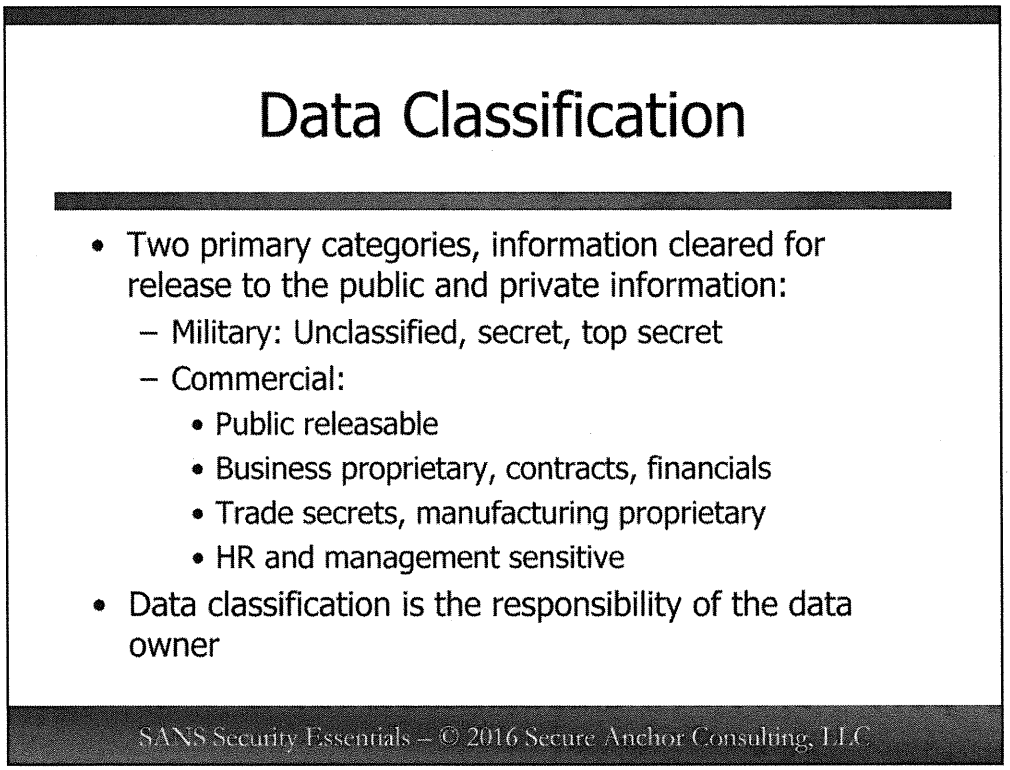
This section discusses the principles of access control. Access control models vary in their approaches to security, and we explore the underlying principles, strengths, and weaknesses of some. A brief discussion regarding authentication and authorization protocols and control will be included.

We also spend considerable time discussing the most common type of access control: the password. We delve into password files, storage, and protection.



Access Control Theory

This section intentionally left blank.



Data Classification

By Sensitivity and By Type

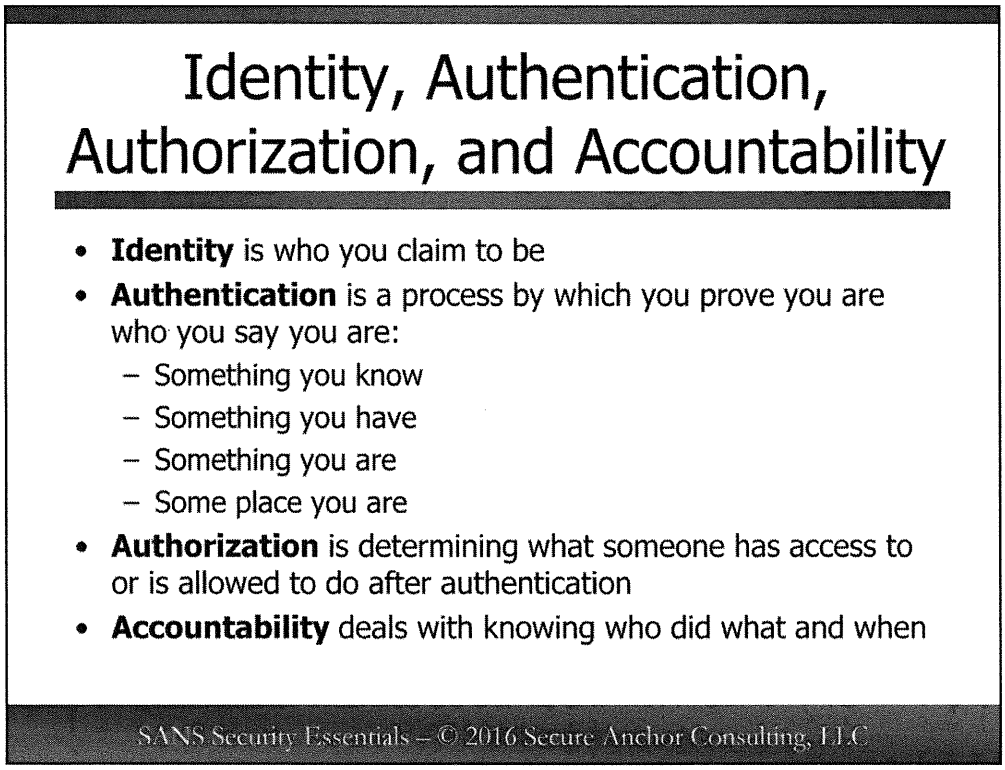
There is information that needs little protection from a confidentiality perspective—information that is considered cleared for public release. All other information needs to be protected, but at what level? The reality is that no organization has sufficient resources to protect all information with the rigor that the most sensitive information requires. Consequently, organizations often classify their data into differing levels so that appropriate protections can be applied based on the sensitivity of the information and on the potential impact of loss. The loss might be in terms of confidentiality (what we usually think of regarding government or corporate secrets), but also could be in terms of integrity or availability.

Governments and their militaries, such as the U.S. Department of Defense (DoD), started the phenomenon of labeling data in order to apply higher levels of protection to sensitive data that could harm their country's national security if it were indiscriminately communicated. Subsequently, this classification practice has become commonplace in the corporate world, as well. A quick listing of the DoD and federal classification levels follows:

- **Top Secret:** The highest levels of protection are given to this data; it is critical to protect.
- **Secret:** This data is important and its release could harm national security.
- **Confidential:** This data is important and could be detrimental to national security if released.
- **Sensitive But Unclassified (SBU):** This generally is information that is sensitive and should not be released, such as social security numbers.
- **Unclassified:** Data owners prefer to keep this information from being released, but the nation would not be harmed if it were.

Generally, the best strategy for classifying data is to use a few clearly delineated categories and train your personnel in distinctive category use. The basic categories of sensitive information are business proprietary, the information about the cost of procuring, profit margins, contact lists, and others. A second category is the information about how products are created. This is often the result of research and development and includes know-how, trade secrets, and an understanding of what will not work and why. A third category is Human Resource and Management proprietary information, which is sensitive because so much personal information is contained in this category. The U.S. government and military require only a few levels of classification, even though they have vast quantities of data to manage. You only need a different category when you have a significant quantity of information that requires significantly different protection.

Who has the authority to classify data and to change data classifications in your organization? Setting the appropriate classification level for data is ultimately the responsibility of senior management. The IT security professional can assist management in making these decisions by using risk assessment techniques to quantify the value of the data and the impact of threats to the confidentiality, integrity, and availability of the data. Access to the data can then be controlled according to the value of the data to the organization.



Identity, Authentication, Authorization, and Accountability

Let's briefly look at access controls, emphasizing the importance of defense-in-depth. To protect critical assets, you have to be able to identify, verify, approve, and track who has access to a given piece of intellectual property (IP).

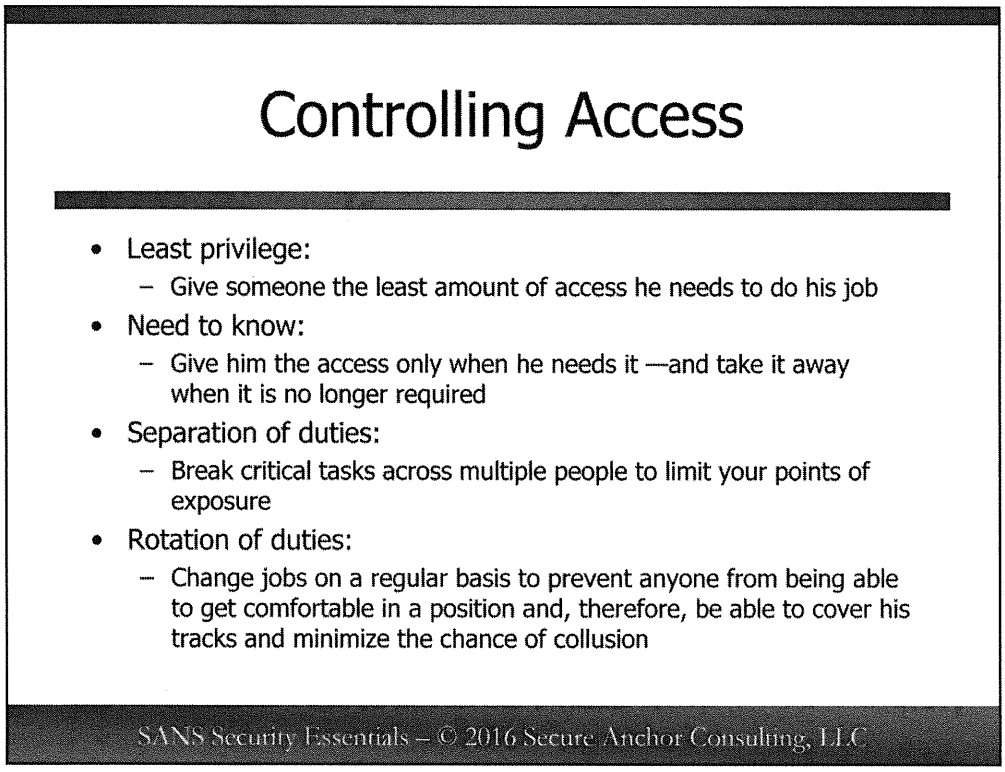
Identification is the process of claiming to be a certain person. Typing in a userID is a form of identification. The problem is anyone could claim she is a given entity, so how do you know that she is who she says she is. This is accomplished through authentication. *Authentication* is proving that you are who you say you are and is done in one of four ways:

- **Something you know:** By remembering a piece of information and presenting it, you can prove that you are who you say you are. The best example of something you know is a password.
- **Something you have:** By possessing something, you can prove that you are a given entity. Token-based schemes in which you carry a token that generates a new password is an example of something you have. If you have the token and can type in the number on the token screen, you can authenticate; otherwise, you cannot.
- **Something you are:** An alternative way to authenticate is by presenting a unique attribute tied to your physical make-up. This is often called *biometrics*. Hand scan, thumb prints, and retina scans are all examples of biometrics.
- **Some place you are:** Global positioning systems (GPSs) can also be used to authenticate that you are in a given geographic area. With sensitive information, you might want to only allow someone to open a document if he is within the walls of a certain five-sided building in Washington, DC.

After you have been properly authenticated, you then have to determine what you are allowed or authorized to do on the system. *Authorization* should be based on a principle of least privilege, where an entity is given only

the minimal access they need to do their job. After access is granted using the principle of least privilege, you want to make sure individuals are held accountable for their actions and you can trace back what occurred on a system through detailed auditing; this process is called *accountability*.

As you can see, all of these measures work together in synergy to properly protect critical assets.

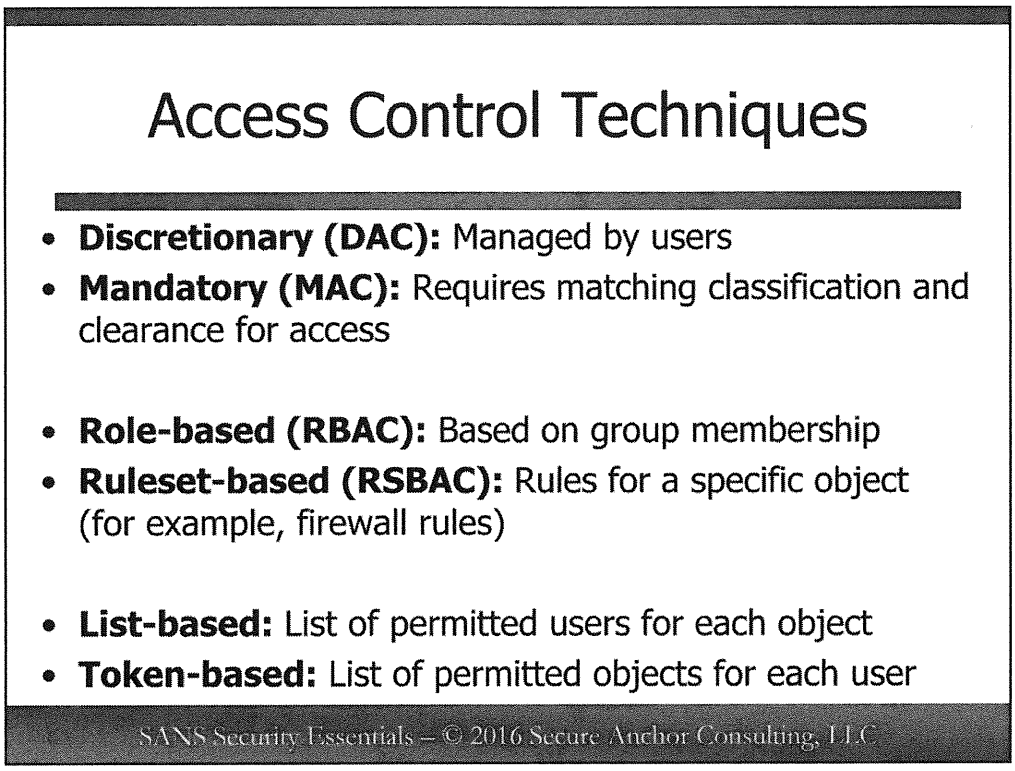


Controlling Access

Now that we have looked at the role that identification, authentication, authorization, and accountability play, we look at some principles associated with access control that you should utilize to make sure your security is as robust as it can possibly be.

In assigning access, you should give someone the least amount of access he needs to do his job. However, this access should not be given all of the time; the access should be granted only when it is needed to perform a job function. For example, if I am the director of HR, the principle of least privilege would say that I need access to every employee's personnel file. On the other hand, the need to know principle would say that you should only give me that access when I have to review a file during a performance assessment—and not all of the time.

With least privilege, we are allowing people to do their job; however, we are only giving them the minimal access needed and no more. In some situations, this works. But, what happens in the case where the minimal access granted is still too great a risk and cannot be taken? In those cases, separation of duties needs to be implemented, where a given task is split between two individuals so no single individual by himself can make a decision. Separation of duties works; but the more people work together, the greater the chance they will collude in order to accomplish a crime. The more people work together, the more the power of separation of duties erodes away, because people build trust. To minimize the chance of this occurring, rotation of duties needs to be performed. This is where people are rotated out of certain jobs at set intervals so the chance of two people colluding is minimized.



Access Control Techniques

Let's start by briefly examining six common types of access control:

- Discretionary access control
- Mandatory access control
- Role-based access control
- Ruleset-based access control
- List-based access control
- Token-based access control

Discretionary access control (DAC) consists of something the user can manage, such as a username or password. For example, a user might choose to give a document password to someone without notifying the administrator. Windows peer-to-peer networks or standard Linux file permissions are good examples of DAC.

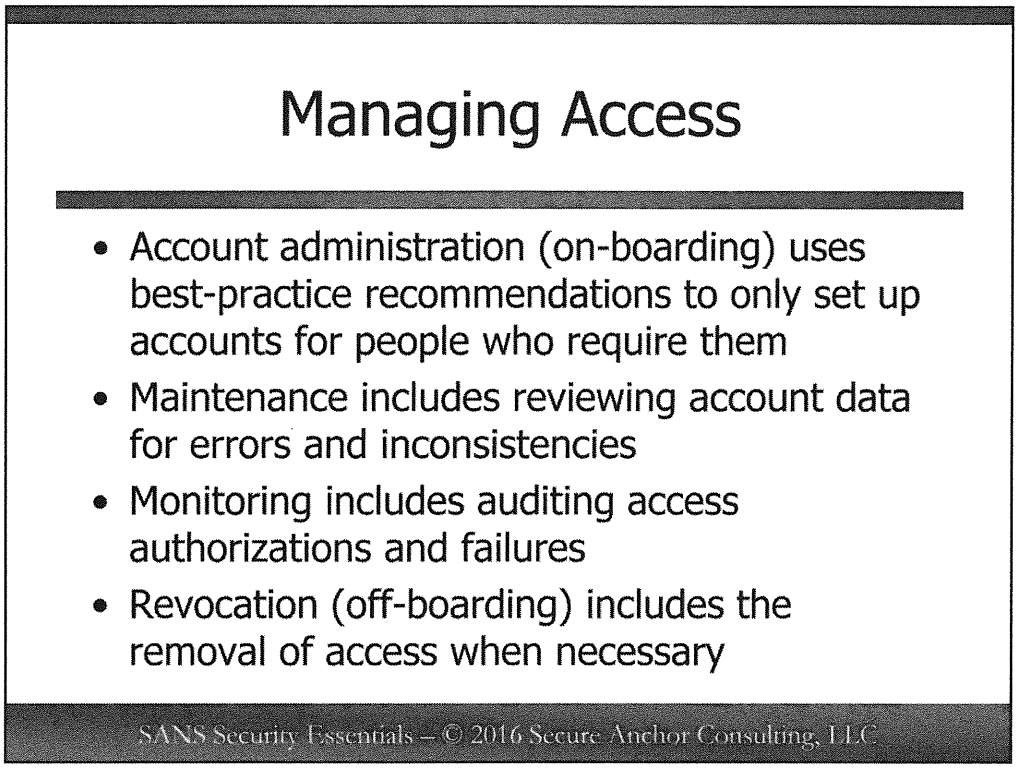
Mandatory access control (MAC) controls all access. Controls are set by the system and cannot be overwritten by the administrator. MAC requires a lot of work to maintain because all data has a classification and all users have a clearance. Users must have the appropriate clearance to access data classified a certain way. Users cannot give their clearance to another person. Security Enhanced Linux (SELinux) originally developed by the National Security Agency and released as an open source project is an example of a system that supports MAC.

Role-based access control (RBAC) assigns users to roles or groups based on their organizational functions. Groups are assigned authorization to perform functions on certain data. Windows Authorization Manager (included with Windows 2003 server) and SELinux include tools to set up RBAC.

Ruleset-based access control (RSBAC) targets actions based on rules for subjects (entities) operating on objects (data or other resources). RSBAC is implemented in a variety of software programs and operating systems (including Linux).

List-based access control associates a list of users and their privileges with each object. Each object has a default set of privileges that applies to unlisted users.

Token-based access control associates a list of objects and their privileges (called capabilities) with each user, the opposite of list-based access control.



Managing Access

User accounts, data, and their relationships must be actively maintained, perhaps by an entire team of employees. This process, called "access management," consists of four tasks: account administration, maintenance, monitoring, and revocation.

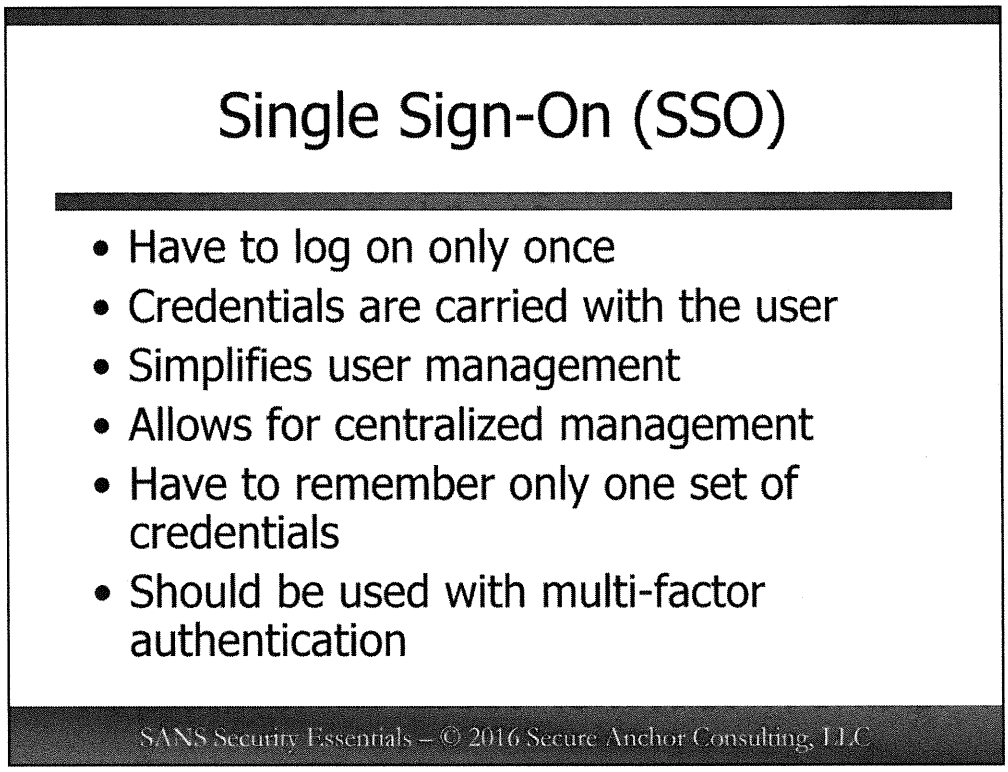
Account administration is a set of best management practices. The administrator verifies the individual before providing access—this is the most important step in the process. This is also an opportunity to teach users not to distribute any access privilege they have (tokens, passwords, and so on)

Maintenance is the process of reviewing account data and spot-checking for inconsistencies or errors. Periodically, account management staff should review and update lists of users and authorizations. Review should take place automatically when employees transfer departments or locations or are assigned new or different duties.

For accountability, authentications and authorizations should be *monitored*. System administrators should log both successful and failed attempts to log on to the system. Logging of the use of systems resources (files, programs, printers, and so on) should be enabled based on risk assessment of the value of those resources.

For instance, successful and unsuccessful access to the payroll database should be logged, but access to a public document store would not necessarily need to be logged.

Almost as important as account administration is *revocation*. Account management staff and system administrators should promptly revoke privileges when they are no longer needed, especially for users who have been fired.



Single Sign-On (SSO)

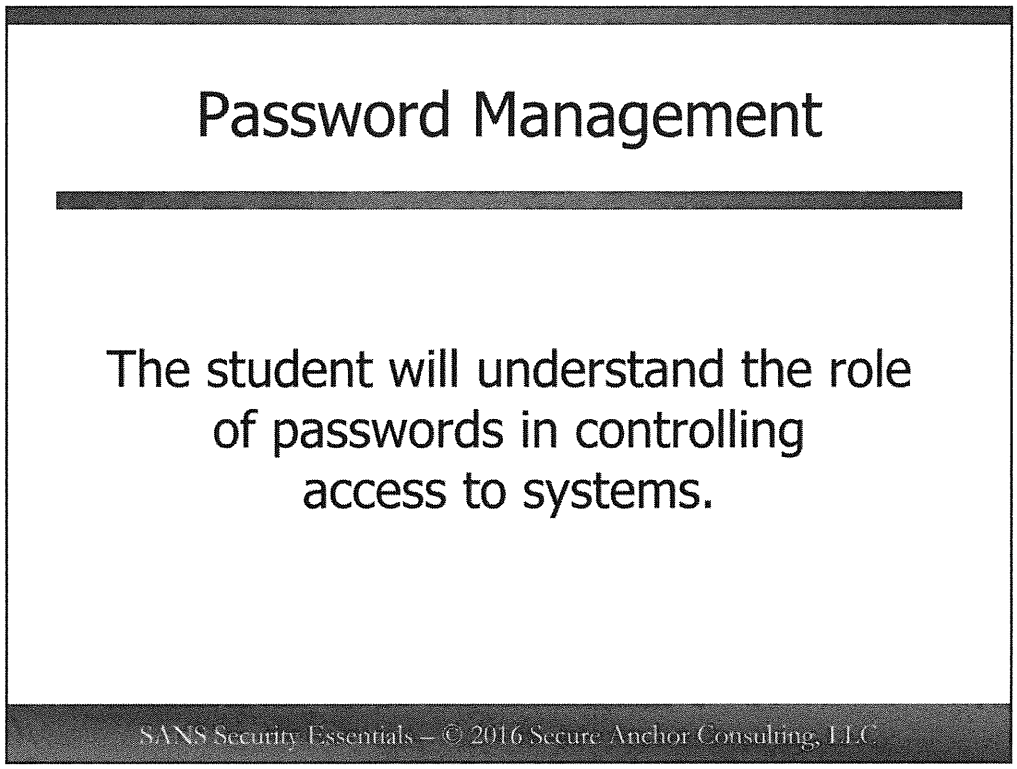
Single Sign-On is a technology that has been very promising for years; however, it seems that we still have to use multiple sets of credentials on a regular basis. The SSO technology allows a user to log on once in the morning and then access any resources for which he has authorization. There is no need for him to repeat the logon procedure over and over again.

There are different ways that Single Sign-On can be implemented. One of the older, and still common, ways is the use of scripts that will mimic the login process between different servers. It is easy to implement but has some very serious security implications, because you often have credentials stored in plain text files.

Another way that SSO is implemented is through the use of a central directory service, such as LDAP or Microsoft Active Directory. This allows the creation of a user account once on a single platform. From that single account, the user will be granted access to different platforms or services.

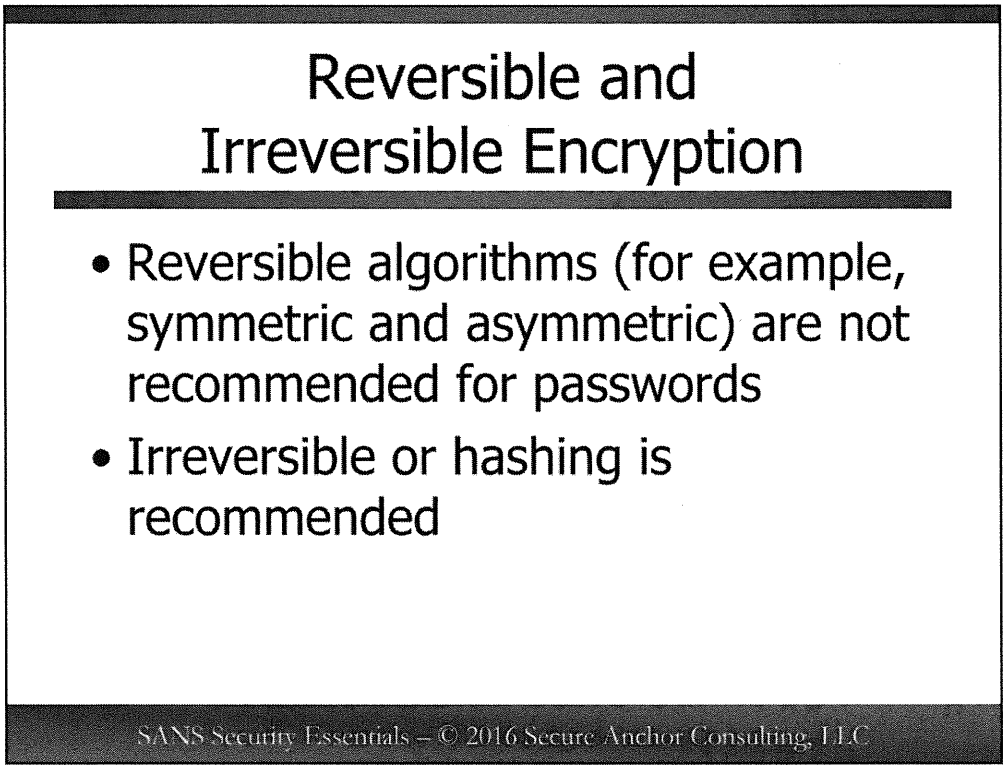
Kerberos, which is part of Windows 2000 and subsequent versions of the Windows Operating System, can also offer the SSO through the use of tickets where credentials are stored. Not all operating systems are kerberized or can make use of Kerberos. It is often necessary to install a third-party software package; and there are some compatibility issues between the different versions of Kerberos.

SSO can save you a great amount of administrative time, but will demand some initial investment in money and human resources. Ensure you select the type of technology that will properly support all of your platforms and legacy applications.



Password Management

This section intentionally left blank.



Reversible and Irreversible Encryption

Reversible encryption is the kind of encryption you use when you need secure storage for confidential information. Because you want to be able to re-access your data, using irreversible encryption is not an option. Microsoft's Encrypted File System (EFS) is a good example of reversible encryption in action.

When reversible encryption is used during the authentication process, generally the system will follow these four steps:

1. Request a username and password from the user.
2. Decrypt the password stored with the username in the password database.
3. Match the decrypted password with the password supplied by the user.
4. Allow or deny access.

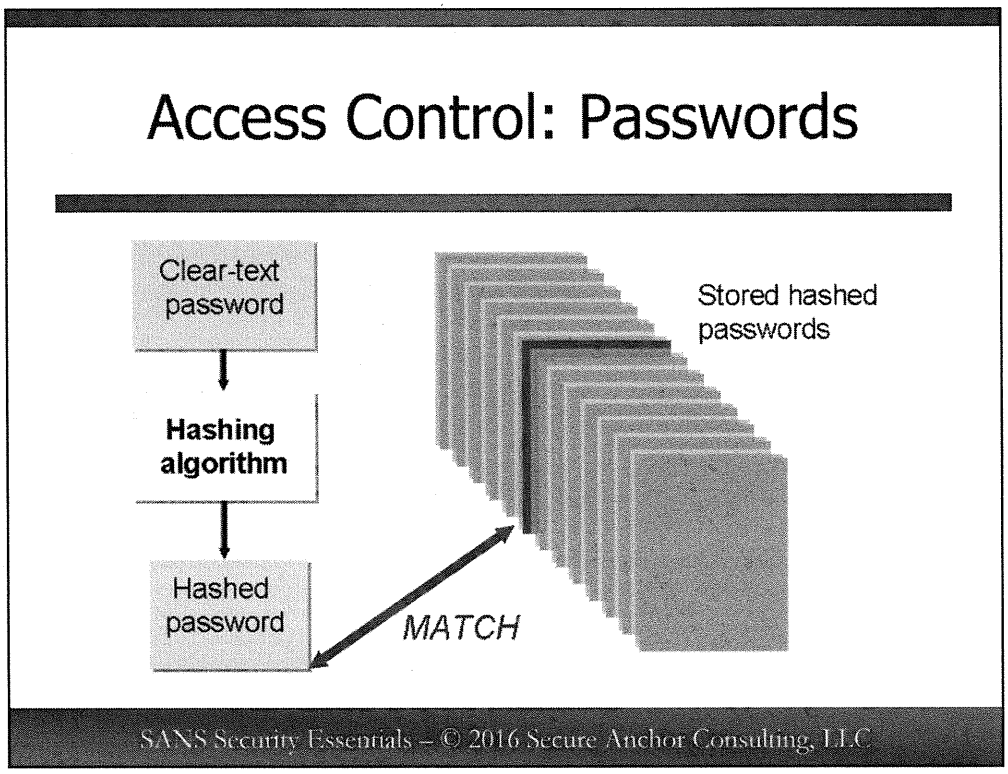
Reversible encryption can be secure for as long as you are able to keep attackers from discovering the key (or algorithm) that is used to encrypt the password. In complex, closed-source operating systems or applications, it can be very difficult for an attacker to discover where exactly in the system this encryption key has been stored. Therefore, searching through binary files is generally not the approach taken to discover the key.

In step 2 of the authentication process, you can see that the system accesses the encryption key in order to decrypt the password stored in the password database. By attaching special debugging software to the authentication process, an attacker can follow each step this process takes and look at all the variables used to complete the process, including the variable used to store the encryption key.

After this key or algorithm has been detected, it can generally be used to decrypt the entire user database. These steps are also referred to as "reverse engineering." They are not only popular for reversing encrypted passwords, but are also commonly used for detecting algorithms used to generate, for instance, software license keys.

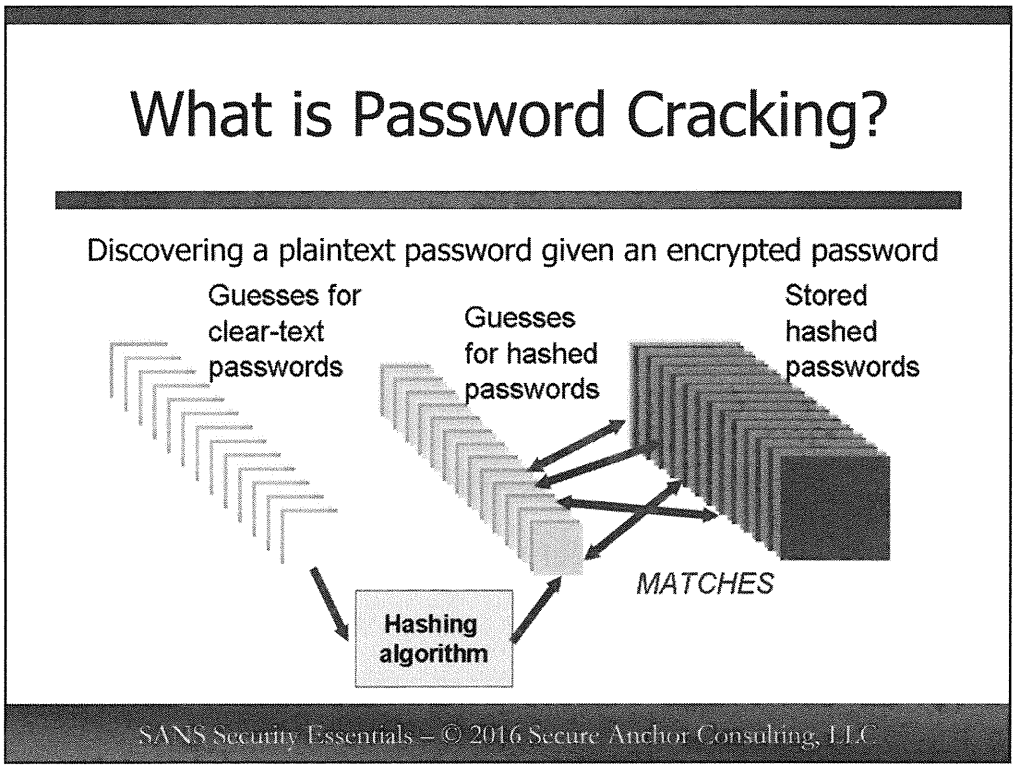
The ease with which skilled attackers will be able to detect encryption keys or algorithms makes reversible encryption an insecure solution for storing passwords. It does help create a barrier against shoulder surfing passwords from printed configuration files on the desk of an administrator, but makes a poor solution when compared to irreversible encryption.

Irreversible encryption is known by a lot of different names, including one-way encryption, one-way hashing, and simple hashing. We continue to refer to this type of password storage as "hashing."



Access Control: Passwords

This slide shows the process is which passwords are stored on the system and how a user is authenticated using hashes.



What is Password Cracking?

Password cracking is an offline process of attempting to guess passwords, given password file information. This section begins with a discussion of what password cracking is, why it is important, and what methods are available.

Let's back up for a moment and think about why passwords are so important. Often, passwords are the first line of defense against interactive attacks on a system. Because it is fairly easy for someone to figure out a user ID, the only thing protecting that user's account is her password. If an attacker can gather no helpful information to aid in the attack (such as password file contents or sniffed network traffic), he must resort to either creative or brute-force password guessing.

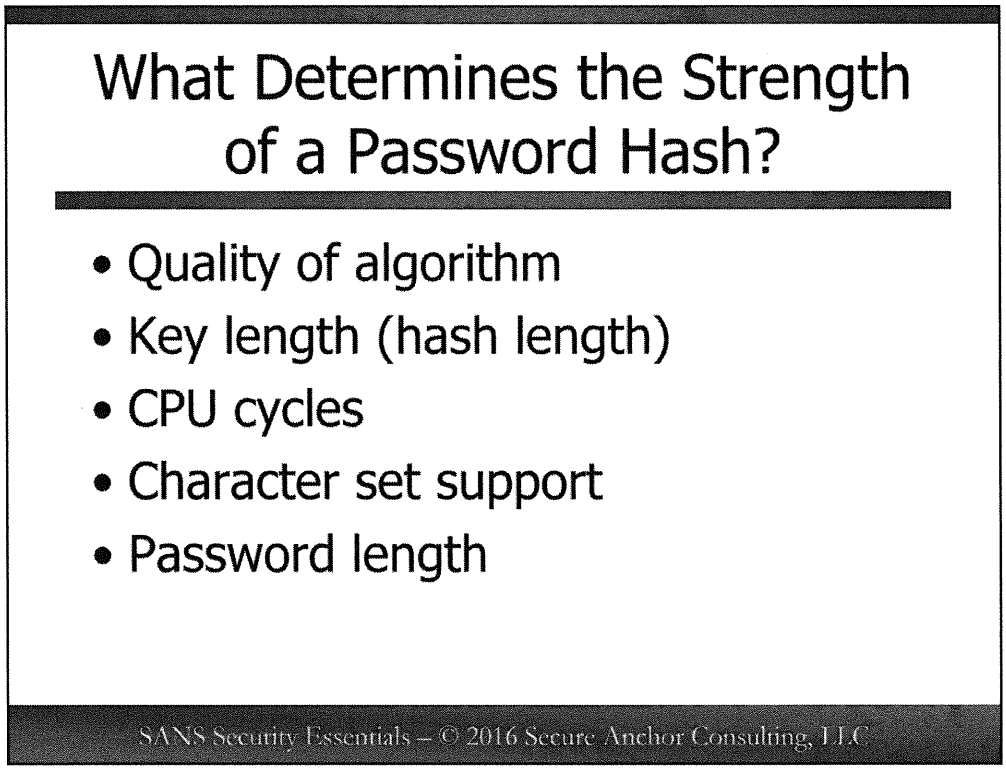
If an attacker can at least read the password file or obtain a copy, his chances of successfully obtaining an actual password increase significantly. Even if the attacker obtains only a lowly user-level password, it's fair to assume he will log in to the target system as the user and then attempt to break into the root account via local vulnerabilities.

Password Storage

In many companies, passwords are more than just the first line of defense—they're the only security measure protecting servers and internal information. Because most user IDs consist of an employee's first initial and last name (or something similar), it's fairly easy to discover valid user IDs for individuals at a company. Then the only other piece of information needed to gain access is a user password. So passwords must be protected, and they must be very difficult to guess.

Unauthorized disclosure, unauthorized modification, and unauthorized removal are all threats to password integrity. If users disclose their passwords (intentionally or not) by writing them down or sharing them with other people, malicious parties might obtain them. It's even worse if attackers can modify the password data directly because they could change passwords without needing to know the originals. Of course, changing a password is risky for an attacker—users tend to get suspicious when their passwords suddenly stop working.

Operating systems protect passwords by using strong cryptography to hide the original content. Even if the encrypted password is revealed, it is difficult to determine the original.



What Determines the Strength of a Password Hash?

The strength of a hash used for password storage primarily depends on five factors:

1. Quality of Algorithm

A hashing method is only as good as its algorithm. Even an infinite key length will not protect your password if the algorithm used to produce the key has been proven to be an insufficient protection mechanism. In general, however, it is very difficult for cryptographers to define the quality of an algorithm; many algorithms are believed to be reasonably secure until proven otherwise. It is recommended to use an algorithm that has been available for a long time and well tested.

2. Key Length

A larger hashing key means that there are more possible keys to uniquely identify the input of the hash function and, therefore, a smaller chance of collisions. Although this is considered the most important feature of hashes with respect to data integrity, the limited length of passwords employed by users makes key length only a minor contributor when considering the strength of password hashes. Related to password hashes, the key length of a hash could be considered a real issue if it is not able to uniquely identify all possible passwords within the minimum password length and complexity requirements that have been determined in the company's password policy.

3. CPU Cycles Used to Calculate the Hash

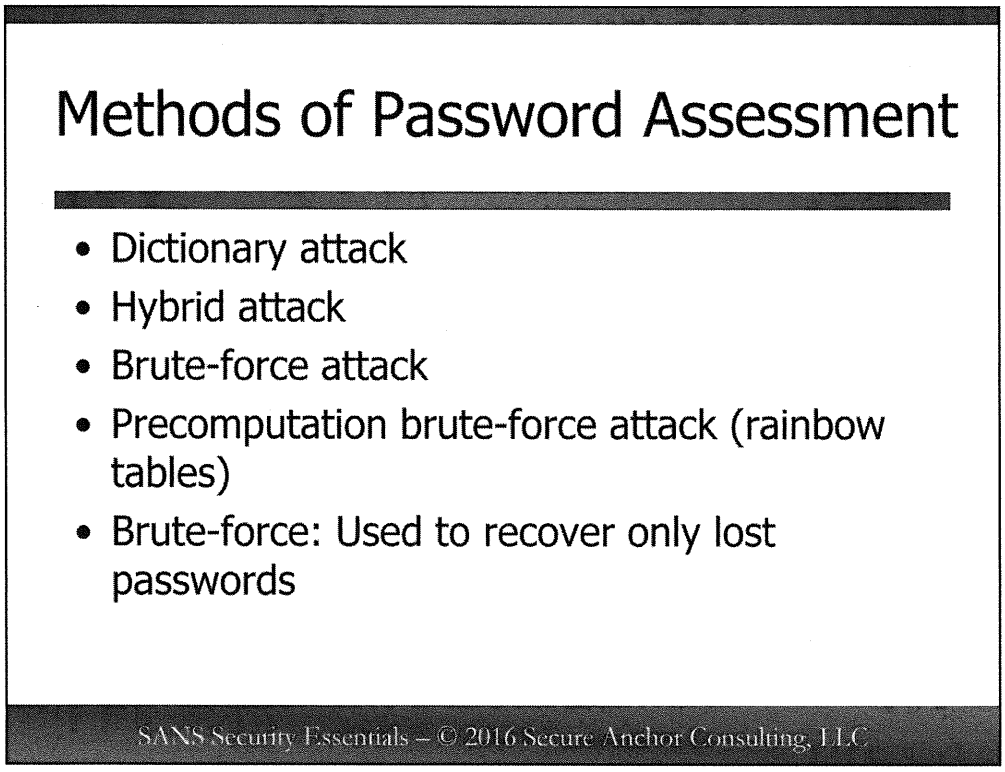
Because hashes cannot be reversed, an attacker must hash possible passwords and compare these hashes with the obtained password hash of an existing user in order to determine the credentials that can be used to log in. The number of password tries per second an attacker is able to launch against a hash depends on the number of CPU cycles the hashing function uses to compute the hash of a password. The more tries per second, the sooner all possible combinations can be tested.

4. Character Set Support

The strength of a chosen password depends heavily on the length of the chosen password and the character set used when choosing the password. A 7-character password made out of a maximum of 26 characters results in 8 billion possible passwords (which is generally easy to brute-force within hours). When a 52-character character set can be used, it will take 128 times longer to brute-force.

5. Password Length

Because most users do not like to be forced into using characters that need <ALT> keys to access, the effective password character space is generally limited to alphanumeric characters and punctuation marks. This limited character space makes the length of a password important. Through the length of the password, the possible password combinations can be increased and brute-forcing can be made more difficult.



Methods of Password Assessment

Many of us remember how, in the movie *WarGames*, a teenager breaks into the government's super-secret WOPR computer by guessing the username and password of the scientist who created the WOPR's software. The teen researched information publicly available about the scientist and guessed that the man's password was the name of his young son, Joshua. That familiar example illustrates exactly why it is important not to use words or names that might be associated with a person. Such information might be readily available to an attacker, who could use it to make educated guesses and eventually come up with the right password, even if he does not know the user.

Most of us are aware that we shouldn't use passwords that are too short (because all the possible character combinations could easily be tried) or write passwords on sticky notes and put them under our keyboards. Beyond this basic understanding, however, can we quantify what makes a password difficult to guess when a computer is used as the guessing engine? The answer is: It depends on the particular method used to protect the sensitive information.

Password-Cracking Techniques

Computers use one-way hashing algorithms to encrypt passwords for storage. A one-way hash is mathematically easy to compute in one direction (for encryption), but impossible to compute the other way, even for computers. This is important because someone who recovers a password file can't use the hashed values to reverse the one-way encryption function and recover the original passwords. But how, then, does the computer use the encrypted information to authenticate users?

The technique is simple. Even though hashing functions can't be reversed, they always can produce the same output given the same input. Thus, the computer stores only the hashed passwords (rather than original passwords) on disk. When a user attempts to authenticate to either the machine itself or the network, the computer applies the hash algorithm to the password the user has supplied for authentication. If the hash of the user-supplied password matches the hash stored on disk, the password is correct, and the user is successfully authenticated.

Password cracking is the process of trying to guess or determine plaintext passwords, given only encrypted passwords. The process does not actually break the encryption; it mimics the actions that would take place if a user tried passwords

until guessing the right one. Each guess is hashed and compared to the stored value. When a match is found, the user is authenticated. Cracking usually is done using a computer to generate many guesses and compare each against the hashed password value until a match is found. Usually, the cracking operation is performed offline against a recovered password file.

The general method for cracking is the following:

- Find a valid user ID.
- Find the encryption algorithm used.
- Obtain the encrypted password.
- Create a list of possible passwords.
- Encrypt each password in the list.
- Determine whether there is a match.

There are four general methods for cracking passwords. The main difference among the alternatives is the speed of performing the crack versus the complexity of passwords that can be cracked. For example, one method that is extremely quick will crack only passwords with a low complexity, such as passwords that contain only letters. More complex passwords might contain letters, numerals, and special characters and, therefore, will take longer to crack.

The Dictionary Attack
The fastest method for cracking passwords is a dictionary attack, testing all the words in a dictionary or word file against the password hashes. When a dictionary attack program finds the correct password, it displays the result. There are many web sites that have downloadable dictionaries you can use. These attacks are quite effective because people tend to choose dictionary words for passwords.

Common countermeasures against the dictionary attack are policies and filters. A policy is a written set of rules adopted by an organization to encourage users to choose stronger passwords. For example, an organization might insist that users choose passwords that aren't dictionary words.

As you might guess, a policy doesn't help much if it's not enforceable—that's where filters come in. A filter is a technical mechanism that forces users to choose passwords that adhere to some standard set by the organization. Filters are usually deployed by replacing the program used to change passwords to one that allows only passwords that satisfy all criteria.

The Hybrid Attack
The hybrid attack builds upon the dictionary method by adding numerals and symbols to dictionary words. Many users choose passwords such as bogus11 or he11o!! (in which the ells are replaced by ones) just to satisfy policies and filters. These passwords are just dictionary words slightly modified with additional numerals and symbols. The hybrid attack rapidly generates these passwords and computes their hashes. As a result, this type of password is still easily crackable, even though it will pass through many password filters and policies. Several hybrid attack tools have configurable rule sets that allow the attacker to specify combinations and permutations of dictionary words to try. These tools are surprisingly effective and easy to use.

The Brute-Force Attack
The most powerful cracking method is the brute-force method. It will always recover the password no matter how complex it is—it's just a matter of time. Very complex passwords that contain characters not directly available on the keyboard might take so much time that trying to crack them is not feasible on a single machine using today's hardware; but most complex passwords can be cracked in a matter of days. This is usually much shorter than the password expiration time used by most site administrators. Using a real-world cracking tool is the only good metric for setting password expiration times.

The Pre-Computation Attack
As we previously explained, the strength of a password hash largely depends on the time it takes to generate a certain hash. Efficient password cracking is not possible if it takes a lot of CPU time to determine the hash that goes with a certain password. By pre-computing hashes of possible passwords and storing the results in a database or table, the CPU time can be invested at times when there is plenty of processing power available. By the time the real cracking needs to be done, matching hashes with passwords is only a matter of searching through the pre-computed tables, which requires more memory, but takes only a fraction of the time needed to brute-force a hash. The files containing the pre-computed password hash values are called "rainbow tables."

Cracking Motivation
There are many reasons for attempting to discover users' passwords. Impersonating users is the most obvious reason, which would be useful to attackers and support personnel alike. Another reason is that system administrators might want to recover forgotten passwords or migrate from platform to platform (for example, Windows to Unix). With that said, the paramount reason for cracking passwords is for the system administrator to audit the strength of each user's password. There are password filters for some OSs; but how do you know whether or not these filters are effective? Without testing the passwords generated by users against a real-world password cracker, you are guessing at how much time it would take an external attacker or malicious insider to obtain those passwords and are, therefore, guessing at how significant the application of a major layer of defense is.

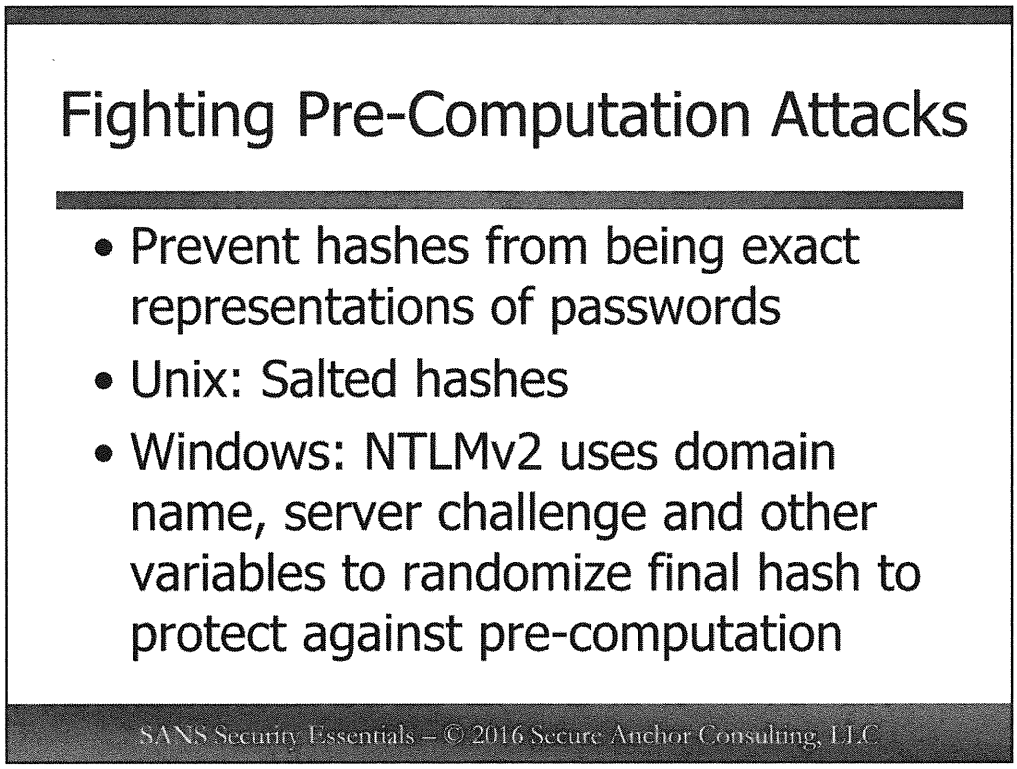
Another possible reason why an administrator might want to crack users' passwords is to aid in migrating users from one platform to another. This practice is not recommended because it usually results in a security violation. For example, suppose a company wants to migrate two AD domains into one. Instead of moving all of the accounts in one domain to the other, giving those users the same temporary password, and having them all log on and change their passwords, a company might want to make the process transparent. One way is to crack all the passwords in one domain and then manually add each new account to the new domain with its old password. Then the users can log on with their old passwords and not even know their accounts have been migrated. The downside is that the technical staff doing the migration now knows the users' passwords. As you can imagine, problems could result.

Some might argue that any cracking of users' passwords is a potential security violation because once the password is cracked, someone other than the user will know that user's password. To avoid this problem, an organization can develop a customized cracking scheme in which strong passwords are never cracked. For example, if a company's written password policy says that all passwords must contain letters, numerals, and special characters, the password cracker can be configured to crack passwords that only contain:

- Letters
- Numerals
- Special characters
- Letters and numerals
- Letters and special characters
- Numerals and special characters

If the password adheres to the policy, which in this case means it contains letters, numerals, and special characters, it will not be cracked. How? Remember that cracking programs build up lists of words to encrypt and compare them against the ciphertext in the password file. If we instruct our password program not to add a certain type of potential password to its list, that type of password will never be cracked.

Only passwords that do not follow the policy will be cracked and seen by the administrator. Users with cracked passwords could then be instructed to change their passwords immediately, and the administrator would not see the new values.



Fighting Pre-Computation Attacks

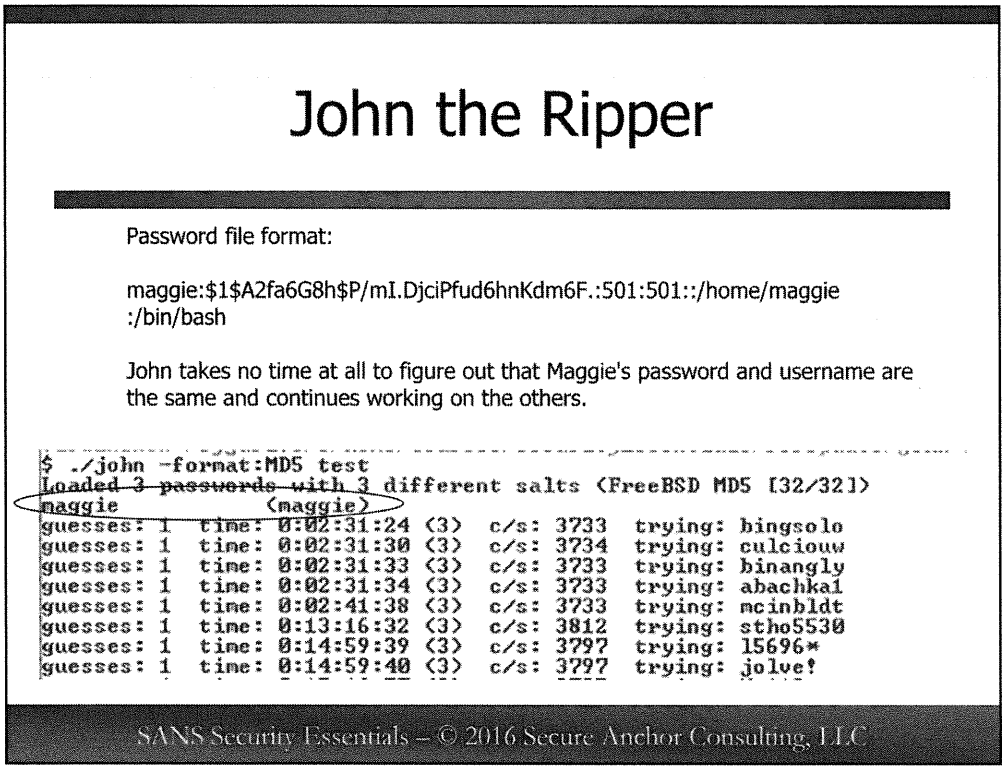
Pre-computation is only successful if you are certain that the hashes in the pre-computed tables will match the hashes of passwords discovered during future audits. You can be certain this is the case when a hash is an exact representation of a password.

For example: "B1BAAD710F034F2E6801C1C509D0D7A1" is the exact NTLM representation of the password "MHrtsk3r". If both this password and the hash are included in a rainbow table, we can be certain that we are able to crack this password on all Windows systems.

Attackers, who are able to sniff a NTLM string off the network, could efficiently attack this string by matching it with the entries in the rainbow table. The only way to keep attackers from cracking these strings using efficient approaches, such as pre-computed rainbow tables, is to make password strings domain, machine, user, or even session dependant. Microsoft acknowledged this security issue and introduced NTLMv2.

With NTLMv2, the hash sent over a network is not a unique representation of a user's password. Instead, the hash is a function of username, domain name, client challenge, server challenge, and the NTLM hash of the password. Because there is no one representation of a password, it is impossible to pre-compute generically useable tables.

Modern Unix systems follow a similar approach, by storing passwords using a salted MD5 hash.



John the Ripper

John the Ripper is powerful and fast. John combines several modes into one program and is fully configurable.

Traditionally, Unix passwords were encrypted with the standard DES crypt() function. As computer processors became more powerful, passwords became easier to crack; and as the Internet emerged, tools for distributing the task of password cracking across multiple machines became available. As of about 1999, consumer computers were capable of computing the DES crypt() function fast enough to make the algorithm effectively unusable for password protection.

Part of the speedup in cracking the DES passwords came from a technique known as "bit-slicing," proposed by Eli Biham in 1997. An initial implementation of this software technique encrypted at an average rate of 137 megabytes per second, as compared to the fastest method used by Crack (Eric Young's libdes), which operates at 28 megabytes per second.

To better protect passwords, many operating systems (including several GNU/Linux flavors, OpenBSD, and FreeBSD) chose to employ stronger hashing and encryption algorithms, such as MD5 and Blowfish as alternatives to DES. Also, salts were added to randomize the passwords and make them harder to perform traditional attacks.

Although a version of crypt() that uses MD5 has been developed and can be used by Crack, John the Ripper is much faster for auditing passwords encrypted with these alternate algorithms. John's MD5 cracking routine uses a simplified central computing loop that decreases the cost of each hash computation. John also can use the DES bit-slicing technique, so it is much faster than Crack, even when used against DES-encrypted passwords.

> More information on attacks against DES and salt functions with an interesting chart depicting the number of encryptions per second with the advancement of processors is available at http://www.usenix.org/events/usenix99/provos/provos_html/node13.html.

John has built-in support for the following types of passwords:

- Standard and double-length DES
- BSDI's extended DES
- MD5
- Blowfish
- Andrew File System (AFS) passwords
- NTLM passwords

If your cipher algorithm isn't listed in the previous list, you can extend John to support it, thanks to its modular design. John itself takes advantage of this architecture to bring you optimized modules for different architectures, some of which use assembly routines for greater speed.

Cracking Modes
John has several modes, each of which goes about cracking passwords in a different way. To use any of the supported algorithms, you'll have three modes from which to choose for your audit. The mode you choose depends on the strength of the passwords themselves and how much time and processor power you have. A fourth mode, external mode, lets you use external modules for algorithms that aren't directly supported. These modes can be selected on the command line or by editing the john.ini file. This module does not cover advanced configuration of John the Ripper, but take a look at the example john.ini that comes with the distribution for ideas on how to get the most out of the software. The four modes are described in detail next.

Wordlist Mode
Wordlist is the simplest of the crack modes offered by John the Ripper. John does not do any type of sorting of the wordlists, allowing you to further optimize its performance by putting the most common words at the beginning of the list. Like Crack, John will perform substitutions and other transformations on each word if configured to do so in the john.ini file.

Single Crack Mode
Single crack mode uses the username and GECOS information to guess passwords. John also adds previously guessed passwords to the list, helping to detect users with the same password on several accounts. As with wordlist mode, John will transform each guess as configured in john.ini. Because it doesn't take long to try this short list of guesses, single crack mode is much faster than wordlist mode and should be used first.

Incremental Mode
Of the three standard modes, incremental is the most powerful, but also the most time-consuming. It tries every combination of letters, numbers, case, and special characters. Because it tries every combination of every possible length, it runs indefinitely.

External Mode
Though John is powerful enough for most purposes on its own, it can be extended to include whatever custom routines you can define. These routines are developed using a subset of C compiled at runtime.

Cracking a Red Hat Password File
In addition to being able to crack standard Unix password and shadow files, John can also crack alternative password authentication systems, such as those used by Red Hat and other Linux and BSD Unix systems. Instead of using DES to encrypt and store passwords, Red Hat Linux used the MD5 algorithm to store passwords with a salt value. This alternate encryption mechanism will stop attackers from using Crack against user passwords, but John has been extended to support this functionality.

John has no problem handling the MD5 format. Notice in the following example, Maggie's encrypted password starts with a "1" indicating that it has been hashed with MD5. The portion of the string following the "1" but before the "$" (A2fa6G8h) is Maggie's salt value. The rest is the encrypted password itself (in this case, "maggie"). John takes virtually no time to crack this simple password (in this case, the clear text is maggie).

maggie:1A2fa6G8h$P/mI.DjciPfud6hnKdm6F.:501:501::/home/maggie:/bin/bash

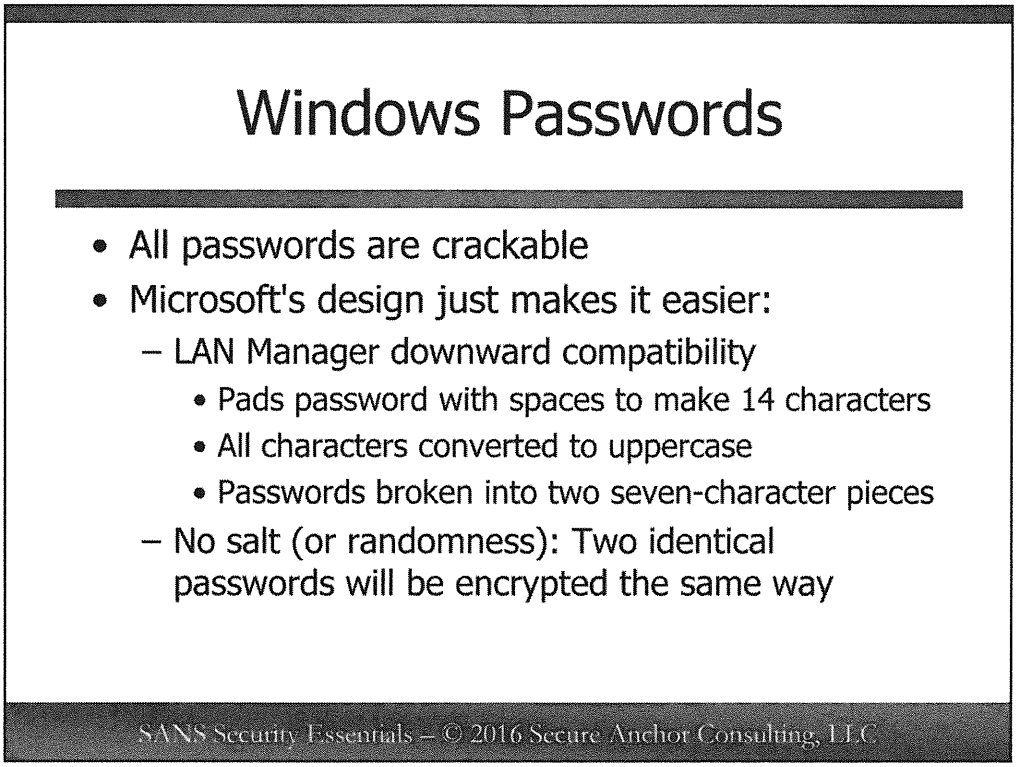
Suppose this entry and two others are in the file test. When we run:

john -format:MD5 -w:password.lst test

we see that John takes no time at all to figure out that Maggie's password and username are the same and continues working on the others:

```
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
maggie          (maggie)
guesses: 1  time: 0:02:31:24 (3)  c/s: 3733  trying: bingsolo
guesses: 1  time: 0:02:31:30 (3)  c/s: 3734  trying: culciouw
guesses: 1  time: 0:02:31:33 (3)  c/s: 3733  trying: binangly
guesses: 1  time: 0:02:31:34 (3)  c/s: 3733  trying: abachka1
guesses: 1  time: 0:02:41:38 (3)  c/s: 3733  trying: mcinbldt
guesses: 1  time: 0:13:16:32 (3)  c/s: 3812  trying: stho5530
guesses: 1  time: 0:14:59:39 (3)  c/s: 3797  trying: l5696*
guesses: 1  time: 0:14:59:40 (3)  c/s: 3797  trying: jolve!
```

As John makes its way through the wordlist, it keeps you apprised of its progress. After each word it tries (with all the associated substitutions and combinations configured in john.ini), it tells you how long it took and the number of combinations it tried per second.



Windows Passwords

As we mentioned previously, computers store passwords as one-way cryptographic hashes. Passwords are sensitive data—they can be used to impersonate any user, including the administrator. It wouldn't be wise to store them in the clear.

The Windows operating system is no exception. Windows uses its own hashing algorithm to store passwords. The particular algorithm older versions of Windows (for example, 95 and 98) uses, however, has weaknesses that make it easier to derive the original cleartext passwords from the hashes. Although this was created for older versions, the current version of Windows has the feature turned on (by default) for downward compatibility reasons.

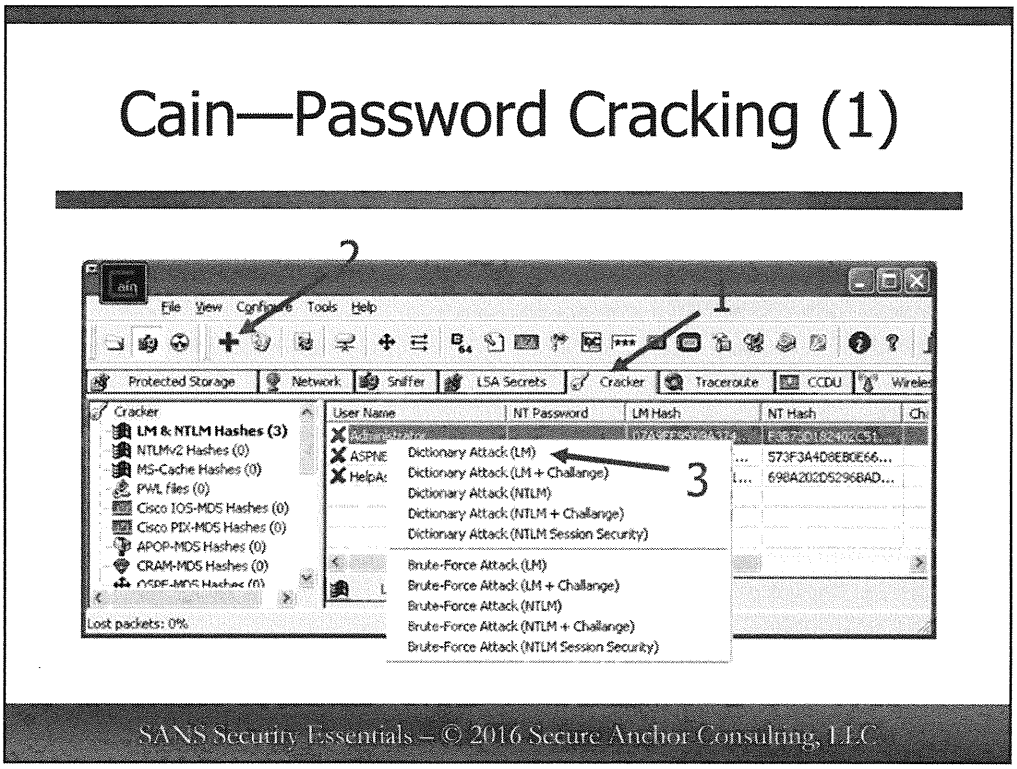
All passwords are crackable using brute-force techniques. The question is, how long does it take? The goal of encryption is to make brute-force attacks take so long as to be unfeasible. Either the attack takes more time than the attacker cares to spend, or it takes so long that the information is no longer useful when it is finally recovered.

Three major design flaws in Windows NT and Windows 2000 allow passwords to be cracked very quickly. This scheme severely weakens every password by breaking it into two seven-character words before applying the hash algorithm. Instead of trying to crack a password that is 10 or 12 characters long, an attacker has to crack only two seven-character (or fewer) passwords.

Because Windows NT breaks up the password, and most people who use numbers or special characters in passwords put them at the end, it's easy to crack even those passwords that seem complex. For example, to crack password#7, you have to crack "passwor" (which is fairly easy) and "d#7" (which is harder, but still easy because it's so short).

Another flaw in the LAN Manager scheme is that it automatically converts all lowercase characters in your password to uppercase, reducing the number of passwords an attacker has to try by almost a factor of ten! Without password filters or policies in place, users are likely to choose passwords containing nothing but letters. The effects of this design flaw become even worse under that circumstance.

The third issue is that LAN Manager does not use salts. Without salts, two users with the same passwords will have the same ciphertext, resulting in a tremendous cost savings for attackers. Now they only have to encrypt each word once. If any user has that password, there will be a match. If salts were used, an attacker would have to find out the salt for the user and then encrypt all possible passwords with that salt to see whether there was a match. After a match was found, the attacker would have to move on to the next user and do the same thing. As you can see, cracking salted passwords can take much longer.



Cain—Password Cracking (1)

Cain is a wonderfully (or dangerously!) powerful password-auditing and cracking tool for Windows systems. An amalgamation of several other password-cracking tools, Cain includes a tremendous amount of functionality, capable of cracking many different types of passwords using a variety of techniques. Written by Massimiliano Montoro, Cain is free to download from http://www.oxid.it. Note that the source code to Cain is not publicly released.

Cain is one of the best Windows password-cracking programs on the market for several reasons:

- It is easy to use and has a nice graphical user interface.
- It takes advantage of the weak LAN Manager scheme and can crack passwords extremely quickly.
- It has the ability to extract password hashes using a process called "DLL injection." Windows password hashes are otherwise inaccessible to users, even the administrator, when the operating system is up and running.
- It provides the option to sniff a challenge/response dialogue from the network and use that information to crack the password.
- Current versions can even get around Microsoft's SYSKEY protection mechanism.
- It is free to download and use.

SYSKEY, enabled by default on Windows 2000 and introduced in Windows NT Service Pack 3, encrypts the password hashes with a 128-bit key that is usually stored on the local hard disk. The key can also be password-protected or stored on removable media.

Using Cain

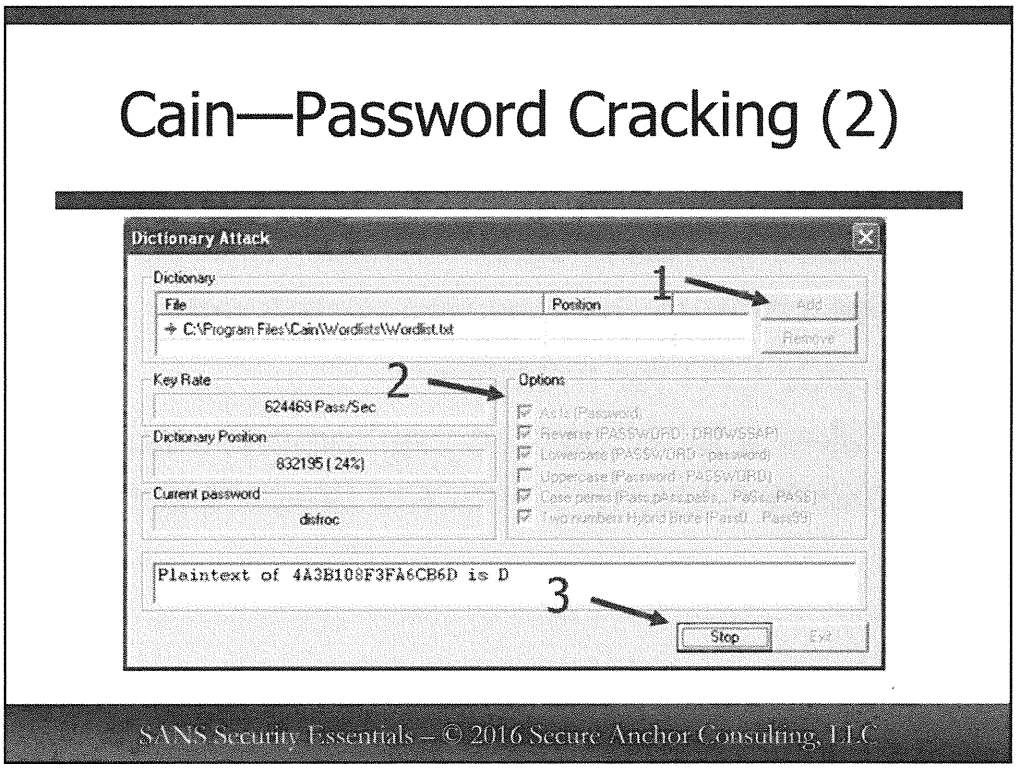
After downloading and installing Cain by following the installation wizard instructions, simply run Cain on the system you want to use for cracking passwords. Collecting the NTLM or LANMAN hashes for the local system to attack is a simple three-step process, as shown in this slide:

1. Click the *Cracker* tab.
2. On the toolbar, click the blue "+". This opens the "Add NT Hashes from" window, prompting you to collect hashes from the local system, a text file, a Security Accounts Manager (SAM) file, or from the local SAM database. Upon completing the wizard, Cain identifies all the username and password hash information it was able to collect.
3. Right-click the account you want to crack and select the password-cracking method.

Cain supports multiple password-cracking techniques:

- **Dictionary attack:** This enumerates each word in a supplied dictionary as a potential password. Cain implements a hybrid approach to the dictionary attack, allowing the user to try each dictionary word as-is ("PASSWORD"), reversed ("DROWSSAP"), lowercase ("password"), multiple-case permutations ("PaSsWoRd," "pAsSwOrD," and so on) and appending a two-digit value to the end of the word ("password00," "password01," and so on).
- **Brute-force attack:** This allows the user to select the desired character set to brute-force (all alpha, alpha and numbers, alpha and special characters, and so on) and exhaustively enumerates all possible character combinations. This attack is successful only when sufficient time is given to Cain to enumerate all password possibilities; a sufficiently complex password might require more time than a single lifetime can manage to crack the password using current hardware.
- **Cryptanalysis attack:** This uses a time/memory tradeoff optimization technique with pre-computed tables of possible passwords. We examine this option in more depth later in this module.

Let's examine the use of Cain when we select the dictionary attack option.



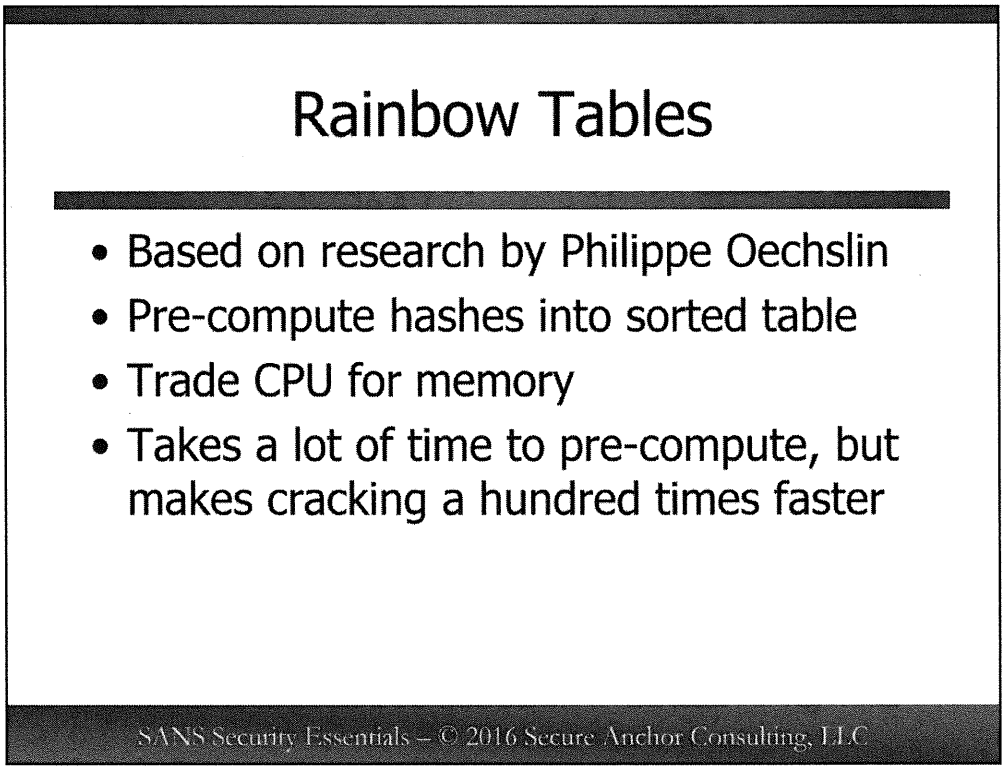
Cain—Password Cracking (2)

This slide is an example of Cain's dictionary attack mode in progress. Cain comes with a simple default dictionary wordlist of 306,000 entries that is useful for simple password audits. A more comprehensive dictionary wordlist that consist of nearly 4,000,000 words is available with John the Ripper at ftp://ftp.openwall.com/pub/wordlists/all.gz.

After adding one or more dictionary wordlists, select the desired password permutation options. Note that adding two numbers to the end of each wordlist increases the time needed to test all the words 100 times!

After selecting the desired options, simply click the *Start* button. Cain reports the number of words per second it can test, a progress indicator showing how much of the dictionary has been tested, and the current password.

In the example on this slide, Cain was able to determine a portion of the LANMAN password as "D". This is an indicator that the password is 8 characters in length. Because LANMAN passwords are split into seven-byte portions, Cain was able to evaluate the second portion of the password independently.

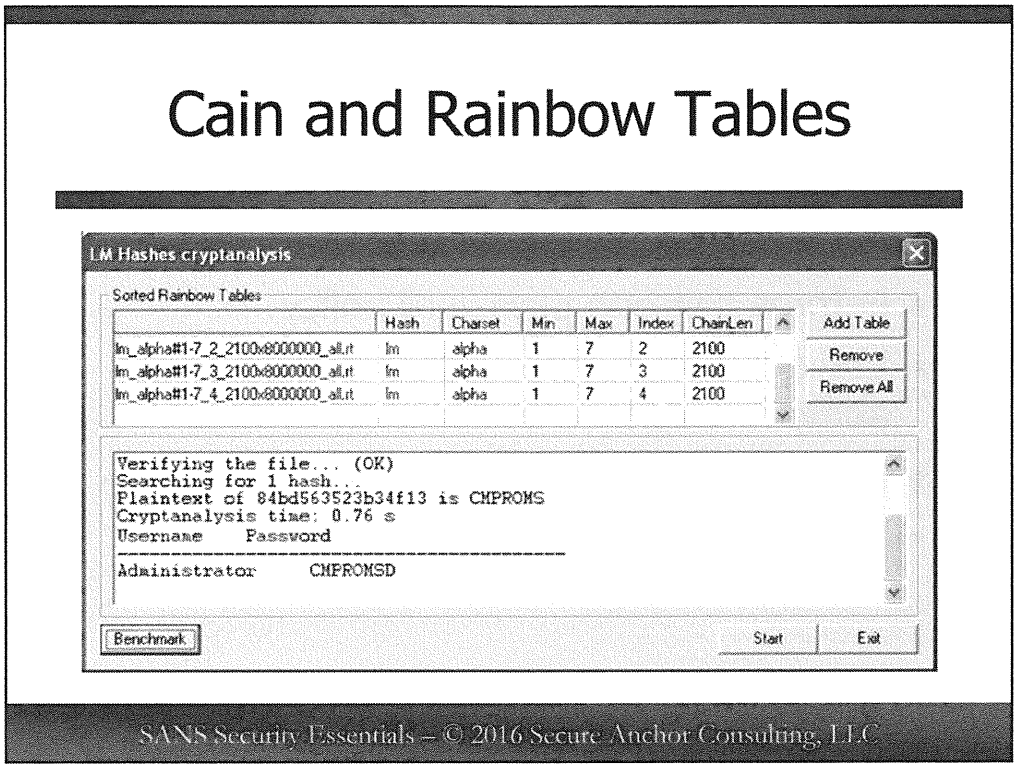


Rainbow Tables

Rainbow crack and the rainbow tables are an implementation of a faster cryptanalytic time-memory trade-off, published in 2003 by Philippe Oechslin. For those interested in the exact mathematic background of Philippe's research, I recommend reading his paper at http://lasecwww.epfl.ch/php_code/publications/search.php?ref=Oech03.

Research on pre-computing password hashes was not entirely new. In 1980, Martin Hellman described a cryptanalytic time-memory trade-off that reduces the time of cryptanalysis by using recalculated data stored in memory. Philippe managed to further reduce the number of necessary calculations. Reducing the necessary CPU cycles obviously makes this method even more suitable for cracking password hashes.

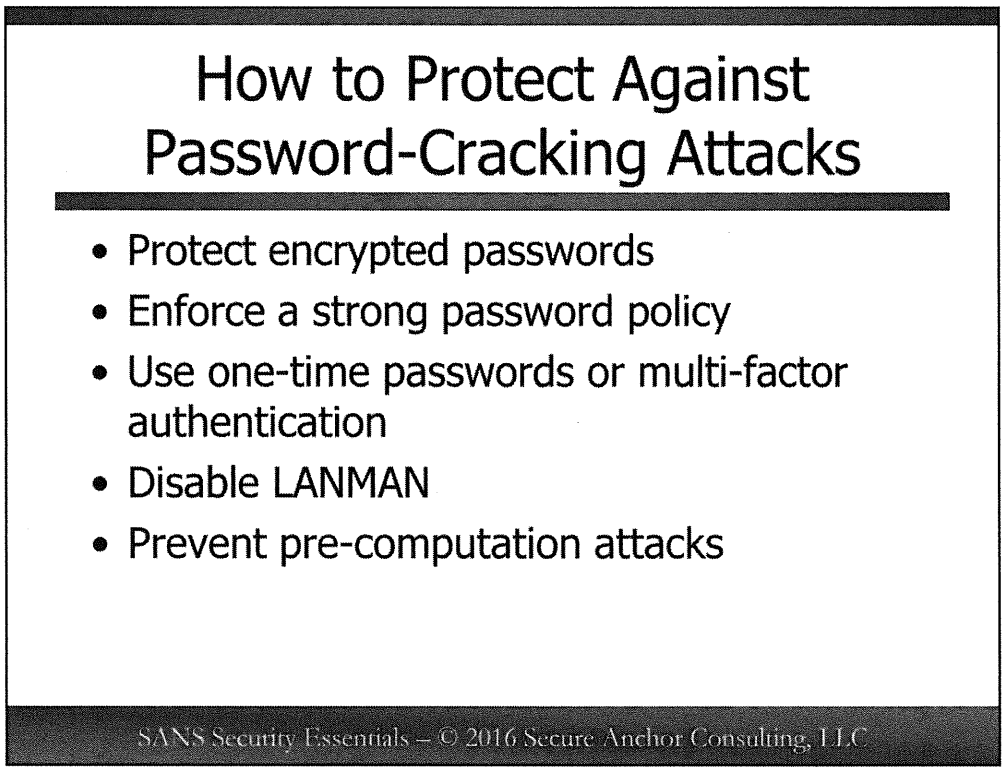
Rainbow Tables is the name given to the files that are produced by pre-computing password hash values and storing the data in an optimized manner to reduce the amount of disk space needed. After pre-computing the hash values, the cracking tool has to search only the tables for a given hash value to determine the corresponding plaintext password.



Cain and Rainbow Tables

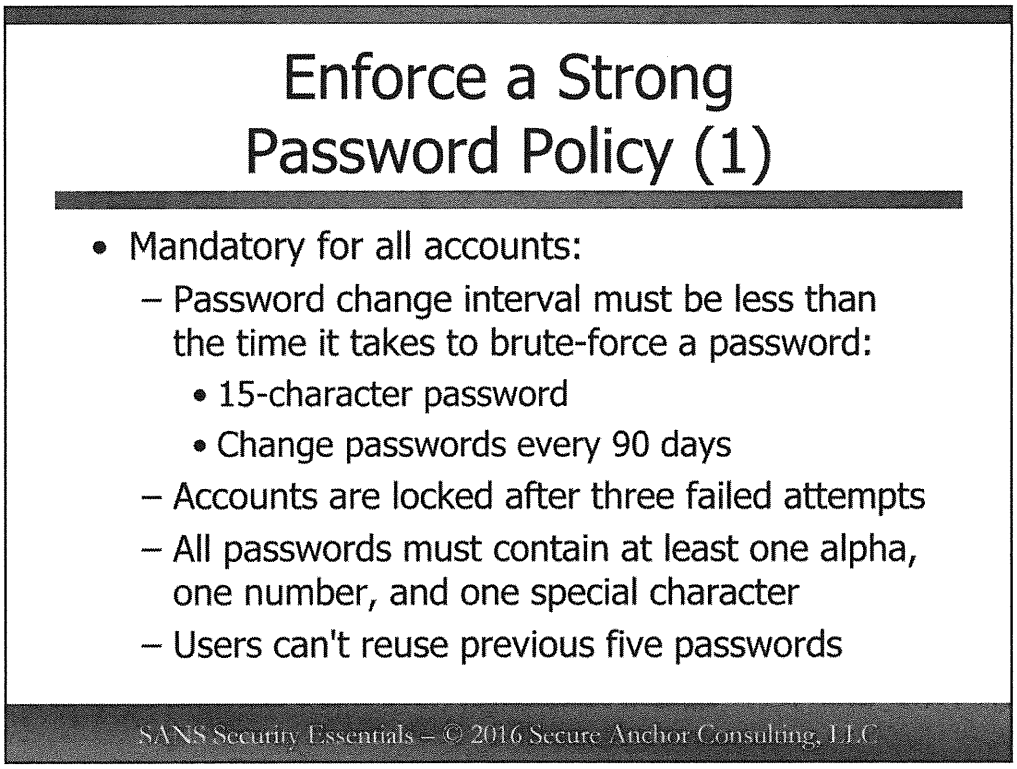
After precomputing the rainbow tables with Winrtgen or Zhu's Rainbowcrack tools, they can be incorporated with Cain to leverage the Cryptanalysis Attack option.

This slide presents the output from a successful compromise of the LANMAN hash for the Administrator account demonstrated in a previous slide. Although the dictionary attack wasn't successful at revealing the first seven characters of the password with a dictionary attack, leveraging rainbow tables we were able to identify the administrator password "CMPROMSD".



How to Protect Against Password-Cracking Attacks

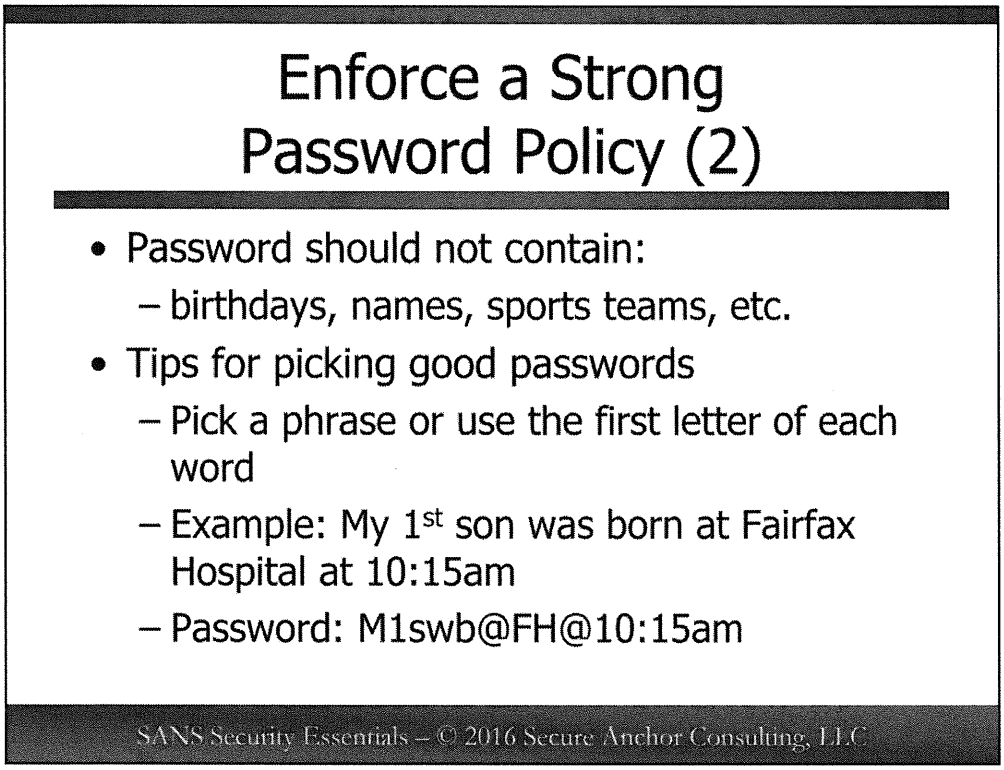
Now that we've examined some of the techniques that we can use to audit passwords in our own organizations, the next logical question is how do we prevent other people (attackers) from doing the same? Let's examine several techniques to protect against password-cracking attacks.



Enforce a Strong Password Policy (1)

A password policy is critical for a secure system. Even when you cannot strictly enforce a policy through automation, it's good to have a published policy so your users at least know what you expect. Here are a few general guidelines:

- Password change interval must be less than the time it takes to brute-force a password:
  - 15-character password
  - Change passwords every 90 days
- Accounts are automatically locked after three consecutive failed login attempts.
- Passwords must contain at least one letter, one numeral, and one special character.
- None of a user's previous five passwords can be reused.



Enforce a Strong Password Policy (2)

Even the most strictly enforced policy won't help if users don't protect their passwords. Your users' habits are the first line of defense against password theft. Here are some guidelines for users taken from the manual page for the Unix passwd(8) program:

- Don't write your password down anywhere or place it in an unencrypted file. Memorize it.
- Use unrelated passwords for systems controlled by different organizations.
- Don't share your password with anyone, in particular someone claiming to be affiliated with a vendor or your technical support department.
- Don't let anyone watch you enter your password.
- Don't enter your password on a computer you don't trust.
- Use a password only for a limited time, and change it periodically.

Choose a Hard-to-Guess Password

You can limit dictionary attacks by automatically enforcing a strong policy when users change passwords and by cracking passwords periodically. But these countermeasures cannot weed out every easy-to-guess password—it's always possible the attacker will have access to some piece of information about the user that your countermeasures don't. Again, we must rely on user education to fill in the cracks. Here are some do's and don'ts on the art of choosing a hard-to-guess password.

Do:

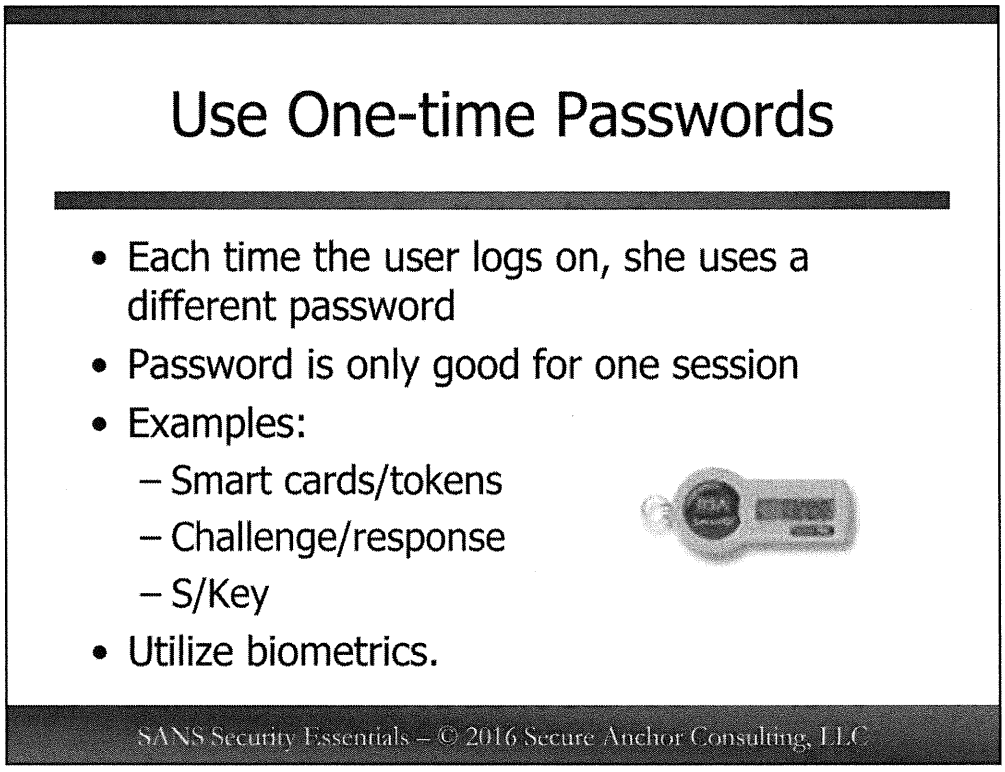
- Use a mixture of uppercase and lowercase letters as well as numerals or punctuation.
- Make sure new passwords are unrelated to any previous password.
- Use long passwords (at least 8 characters).
- Use a word pair with punctuation inserted, a passphrase (an understandable sequence of words), or the first letter of each word in a passphrase.

Don't:

- Use dictionary words (from any language) or jargon.
- Use a name, including that of a spouse, parent, child, pet, fantasy character, famous person, or location.
- Use any variation of your full name or account name.
- Use accessible information about yourself or your environment, such as your phone number, license plate number, or Social Security Number.
- Use a birth date.
- Use a simple pattern, such as a backwards word or a word preceded or followed by a digit.

A hard-to-guess password never contains birthdays, names of people or sports teams, or special interests. Anything that can be viewed while sitting at your desk should never be used as a password. Attackers easily can target an individual. Anything in the user's work environment, home, or web site that stands out as a potential password is too easy to guess.

Instead of picking words as passwords, try picking a phrase.



Use One-time Passwords

One-time passwords are very effective against password-guessing. Because the passwords change each time the user logs in, there is really no password to guess. The drawbacks are implementation costs and complexities and ongoing operating costs.

There are strengths and weaknesses to each of the approaches to one-time passwords. You will want to research each possibility to see which one is right for you. This section covers the approaches briefly to explain some of the options available to protect against password-guessing.

The most common way to implement one-time passwords is using token-based devices such as SecurID tokens. A user must have such a device handy when logging on to the system. The device is triggered by the time of day, so every minute the password changes. When the user wants to log on, she reads the current password from the token's display and types it in at the password prompt.

Instead of a time-based algorithm, some devices also use what is known as "challenge/response." The user presents her user ID to the system, which responds with a challenge. The user then types the challenge into the device, which generates a response. The user then types the response into the system at the password prompt.

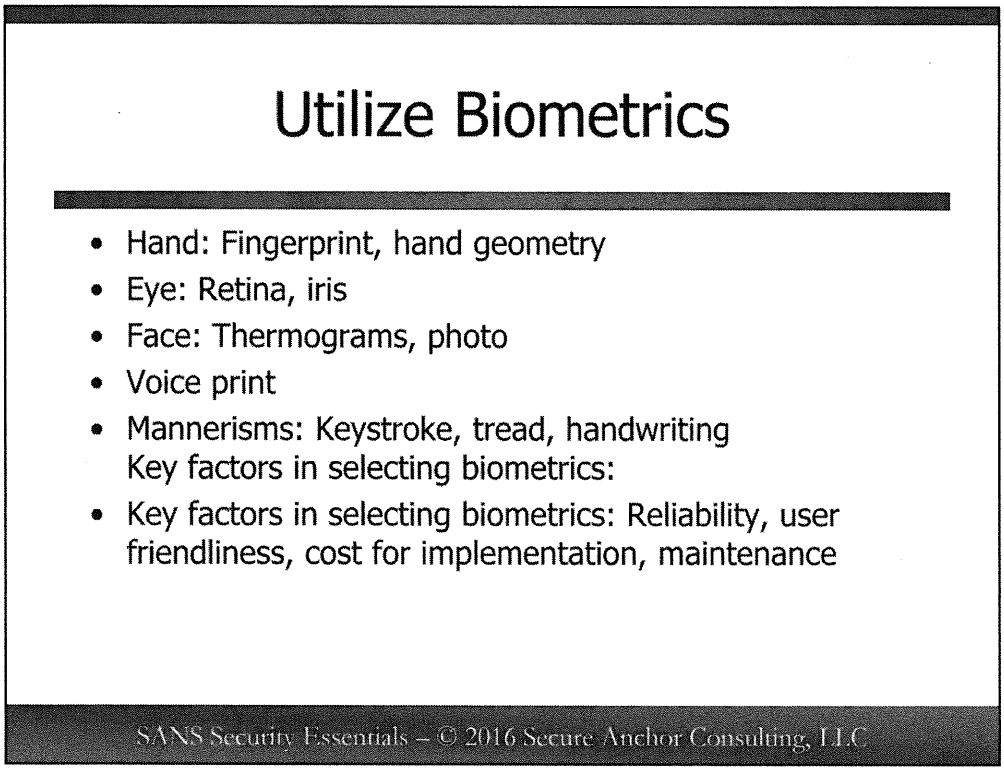
Some software-based one-time password systems exist that are usually less expensive. Often, the software itself is free and there are no cards to buy. The users themselves usually can handle the initialization of software-based one-time password systems, whereas a trained staff often is required to program token-based devices. One common implementation is called S/Key, which computes its one-time passwords when the system is first configured. Each user gets a pre-computed list of passwords. Each time a user logs in, she uses a different S/Key-generated password.

In general, one-time passwords are a good countermeasure against keyboard and network sniffing. Because each password can be used only once and cannot be used to deduce subsequent passwords, it doesn't usually matter whether the password gets sniffed while the user types it.

One-time passwords are especially useful when a user is taking a trip to a place where encrypted communications tools are not available. The user could take along the token-based device and generate passwords at will. With a software-based system, the user could print a list of the next several one-time passwords. Such a list might look like:

489: PER WU CERN EGAN RENA COED
490: LIES BAH NE FINK GAP TONG
491: KUDO CHUG LEW LAIN KNOT SOWN
492: CLOG GYM GROW HIND BAH MASS
493: TWO CHUM PIT GELD CAIN VAT
494: NOB JAN IQ FIB SEEN LAMB
495: SO MEEK ALIA PA ARK COMB
496: DOME SORE MUD TEST JOT MODE
497: TILE ALP COT JADE NEIL LYON
498: GOAD SOW WALL RUST FAST DRAW

When logging in, the user is presented with the sequence number, which in this case starts at 498 and decreases by one with each successful login. The user types the corresponding passphrase and crosses it off the list.



Utilize Biometrics

Biometric mechanisms use people's physical characteristics to identify them uniquely. The fingerprint is a commonly used physical characteristic that varies from person to person. We're probably all familiar with the finger and thumbprint scanners that take advantage of this human trait. Fingerprint scanners have advanced to the point of being incorporated into PC cards, which can be inserted into a desktop or notebook computer to authenticate users, or being incorporated into the laptops themselves.

Even more common than the fingerprint biometric is the photo ID, which is very useful because it can be used with "manual" biometric mechanisms. No computers need to be involved to compare the photo to the face of the person standing there in the flesh.

Effectively, biometric authentication boils down to a very long password that can never be reset. Biometric authentication systems that rely solely on biometrics for authentication are at risk when the identifying credentials for a person are stolen (such as reproducing a fingerprint left on a glass bottle), because it is not generally possible for someone to change his credentials after he has been compromised. Some biometric authentication systems require a two-factor authentication (such as biometrics and a PIN) to mitigate this risk.

Due to the unique nature of biometrics, we discuss some other critical points in a biometrics environment.

Portal throughput is the amount of time it takes for the system to authenticate an entrant and begin processing the next entrant. Ten seconds generally is the accepted tolerable portal throughput duration. Although ten seconds seems a reasonable amount of time, at the beginning of the work day, allowing only six employees through the door each minute might not be sufficient. Contrast this with the few seconds it takes to swipe a badge.

Error rates indicate the accuracy of the biometrics system. Type A or Type I error is the percent of readings in which the system fails to accept a genuine user. This is known as the False Reject Rate (FRR). False rejects might be due to an inability for the system to read the biometric (dirty scanner or user too far from reader) or an

inaccurate reading. In addition, certain physical characteristics change over time or in specific conditions. For example, diabetes causes a change in blood vessels in the eye, which affects the accuracy of retina scanners. Certain repetitive tasks can affect fingerprints over time.

Type B or Type II error is the percent of readings in which the system accepts an unauthorized user. This is known as the False Accept Rate (FAR). False accepts usually result from an intruder attempting to compromise the system. Examples include using a recording to dupe a voiceprint identifier or a fake hand to dupe a hand geometry scanner. Systems can be tuned to minimize either the FRR or FAR. Systems that are less rigid on matches will reduce the FRR; systems that are more rigid on matches will reduce the FAR. The crossover error rate (CER) is the rate at which the FRR and FAR are equal and is the generally accepted measure for a biometric system's accuracy.

Biometrics are also characterized by their intrusiveness. Users will cause a higher FRR if they are reluctant to approach the sensor. Systems, such as voiceprint, are significantly less intrusive than systems, such as the iris scanner, which require the user to place his eye in close proximity to the reader. In addition, systems that users must contact tend to require regular cleaning and replacement of the contact components.

Currently, the most accurate biometric systems are facial thermograms, but they often are too expensive for mass deployment throughout an enterprise. This technique senses the heat in the face caused by the flow of blood under the skin to uniquely identify a person. The most mature systems are fingerprint-based, because this technique has been around for significantly longer than the other biometric techniques.

Facial thermogram systems are being deployed in areas where the identification of suspects for law-enforcement departments is helpful in fighting crime.

The key factors in selecting a biometric mechanism are usually reliability, user acceptance, and cost. Three quantities typically associated with the reliability of a biometric mechanism are:

- False acceptance rate (FAR): The percentage of impostors the biometric mechanism falsely authorizes
- False reject rate (FRR): The percentage of legitimate users falsely rejected
- Cross error rate (CER) or equal error rate (EER): The rate at which the FAR and FRR are equal

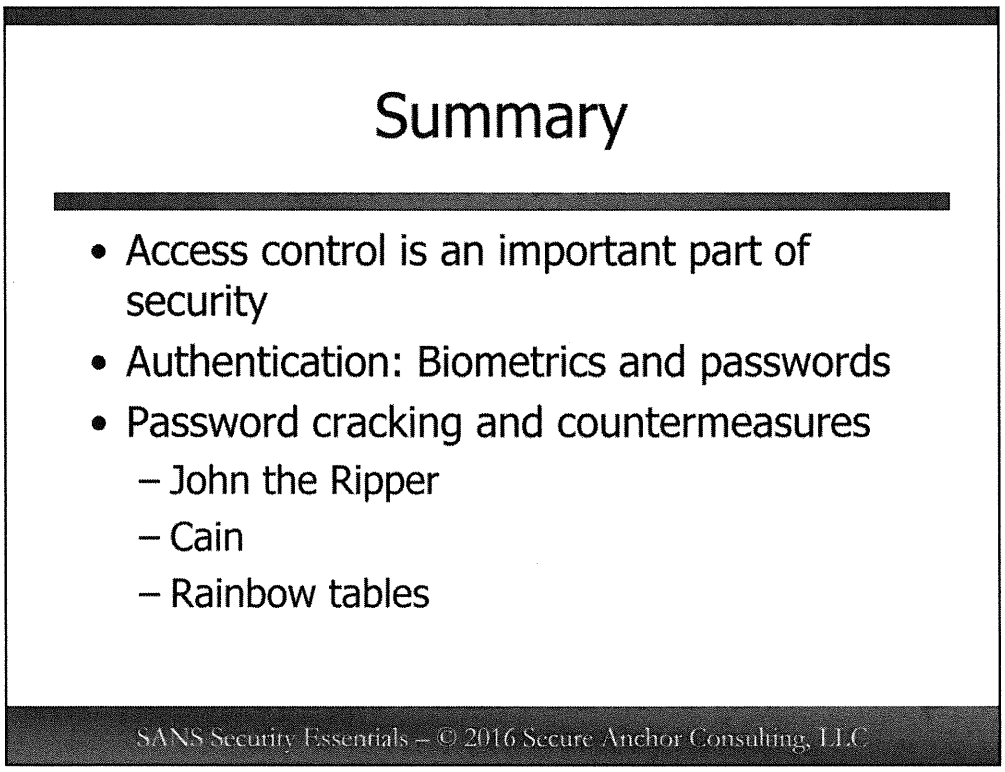
Vendors of biometric systems often quote these figures as they pertain to their products. But the numbers usually pertain to laboratory, rather than real-world, conditions.

The user-friendliness of a biometric mechanism is an important factor when choosing one. If people don't like to use a system, they will find ways around it. People might dislike a biometric mechanism if it's intrusive. People don't like to touch things other people have touched, nor do they like to get too close to certain types of machines. For example, they might prefer a voiceprint identification to a fingerprint or retinal scan mechanism.

Enrollment is the process by which the user's biometric is initially recorded, so it can be used for comparison each time the user tries to gain access. Enrollment should not be difficult, stressful, or time-consuming, or the user will be hesitant to use the system. At the same time, enrollment must sample adequate information to be extremely confident that the person enrolling is authentic and to avoid pushing up the FAR or FRR.

Cost can be the most influential quality when choosing a biometric system. As you might have gathered, the up-front cost of the system is not the only expense to consider—you need personnel to maintain the system and take on the task of enrollment. In addition, systems that the user must contact are more susceptible to wearing out and needing parts replacement.

Some technologies are still very expensive (thermograms are $10,000 per scanner). High prices can quickly force a security administrator to examine the current threat model to understand exactly what level of sophistication is needed for access control. Sometimes a conventional lock and key is still sufficient. On the other hand, security personnel should understand that taking custody of sensitive data means making room in the budget for an adequate access control mechanism.



Summary

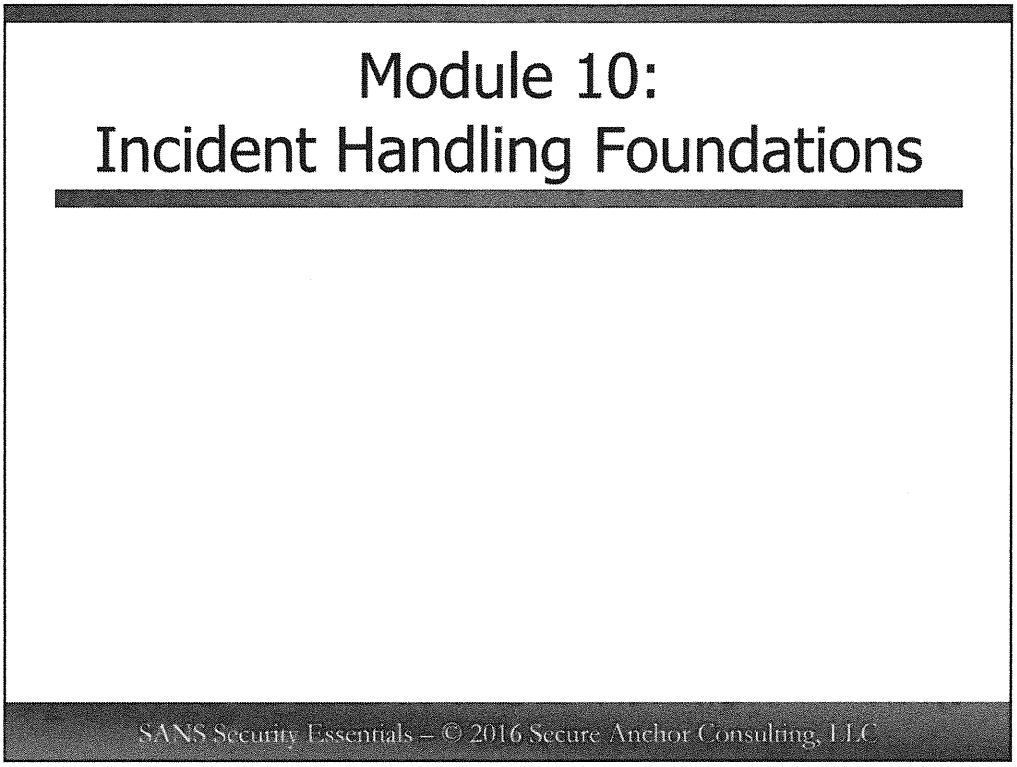
Access control is an important part of any security model that deals with sensitive data or resources. Models have been developed to provide confidentiality (secrecy) of data as well as integrity (trustworthiness).

This module covered two ways to authenticate users: biometrics and passwords. Biometric mechanisms analyze a physical attribute of a person and compare it to recorded data known about the person. One determining factor in choosing a biometric mechanism is the users' acceptance of such a system. An uncomfortable or time-consuming enrollment process can be dreadful to users. Also, users don't like systems that are intrusive or produce too many false rejects. Even though biometrics can reduce user workarounds and password sharing while increasing security, the cost of the systems can be significant and must be weighed.

Often, passwords are the only protection against identity and data theft. Hence, several techniques have been developed to audit and attack passwords.

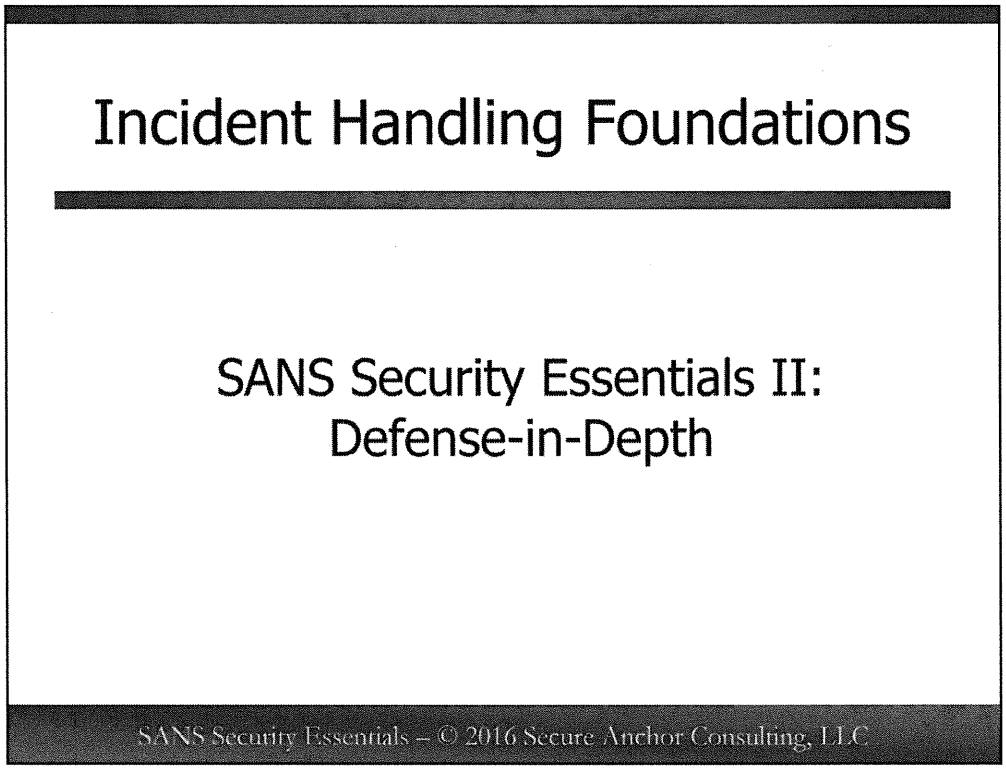
These techniques are surprisingly useful, despite the fact that operating systems protect password data with strong encryption and other countermeasures. The dictionary attack tests all the words in a wordlist, encrypting them one-by-one, against each of the hashed passwords. The hybrid attack also uses wordlists, but transforms each word, adding numerals and symbols. The brute-force attack is the most powerful but also the most time-consuming, trying every combination until a match is found.

Cracking Windows passwords tends to be much easier solely because of design flaws in the LAN Manager authentication scheme. Cain takes advantage of these design flaws for quick password cracking. Disabling LAN Manager authentication in favor of NTLM or NTLM2 is the best way to protect against Cain and other password-cracking tools.



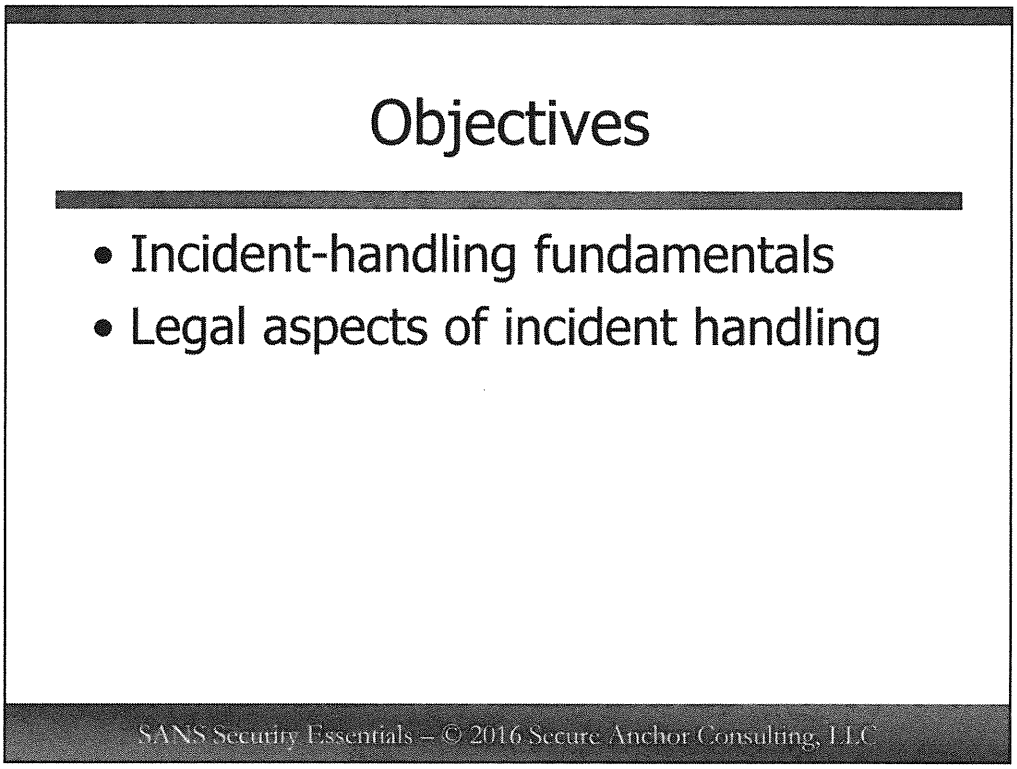
Module 10: Incident Handling Foundations

This section intentionally left blank.



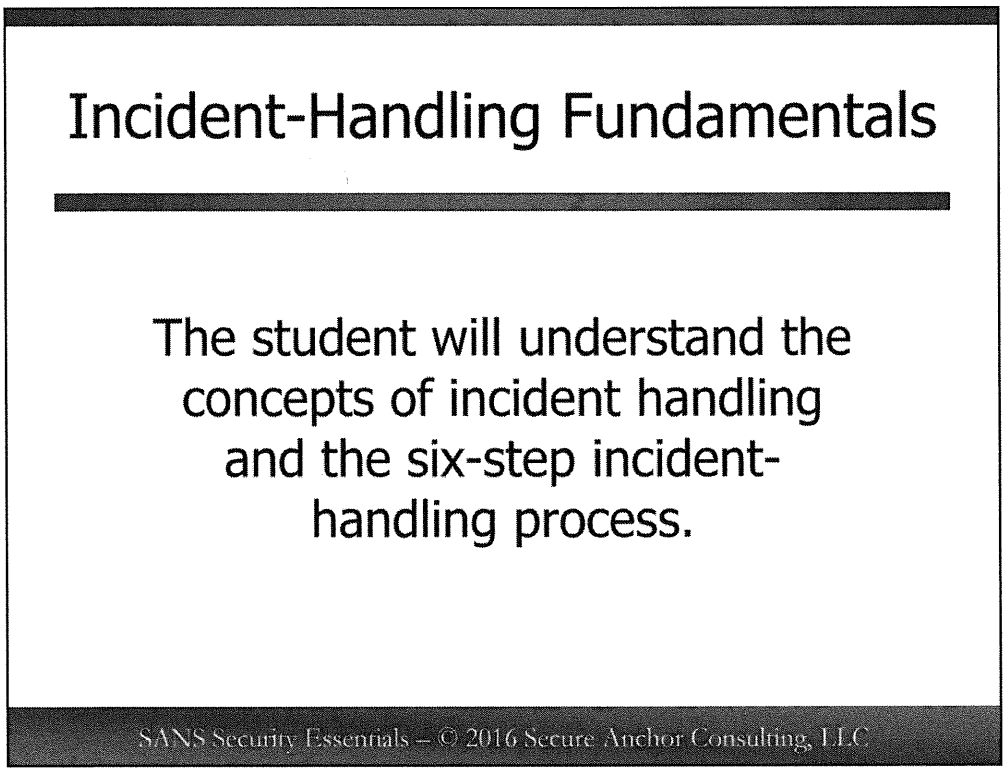
Introduction: Incident Handling Foundations

Have you been hacked today? How would you know? What would you do? Imagine receiving a phone call in the middle of the night and hearing a frantic voice from your company's Help Desk shouting, "Help! We've just been hacked! A group calling themselves the 'VORTEX' just extracted 200 financial records from our server!" As you wipe the sleep from your eyes and glance at the clock, your mind is racing for answers to questions you thought would never be asked. The voice on the other end of the line wants to know how this happened, how can he fix it, and most importantly, who is going to fix it? Obviously, this is a situation in which none of us wants to find ourselves, but chances are this will happen to you at some point in your career as an information security professional.



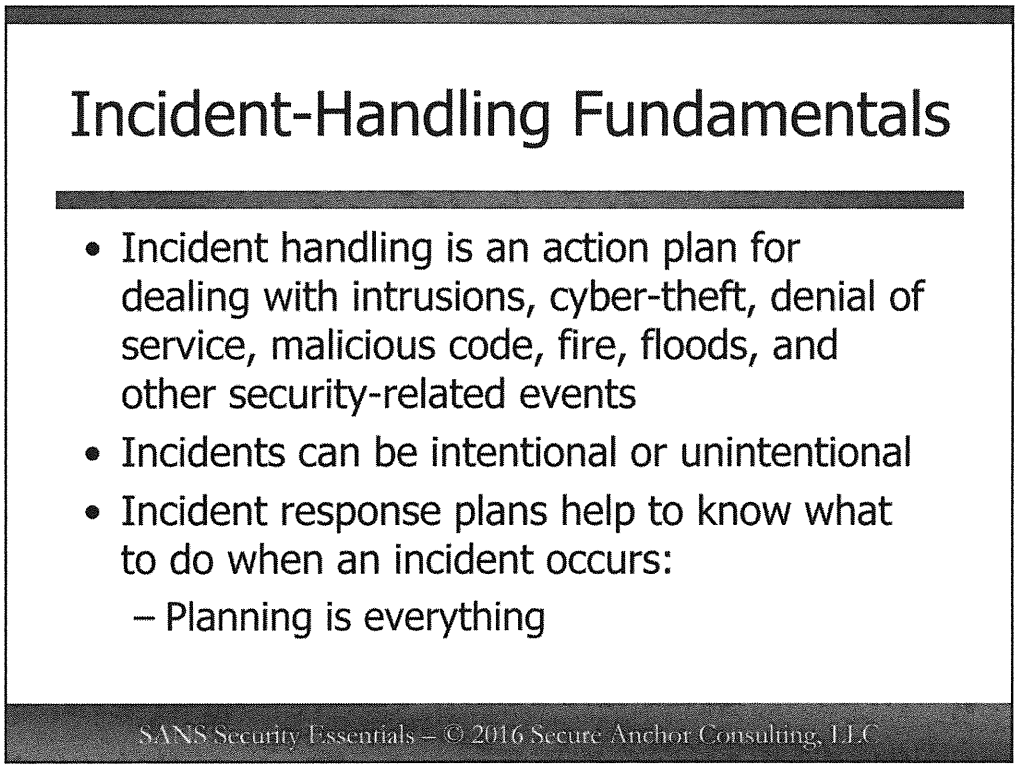
Objectives

A sad fact to consider is that a plethora of companies worry about their network or computer systems getting compromised, but very few have spent time preparing for that eventuality. With new security vulnerabilities released on a daily basis, we have gone well beyond the question of, "What if?" and have landed squarely on trying to answer the question of "When will we be hacked?" Understanding the basis of incident handling procedures is paramount to the success of maintaining a healthy security posture over time. In this chapter, we explore the fundamentals of incident handling and why it is important to your organization. We outline a six-step process to aid you in creating your own incident-handling procedures and, finally, we explain some of the laws relating to compromised systems and how you need to react to these threats in the event they become a criminal matter.



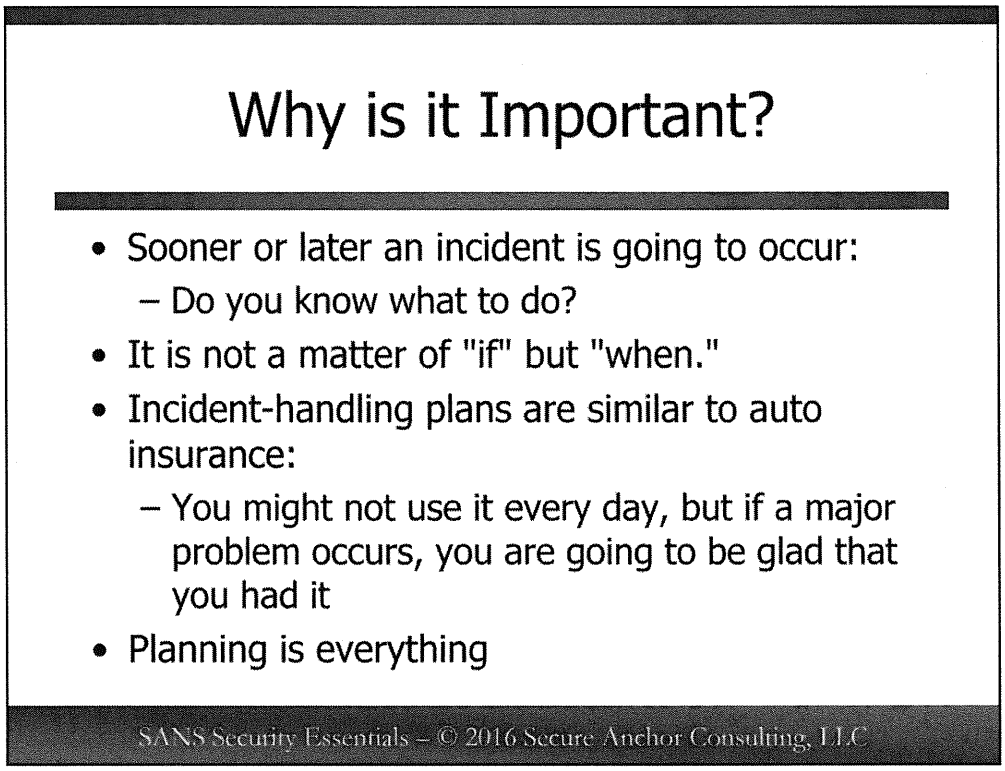
Incident-Handling Fundamentals

This section intentionally left blank.



Incident-Handling Fundamentals

Incident handling is the action or plan for dealing with intrusions, cyber-theft, denial of service attacks, malicious code, and other events. The scope of incident handling goes well beyond dealing with just intrusions; it covers the gamut from insider crime to anything that causes a loss of availability, whether intentional or unintentional. Natural disasters, such as fire or flood can be considered incidents because they represent a threat of harm to intellectual property or even the survivability of a company. Safeguarding intellectual property is becoming increasingly important as we move deeper into the information age. Brand names, proprietary information, trade secrets, patents, copyrights, and trademarks are considered to be extremely valuable data, and, as such, a plan is needed in the event this information ever gets compromised.



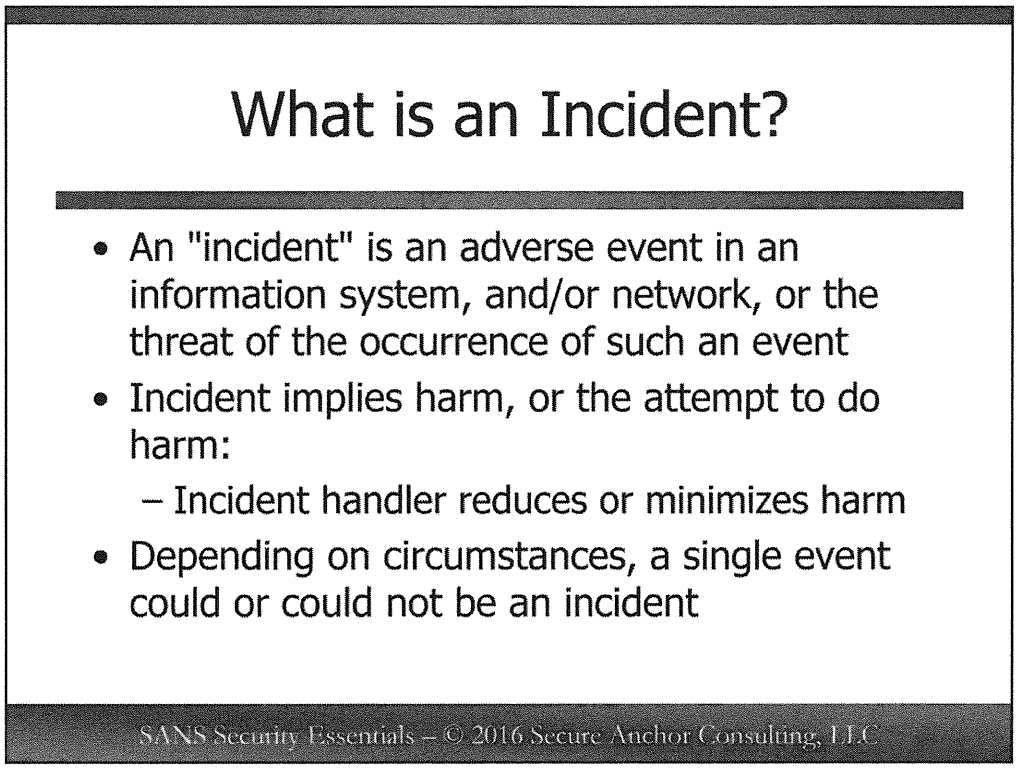
Why is it Important?

Another key point to consider is the concept of taking action during an incident. Observing an attacker in the process of defacing a web server or uploading a root kit is not incident handling. Identifying an action is important, but you must act on that information to secure your systems in a timely manner. One of the best ways to act on an incident and minimize your chance of making a mistake is by having well-documented, proper procedures in place. Being able to rely on solid documentation on what to do when an incident occurs will help in minimizing the chance that a crucial step in the process will be overlooked or forgotten.

Importance of Incident Handling

The size of your organization does not matter; the fact remains that sooner or later, you are going to experience an incident. In competitive markets, being prepared to handle an incident and handle it correctly could be the difference between thriving and disappearing. It might seem shocking given the widespread media attention to security vulnerabilities and exploits, but many companies have chosen to deal with an incident by simply ignoring the evidence of a security breach. The rationale seems to be: "We've never had an incident in the five years we have been in business, so why should we worry about it?" In this case, the truth of the matter is that the company probably had several incidents; however, because the company has not detected and reacted to the incidents, it has ignored the problem. It should be obvious that this mindset is very dangerous. For those companies that adopt this way of thinking, it is only a matter of time before it catches up to them.

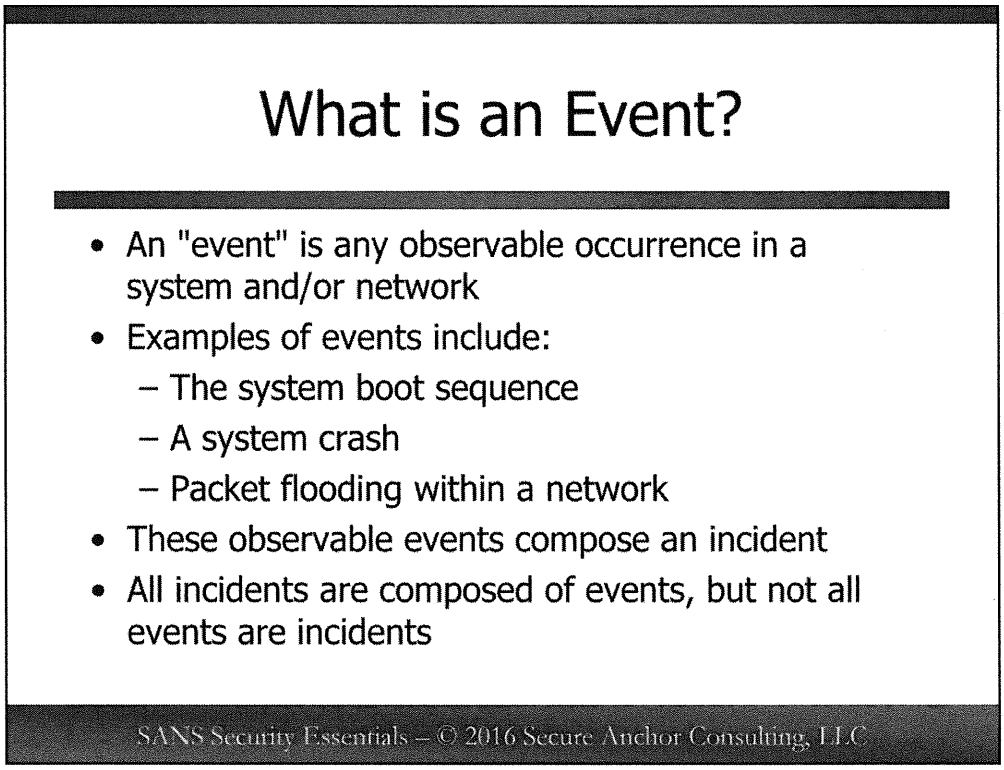
You've probably noticed that planning is a central theme in our discussion on incident handling. It is essential to the success of a strong incident-handling foundation. On the one hand, if you are prepared and know what to do, dealing with an incident can be fairly straightforward. On the other hand, if an incident catches you off-guard, you'll be in for a lot of sleepless nights. Although planning should be considered critical, don't get discouraged if you spend countless hours in planning and preparing for an incident and do not use those plans right away. It is easy to get discouraged and feel you have wasted a lot of your valuable time. Think of it as an insurance policy. Many of us pay our insurance premiums with the hopes of never having to file a claim; but when we do, we are happy and relieved we have that insurance!



What is an Incident?

For the purposes of incident handling, the term "incident" refers to harm or the significant threat of harm. Because we are dealing specifically with harm or the potential for harm, our task is to limit the damage. In the process, we need to ensure that we choose courses of action that do not cause further harm. The term "damage" means "impairment to the integrity or availability of data, a program, a system, or information." As an example, according to one law is: Damage means the defendant's conduct caused either loss exceeding $5,000, impairment of medical records, or harm to a person or threat to public safety.

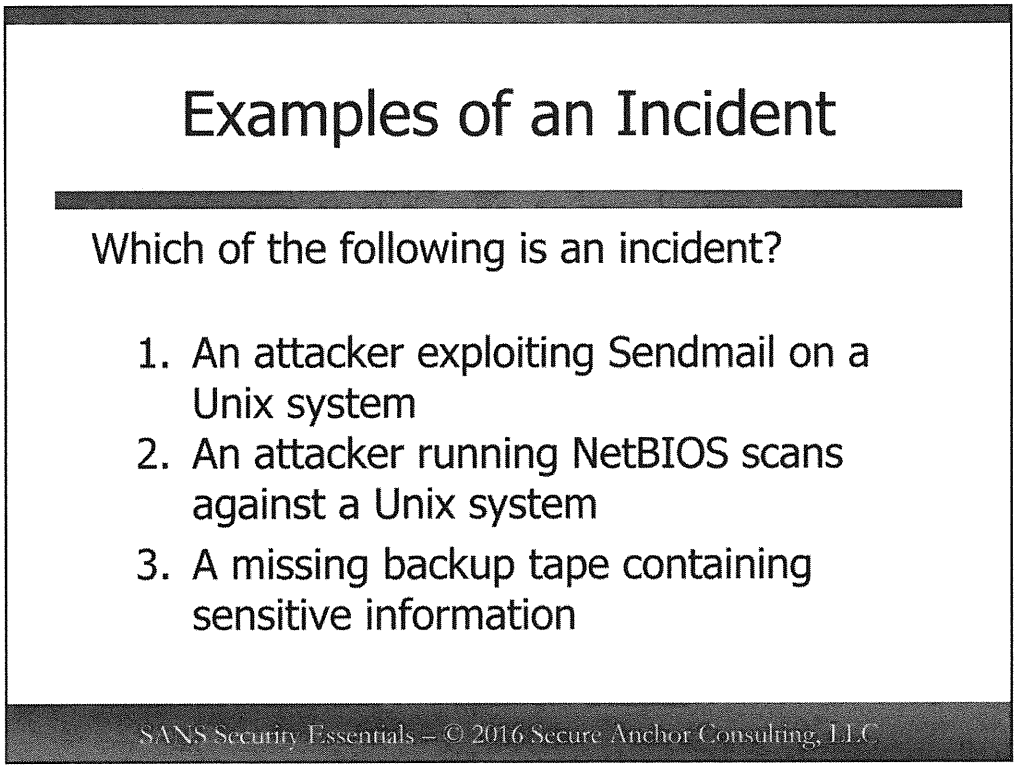
An incident is composed of one or multiple events (a series of events).



What is an Event?

It is important to understand the definition of an event in the context of incident handling. An event is something that happened in time that you either directly experienced or that you can demonstrate actually occurred. An event could be a message you witnessed that appears on a screen or something that you heard. It can also be something that you know occurred because it was collected in a log or audit file.

If you observed it happen or can prove it happened, then you are looking at an event. The key point to keep in mind when defining an incident versus an event is that all incidents are composed of a series of events, but not all events are considered incidents. For example, an unauthorized logon is considered an incident whereas an authorized logon is not, yet both are network events. Some other examples of events include a system boot sequence, a system crash, or even packet flooding within a network.



Examples of an Incident

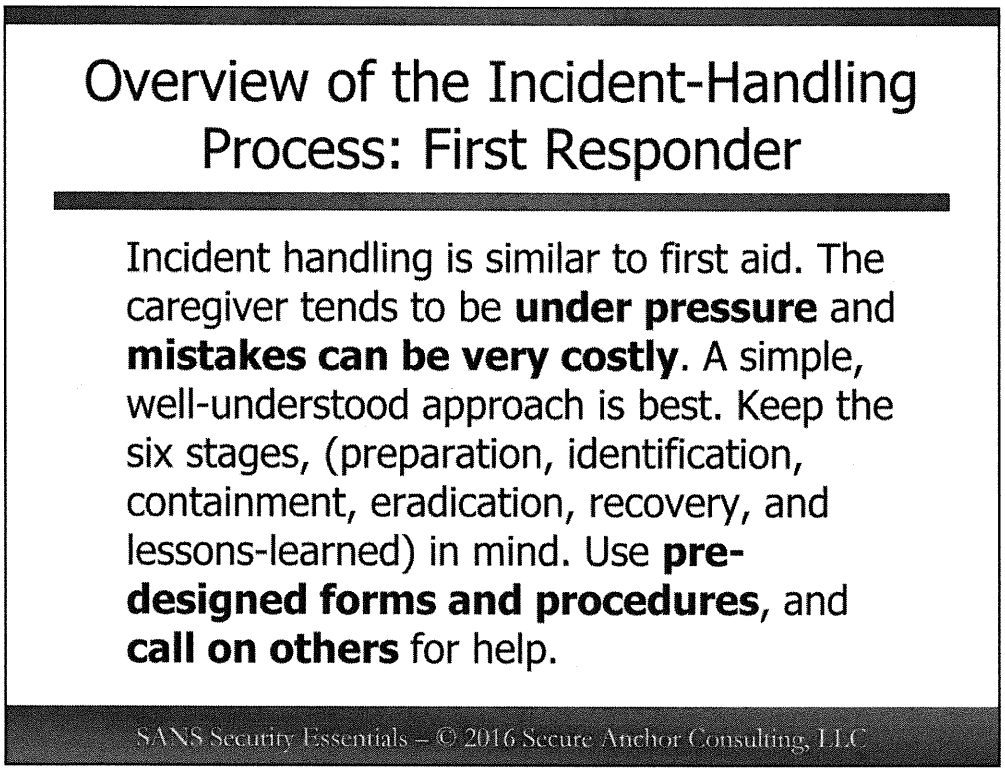
Now that you have a basic understanding of incidents and events, which of the following would you consider an incident?

- Attackers exploiting Sendmail on a Unix system
- Attackers running a NetBIOS scan against a Unix system
- A missing backup tape that contains sensitive information

If you answered, "Yes" to all three, then congratulations! Some might not consider the first example to be an incident because an attacker is running a Windows exploit against a Unix system, which would not be successful and, therefore, would not be a concern. In this example, however, we need to keep in mind the definition of an incident: harm or the threat of harm.

Even though this attack was not successful, a threat is still implied and there is a good chance that the next time the attacker might be successful, using a different target or set of tools.

Hopefully, it is obvious that the last two examples would be considered incidents. One is an unauthorized exploit that allowed an attacker to gain access to a Unix system; and the other, although not as glamorous as a remote attack, is still an incident because the tape is missing.

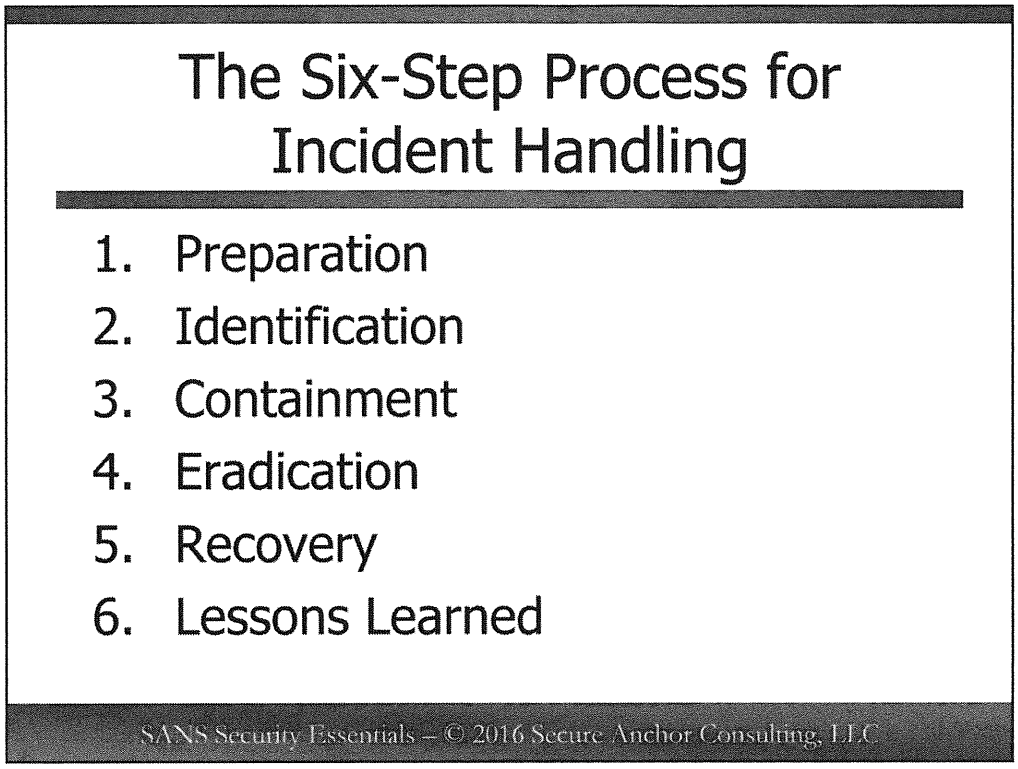


Overview of the Incident-Handling Process: First Responder

A good way to get an overview of the incident-handling process is to compare it to giving first aid. In both cases, time is not your friend. You are under immense pressure, and mistakes can be costly. That is not to say that you need to dive headfirst into a situation without thinking. Law enforcement agencies tell story after story of the well-meaning system administrator that ruined the evidence, usually within a couple of minutes after responding to the incident. You do need to act, but take time to think things through before beginning your work. We strongly recommend that you use pre-established procedures that specify how to act during common attack situations.

As part of the incident-handling process, pre-designed forms should be used to aid in recording events. These forms provide a convenient way to document each step of the handling process and to ensure that crucial information such as dates, events, people involved, and systems affected is not missed or overlooked. Some examples of these forms include important contact information, incident survey, and incident identification forms. There are many templates available from a wide spectrum of Internet resources.

As is almost always the case in legal matters, having corroborated information is better than a single source that claims the event happened. For instance, if two people witnessed a message flash on the screen, it will likely have more validity in court than if one person saw the message. In addition, attackers sometimes use tools to alter or delete their traces in log files. In this case, if you can produce two independent sources for the information, there is more validity to help discount the deleted log files.

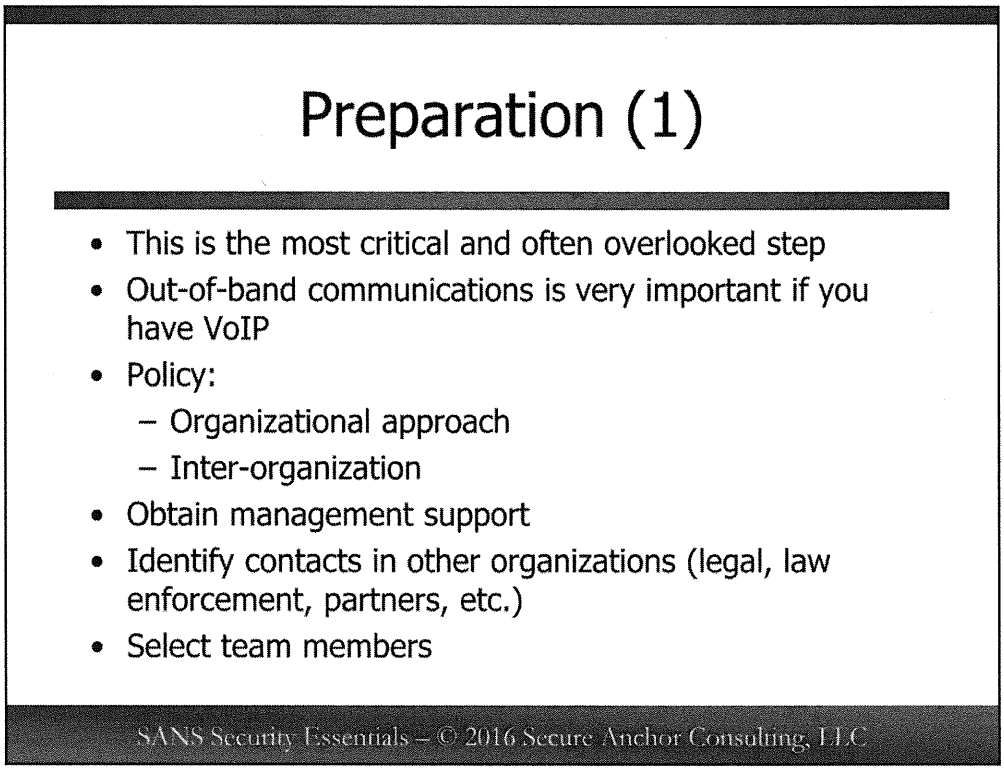


The Six-Step Process for Incident Handling

Based on the importance of incident response across the industry, it is important that a clear and standard process be followed. To create a starting point, the U.S. Department of Energy (DOE) led an initiative to build a six-step process back in the early 1990s. The six-step process used in this course and throughout the industry is based on the original process developed as part of a joint effort lead by DOE.

The six steps listed here can help serve as a roadmap or a compass, if you will, to develop a phased approach to incident handling. Keep in mind that in order for this process to be successful, each step must be followed. The six steps are:

1. Preparation
2. Identification
3. Containment
4. Eradication
5. Recovery
6. Lessons Learned



Preparation (1)

The preparation step is the first and most critical step of the incident-handling process. The tasks associated with this step must be performed in advance, before the incident has occurred. This is the reason why it is often overlooked—or even skipped. SANS recommends that you spend enough time preparing all the elements that are required during an incident, with the goal of increasing the efficiency and success of your incident-handling efforts.

When it comes to incident handling, planning is everything, and preparation plays a vital role. It is very important to have a policy in place that covers an organization's approach to dealing with an incident. One item that a security policy needs to cover is whether a company is going to notify law enforcement officials or remain silent when an incident occurs. The answer to that might depend on the severity of the incident; if so, what guidelines should the responder use to decide whether to call? If you are going to contact law enforcement, have a list of phone numbers for each agency you might need to involve.

Another important item to consider is whether to contain the incident and move into cleanup phases or to observe the attack in an attempt to gather more evidence. The policy should also contain direction for intra-organization incidents and how the company works with other companies regarding an incident.

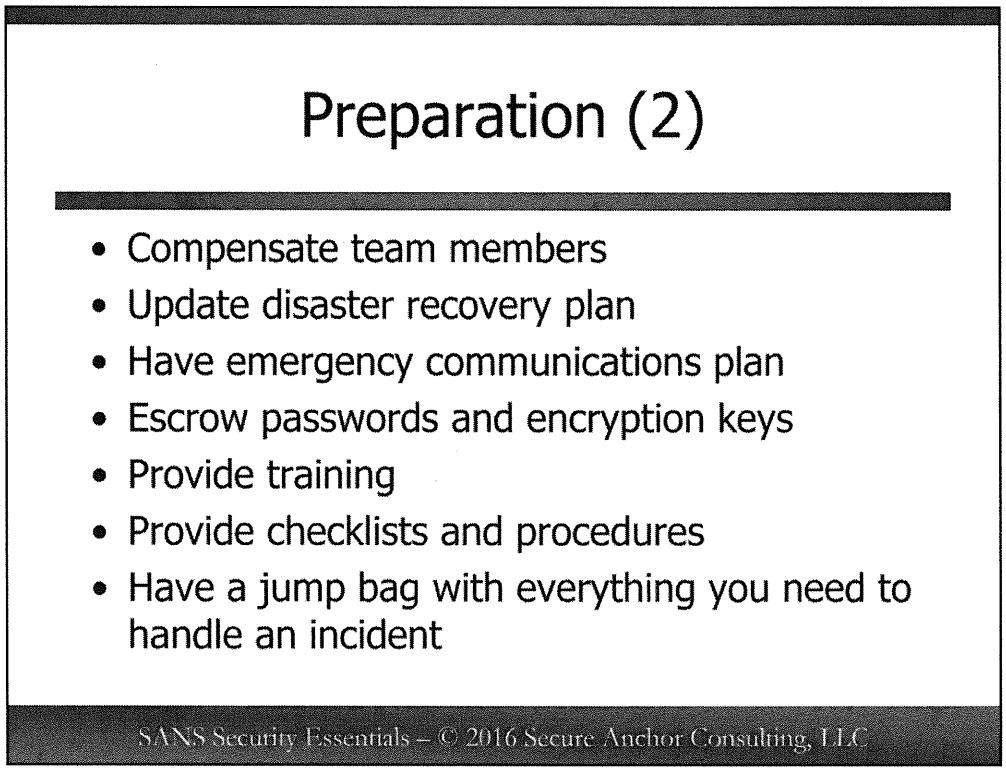
Incident handlers can be under extreme pressure. Consider a worm that infects your entire infrastructure, effectively making your network systems unusable. This is one reason incident-handling teams must never rely of Voice Over IP. If you have a VoIP installation, consider the use of cell phones, walkie talkies, or some other back-up method of communications. Incident handling can become a large-scale effort involving many people on many systems simultaneously. This should be taken into account during the planning phase.

The time to make these types of decisions is before the incident, keeping senior management and legal staff apprised of any changes to policy. Because of the sensitive nature of incident handling, any decisions made

could greatly affect your career down the road if you did not get approval or reach consensus with management. The last thing you or your company wants is for senior management to question or doubt the decisions that were made during an incident.

The last thing you or your company wants is for senior management to question or doubt the decisions that were made during an incident.

When it comes to selecting members of the team, keep in mind that not everyone makes a good incident handler. There are some very smart people in this industry whose personalities do not lend themselves to work under immense pressure and as part of a team. People who like to work solo and need to be the hero usually do not make good team members. Ideally, a person should have a strong technical background, thrive in a team environment, and have the ability to make sound decisions grounded in reality.



Preparation (2)

As the incident response team begins to mature and has responded to several large incidents, it is possible that members of the team will get burned out and leave the team. Although this is certainly understandable, an approach you might want to take is to provide compensation and other rewards for members of the team. This might run counter to your current policies, but keep in mind that incident handlers are often called to perform their duties after normal business hours, weekends, and holidays while under a lot of pressure to get things restored as quickly as possible.

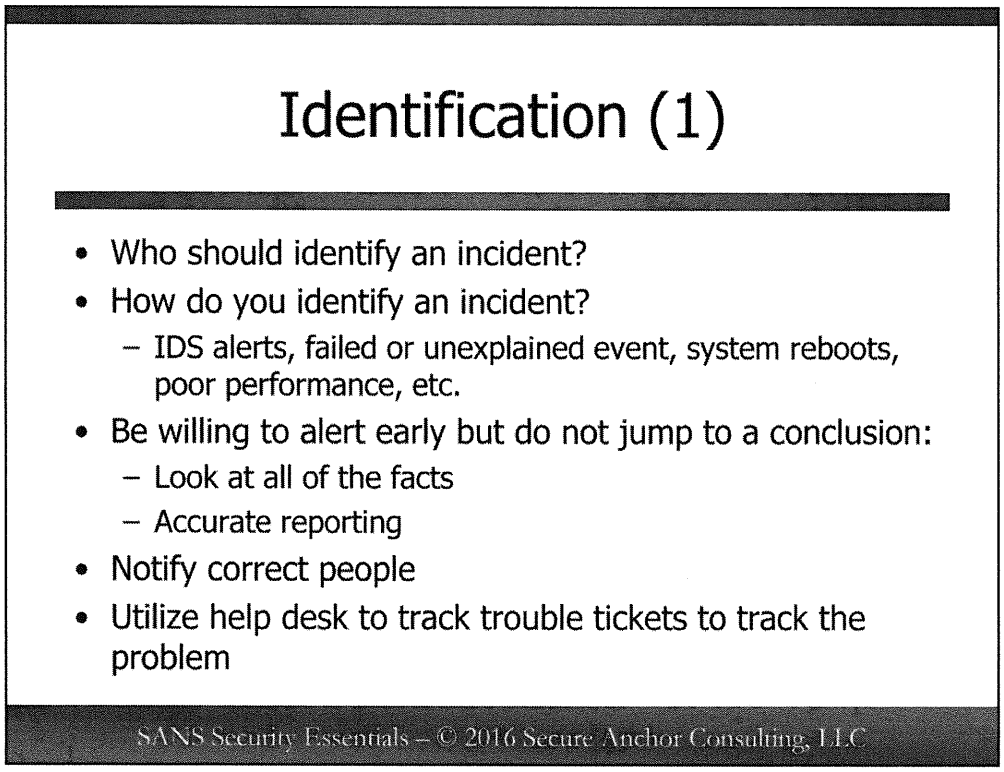
During the preparation phase, an organization should make plans to update its disaster recovery plan to include incident handling. After all, what is a disaster? It is an incident and needs to be handled as such. Although disaster recovery plans are often thought of as a checklist to get a business back up and running as quickly as possible, the skills possessed by the incident-handling team could be put to good use to reach this goal. In addition, the disaster recovery plan and the incident handling procedures guide should contain information for emergency communications.

The issue of making privileged passwords available to others can be a delicate situation. However, in an emergency, a handler might need access to critical systems.

One idea to consider is to incorporate a procedure where system passwords are kept in sealed envelopes in a locked container or datacenter until they need to be used. This might seem cumbersome, but it does work and keeps the passwords private until they are needed by the incident-handling team. In order for this to work, the system administrators must keep the passwords in the sealed envelopes up to date, and the incident handlers must make every effort to tread lightly on the systems, inform the system administrators of any changes made, and above all, never use a privileged password unless they are qualified on that operating system. One thing that will certainly make an incident worse is having someone who has no idea what he is doing issuing commands as administrator or root.

Our computing environments are complex and will change over time. Training is critical for each member of the incident-handling team. Memory fades over time, especially if the members are not working on honing their skills on a regular basis. Having a checklist on how to bring a system down safely or on how to restore a system from tape can help in preventing errors and can reduce the stress on the handler. If your team is following a checklist and the resulting operation fails, it might be the fault of using an outdated checklist on a regular basis, so ensure they are updated to your organization's current environment.

Reaction time to an incident is absolutely critical. Every effort should be made to find members of the incident-handling team who will be able to respond on short notice. For example, an incident handler who has a two-hour commute into work might not be that helpful for a situation that requires immediate attention. One way to mitigate the effects of delayed reaction is using what the military calls a jump bag. This bag should contain in a central location everything needed to respond to an incident. Items such as contact numbers, checklists, telephone, notepad, pencils, and so on, are items that you would want to include. Also, as far fetched as it sounds, spare network cables, a hard drive, mini-hub, and tools for working on a PC should be considered essential.



Identification (1)

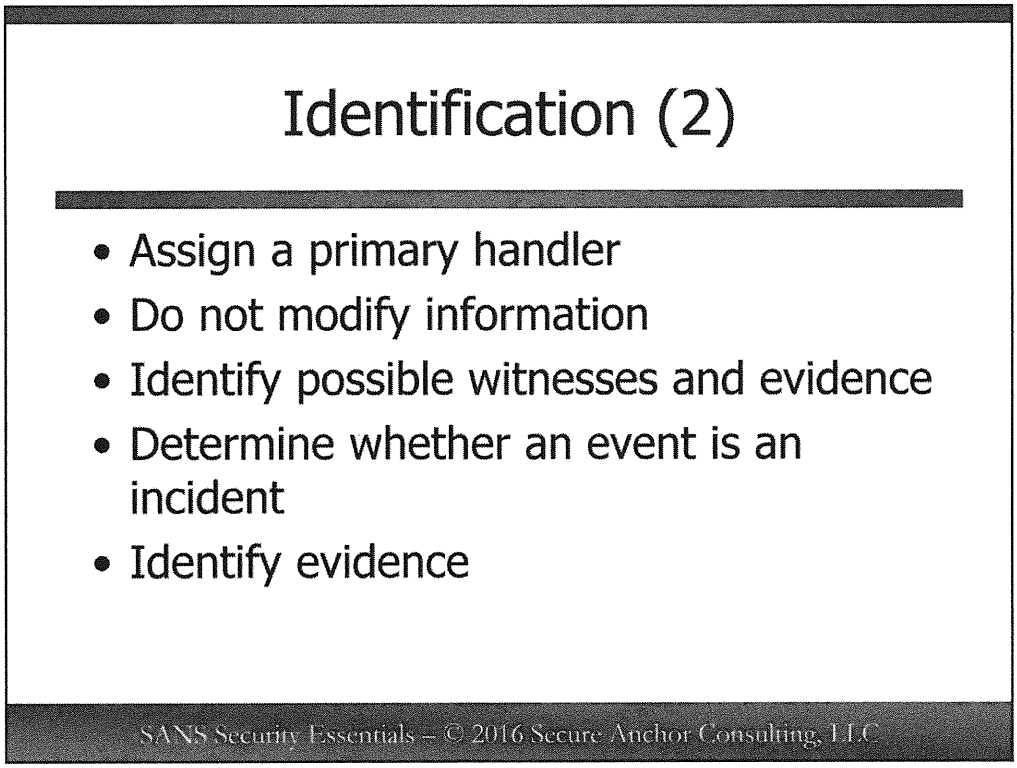
When it comes to identifying an incident, members of the team should stay with their realm of expertise. You would not want a Windows expert digging around a Unix system, and vice versa.

Some possible signs of an incident that might warrant further investigation essentially include anything suspicious, such as intrusion detection alerts, unexplained entries in a log file, failed logon events, unexplained events (such as new accounts), system reboots, poor system performance, and so on.

Being able to correctly identify an incident could be the difference between cleaning up the problem in a few minutes and causing your organization's network to be down for several hours or even days. Obviously, any system outage could potentially cost your company a lot of money, so it is important to be able to identify an incident correctly the first time and respond accordingly. For example, after a fire alarm is pulled and a building evacuated, qualified firefighters respond to the scene and investigate. Only then does the firefighter in charge at the scene authorize re-entry into the building. This should be the paradigm we work under—be willing to alert early, have trained people look at the situation, and be able to stand down quickly at a minimum of expense if nothing is wrong. No matter which course of action you decide to pursue, make certain you have mechanisms in place to correctly identify an incident.

There is nothing wrong with alerting early if you maintain situation awareness, and everyone understands this might not be an actual incident. All attempts should be made to avoid overreacting to the situation and escalating it too fast, only to realize an hour later that you made a mistake. If that happens enough times, you could fall victim to the "boy who cried wolf" syndrome; and then when a real incident occurs, no one will believe you because of the false alarms.

Chances are that your organization has a 24x7 help desk operation that would be ideal for helping out with tracking the incident and maintaining a paper trail. They could also be utilized to facilitate communication and contact other personnel as the situation warrants.

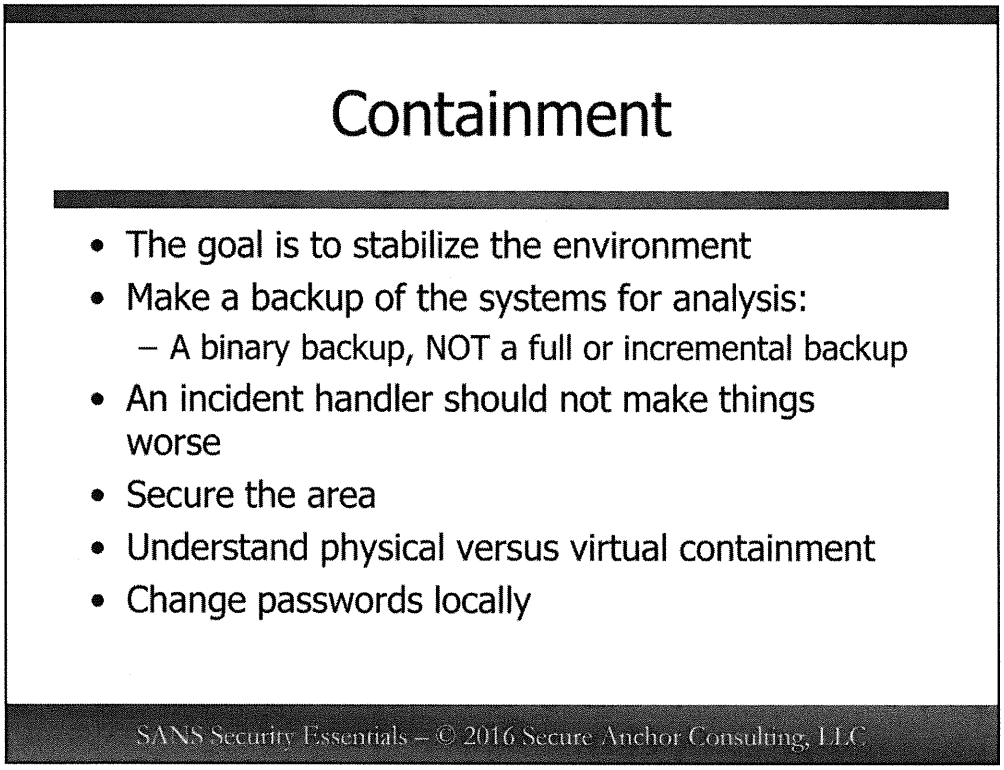


Identification (2)

It is important to keep in mind that a primary handler should be assigned as a team leader to keep the process flowing while also making sure that no steps are overlooked or missed. For smaller incidents, often of the "Would you check this out?" category, there isn't a need to send a core team of incident handlers. It is a recommended practice to have a core team of well-trained handlers and also have incident-handling skills and training as part of the job description for security officers and system administrators. An organization that adopts this approach benefits by having multiple layers of "firefighters."

However, in such a case, it is important to assign tasks in a way that encourages cooperation among the team and allows all members to succeed. When assigning tasks to part-time members of the team, do so in a way that it is clear what is expected of them: the quality of their investigation, their responsibility to preserve and collect evidence, what documentation they should produce, and when it is due. It is also important that they know who they should contact if they feel they need additional guidance or support.

After you determine that the event is actually an incident, the handler might decide to take the steps needed to build a criminal or civil case. In this situation, witnesses should be identified, and a written statement of what they heard or saw should be taken immediately while the information is still fresh in their minds. If a decision is made to involve law enforcement, make sure senior management is notified, unless you have a detailed policy to follow.



Containment

Okay, we have spent countless hours preparing for the eventuality of an incident. We have a good idea of what it takes to identify an incident, but where do we go from there? Being able to identify an incident solves only part of the problem. We are still left with the task of isolating and eliminating the source of the incident. This section discusses some steps that can be taken to contain an incident and, hopefully, limit its damage to the organization.

The primary responsibility of the incident handler is to make things better while adhering to the basic principles of liability and negligence. Negligence for failure to meet a certain standard of care is generally determined by a court of law. Specifically, negligence is defined as, "the failure to exercise the degree of care expected of a person of ordinary prudence in like circumstances in protecting others from a foreseeable risk of harm in a particular situation." In other words, a handler is responsible for meeting the expectations of the prudent person rule. Typically, a company that acts reasonably or with due care generally will not be found negligent.

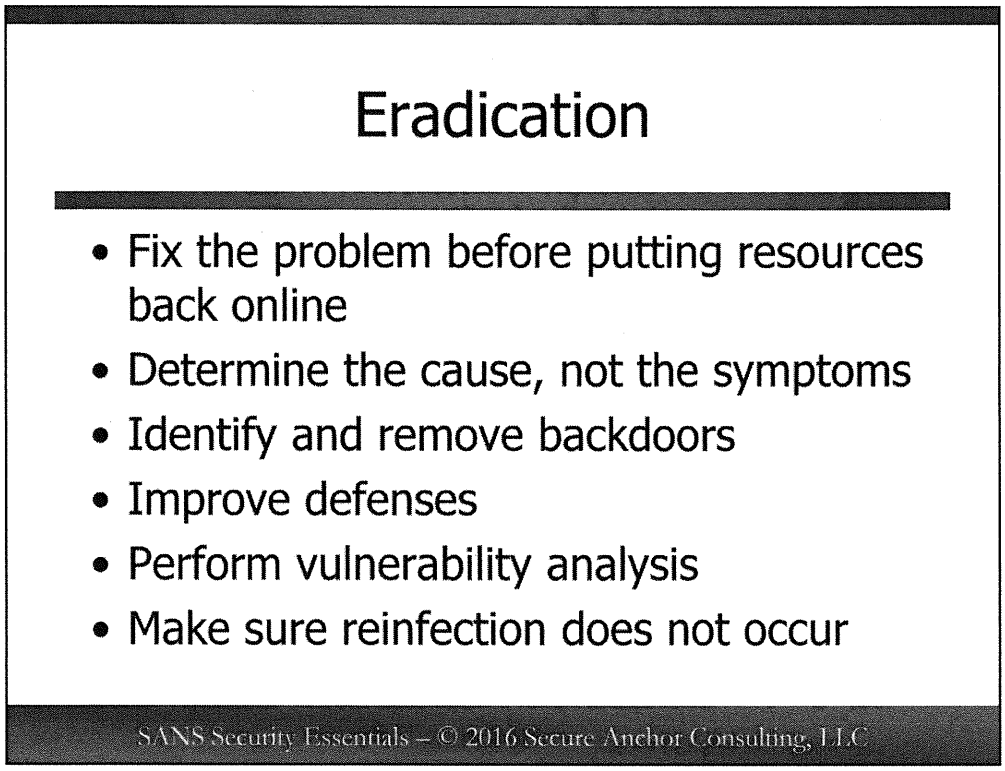
There exists a potential for an incident handler to run into trouble while performing her duties. There is no aspect of incident handling that allows a handler to break the law. As an example, if you suspect someone within your organization of downloading child pornography, you can't download these files to your computer to examine them. Also, a handler needs to exercise due care with regards to a person's privacy under the Electronic Communications Privacy Act.

For instance, if you are an Internet service provider, you cannot just release the personal information of a subscriber simply because someone claims she was attacked from the subscriber's IP address.

You should also be aware that corporate officers within your organization might be held liable for your actions if they are considered unlawful.

In containing an incident, you must first secure the area. In doing so, a forensically sound backup should be made of all infected systems. If the original hard drive cannot be kept for evidence, multiple copies of the backups should be made for future analysis, if needed. One copy should be kept for evidence and the other copy used to analyze the incident. At some point in the containment process, a decision needs to be made of whether the systems should be pulled off the network or whether the entire network should be disconnected from the Internet. Also, passwords should be changed as soon as possible to make sure a compromised account couldn't be used for reentry into the system by a remote attacker.

One of the key aspects of the incident-handling process is to be able to present, with a high level of detail, the different pieces of evidence found and all the actions performed during the whole process. For this purpose, you should take detailed notes of all the events associated to the incident, from the Identification (step 2) to the Recovery (step 5) phase, preferably using numbered paper notebooks.



Eradication

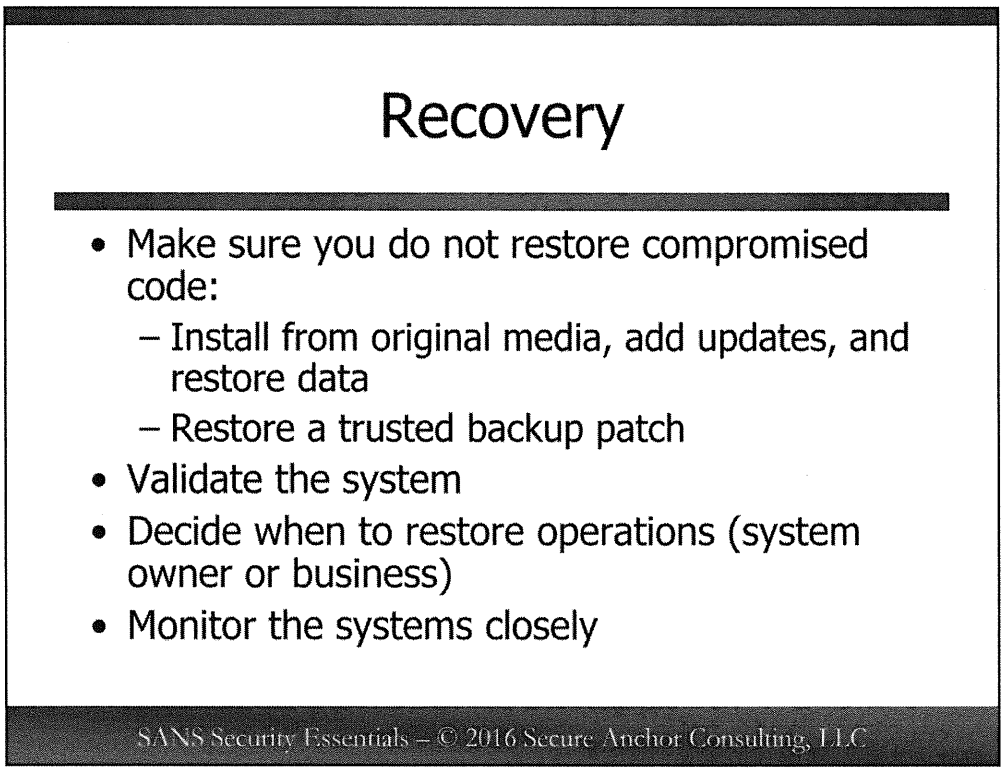
Before the system goes back online, an incident handler must make sure that she fixes the problem or the vulnerability that the attacker used to compromise the system. At first glance, the tendency might be to wipe out the entire operating system and rebuild it from scratch. Although this is certainly an effective way to remove any malevolent code, the opportunity for re-infection via the same channel still exists. There are a myriad of cases where systems were taken offline, rebuilt, and put back on the network only to be compromised again within minutes or hours. This is because a root cause analysis wasn't performed to determine why the incident happened in the first place.

It is not enough to simply recover the system and put it back online: The underlying security mechanisms of the affected systems must be altered, fixed, or upgraded to accommodate any new vulnerabilities. If it is a production system, you might hear voices of dissent from the organization about modifying a server running on a production network. This is an important, and to an extent, valid argument, but the counter is that if the system was compromised, then it must contain a vulnerability that might exist on other servers and could be exploited on a continual basis until the problem is fixed. Further, manually cleaning up the damage from an incident does nothing to prevent the problem from occurring again unless the problem is accurately identified and removed, patched, or otherwise mitigated.

Attackers often try to establish additional ways of ensuring remote access to the compromised system, so they have control of it even if the vulnerability exploited originally is fixed. Such backup access methods are known as "backdoors," and are implemented using several methods. Some of the most common ones include a process of listening on a specific port and offering shells access (without requiring authentication), creating a new user account with high privileges, and scheduling jobs that periodically run programs that open new paths to access the system. As a wide incident handler, you need to not only fix the vulnerability used during the initial system compromise, but also identify and remove every additional backdoor left by the attacker.

After the system is recovered, it is a good idea to run a vulnerability scanner against the affected system to see whether the problem is, indeed, fixed and that no new holes were opened up in the process. There are a number of commercial products, such as ISS Internet Scanner, that work very well and produce nice-looking reports, but open source tools such as OpenVas should not be overlooked. If your organization is on a tight budget, and you need tools that perform the task with great efficiency, then you owe it to yourself to explore the open-source options available.

To sum up, your main goal as an incident handler is to make sure that a new compromise using the same, or even a similar, vulnerability does not happen again.



Recovery

The key point to consider in the recovery phase is to ensure you are not restoring vulnerable code that has already proven itself to be exploitable by any number of attack methods. For example, if you restore a system from tape backup, then you could be restoring a previous state that contained the vulnerability exploited by the attacker. Vulnerable code, in this context, refers to operating system software that hasn't been patched to the latest levels, source code, and/or application software being used on the affected system. Although there is no easy solution, using a file integrity tool such as Tripwire might help in restoring the system to a known good state. Use Tripwire to take a snapshot of the compromised server, restore from tape backup, and run Tripwire again to compare the results. This method will tell you exactly what files were changed, modified, or deleted during the exploit and it gives you a better understanding of how the attack occurred and what can be done to prevent it from happening in the future.

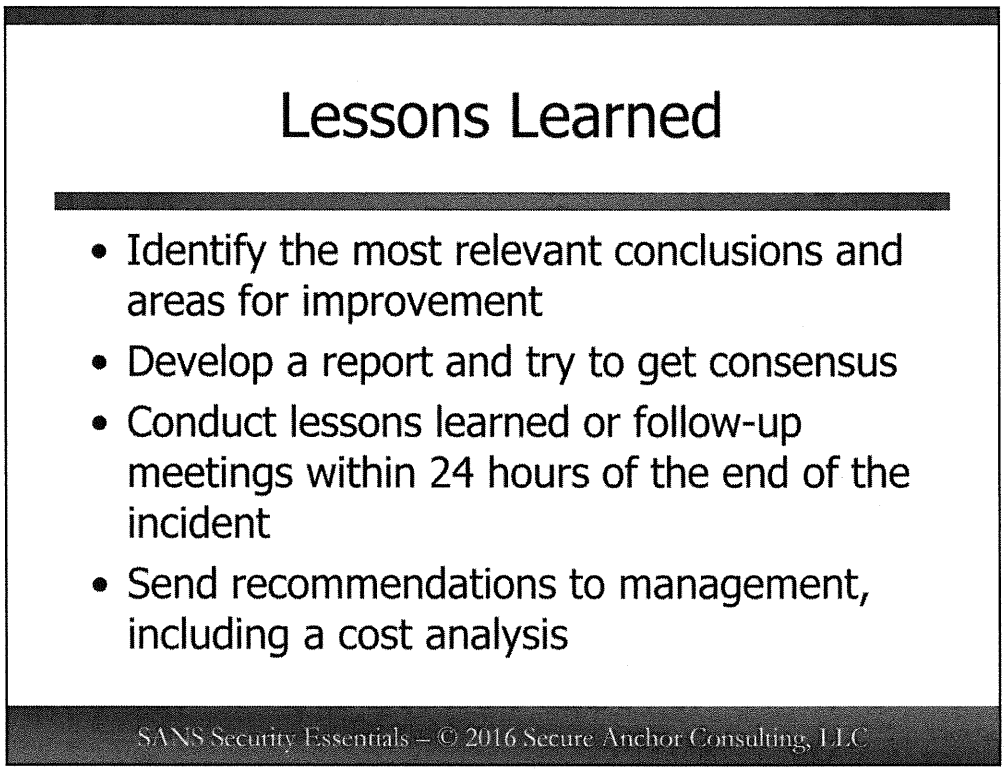
The two main options available when restoring a compromised system are:

- Installing the operating system (OS) and applications from scratch using the official and original media, adding the latest OS and application software updates (fixing the vulnerability exploited during the incident), and finally restoring the data from a backup.
- Restoring the system from a trusted backup and patching the system, at least fix the vulnerability involved in the incident. The trusted backup already contains the latest system and application data available.

Before the system can be brought back into production, the incident handler needs to validate the system along with the system administrator. Removing the vulnerability could have affected other functions of the system that are deemed critical by the business. Anything that breaks after the recovery is likely to be blamed on the incident handler, so every effort should be made to ensure the system is working as normal before turning it over

to the system administrator. In addition, the decision on when to put the system back into production has to be made by the system owner. The handler can give advice and be as helpful as possible, but, ultimately, the final decision of bringing a system back online rests in the hands of the system owner and/or administrator.

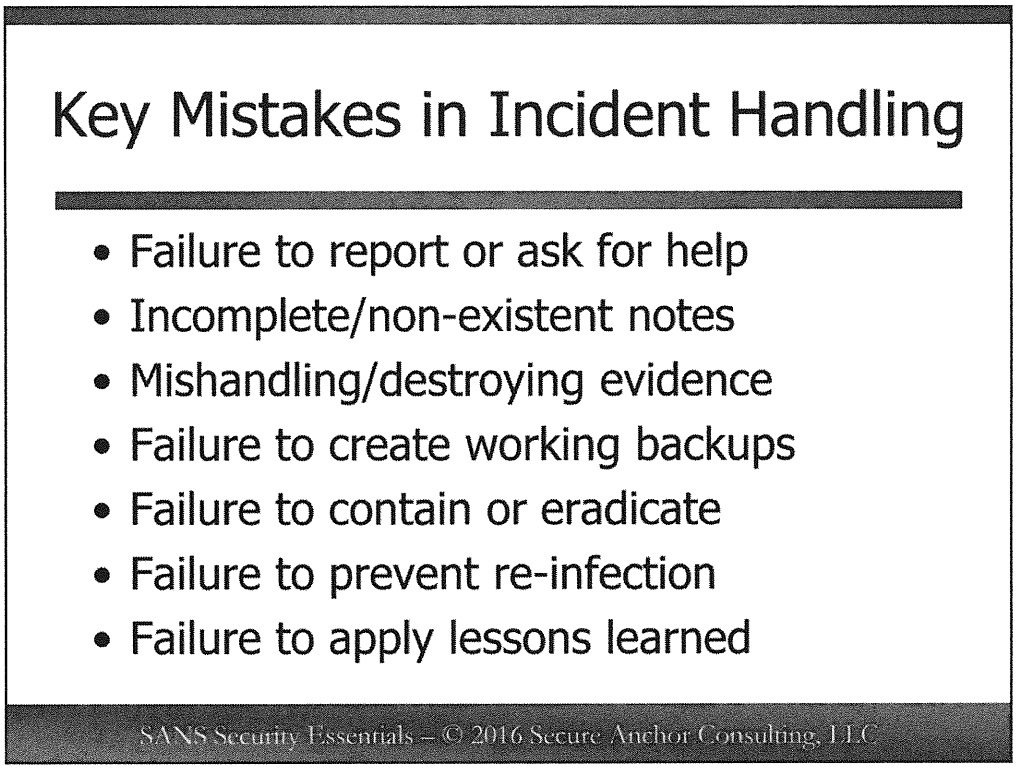
It should go without saying that if the eradication was not complete, or the infection vector was not closed off, there stands a chance of re-infection. Monitor the systems closely for the first few hours of operation to see whether anything crops up that could be attributed to the original incident. Monitoring will also help demonstrate to the organization the importance of an incident-handling team and the dedication of the team members to ensure the problem is taken care of correctly.



Lessons Learned

After the system has been restored and is back in operation, a report outlining the entire process should be drafted by the primary incident handler. It is very important to summarize the incident, identifying the most relevant conclusions obtained to aid in avoiding similar incidents in the future. The report should contain areas for improvement, both in the security infrastructure and in the incident-handling process itself. Additionally, the report must point out new security actions or projects identified during the incident and that must be implemented to increase the overall security of the IT environment.

The goal should be to get consensus with everyone involved. After the report has been drafted, all members of the incident-handling team should meet for a "lessons learned" overview. The goal of this meeting is to come up with a list of items that need to be included in the executive summary of the report. The executive summary should contain a brief synopsis of the entire incident, including the steps taken to recover and recommendations made.



Key Mistakes in Incident Handling

Conducting a follow-up meeting with all involved parties is never a fun task, but it is vital to making sure the organization understands what happened, why it happened, and what steps were taken to make sure it doesn't happen again. During every incident, mistakes occur and there is a tendency to place blame; however, the goal of the follow-up meeting should be to improve the process and learning from the mistakes.

Some key mistakes that are common in many organizations are listed here:

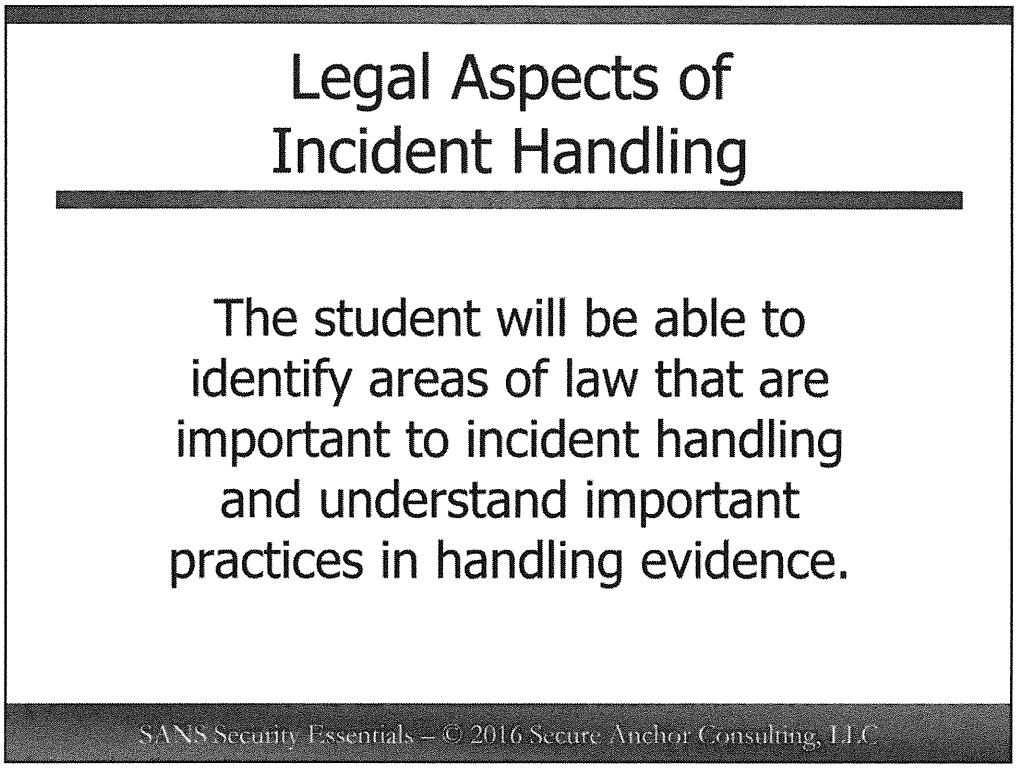
- Failure to report an incident or ask for help
- Incomplete or nonexistent notes
- Mishandling or destroying evidence
- Failure to create working backups
- Failure to contain or eradicate the incident
- Failure to prevent re-infection
- Failure to apply lessons learned

Putting the Steps Together

It might seem obvious that the six-step incident-handling process needs to be customized by each organization to take into account the various policies, network topologies, and other aspects of operation that might affect any of the phases outlined. Every incident is different and needs to be planned for accordingly by developing checklists, getting the required training, and assembling a team that best represents the technologies used by a particular organization:

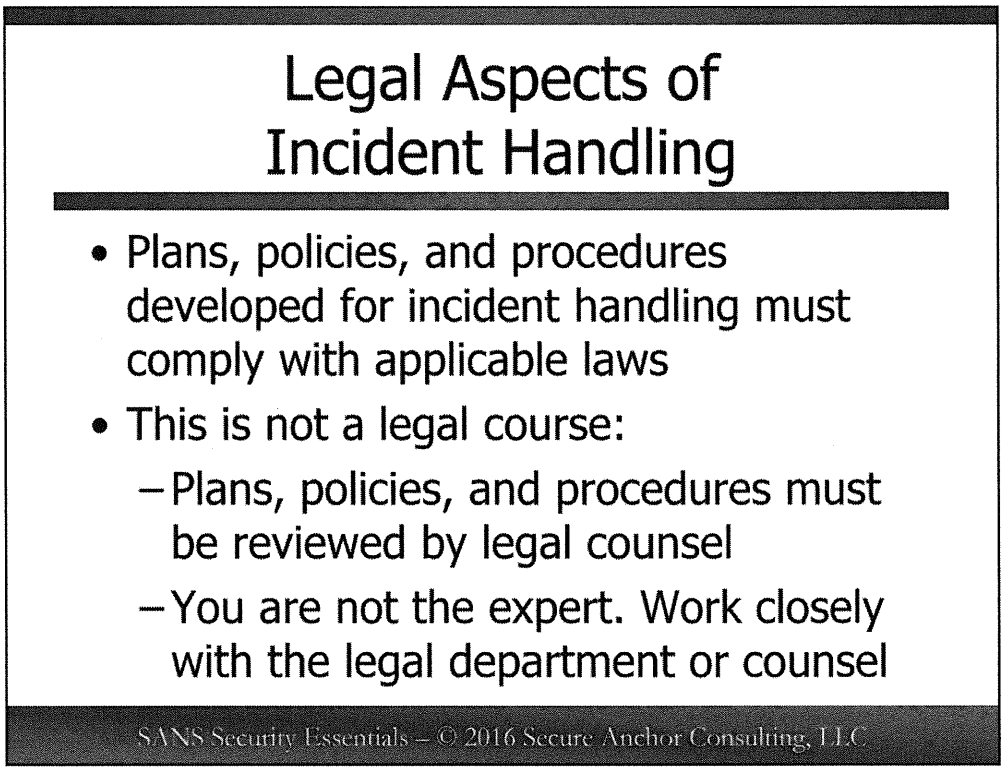
- **Preparation:** It is essential to plan for the eventuality of an incident. Remember, an incident will happen; it's simply a matter of time.
- **Identification:** The ability to distinguish between an event and incident. Staying current on potential vulnerabilities and exploits is a critical step in being able to identify an incident on your system.
- **Containment:** You must isolate the incident to prevent it from spreading or causing more damage to the organization. This might also involve gathering information to be used as evidence or making the decision to pull a system from the production network.
- Eradication: Eliminating the source or cause of the incident is an integral part of the incident-handling process.
- **Recovery:** Restoration of service and turning over the affected system back to the system owner or the administrator. The incident handler should take all precautions necessary to ensure the system is fully recovered before returning it back to the production network.
- **Lessons Learned:** Conducting a follow-up meeting after the incident is critical to understanding what happened and why and to ensure that the proper steps were taken to prevent similar incidents from occurring.

To improve the incident-handling capabilities of the organization, it is strongly recommended you practice the six-step process over the production environment before the real incidents occur. Several tasks, such as penetration tests or specifically designed attack simulations, help test the incident-handling capabilities and policy. On several occasions, these types of tasks have been useful in testing the readiness of an incident-handling team to act.



Legal Aspects of Incident Handling

This section intentionally left blank.



Legal Aspects of Incident Handling

Introduction
All incident-handling plans, policies, and procedures must comply with national, state and provincial laws, rules, and regulations. In Europe, for instance, although European Union (EU) Community Law has precedence in certain cases over national member state laws, member states have considerable latitude in how they adopt such laws. Adoption may vary from country to country. It is often challenging for large multi-national corporations to comply with these laws across the EU.

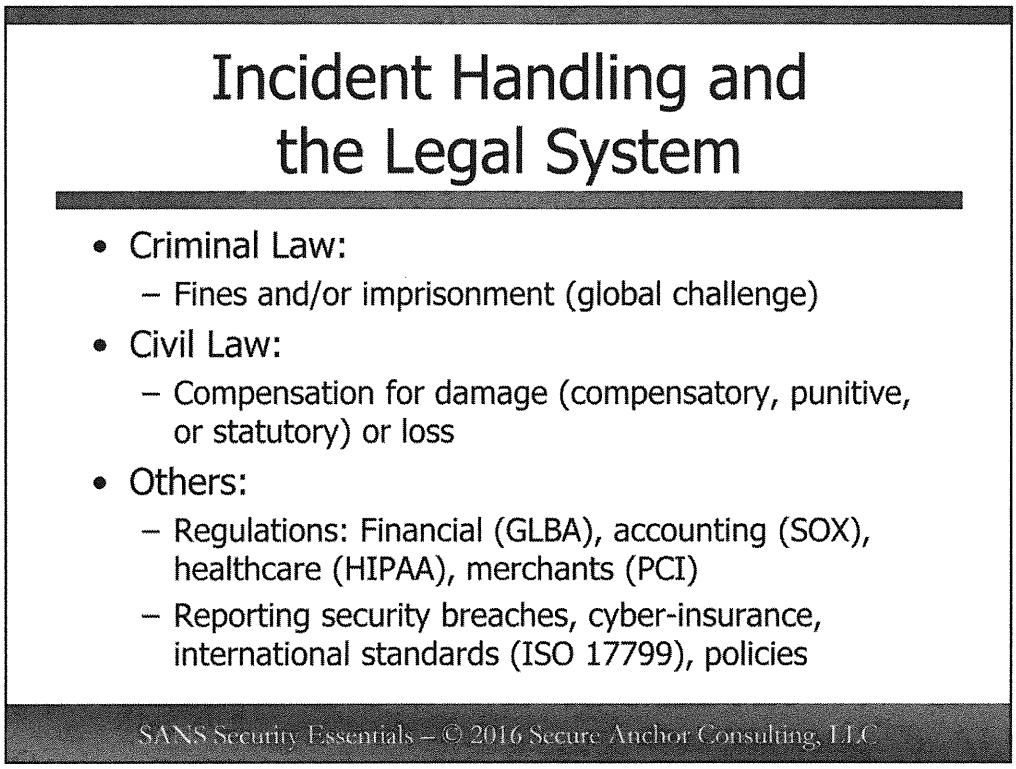
Legal Systems
The two dominant legal systems in the world are Common Law and Civil Law systems. The Common Law system evolved in England over several hundred years and is often referred to as *judge-made law*.

Common Law
Common law is supplemented by written statutes and legislation. It is the system in operation in the United States, most parts of Canada, Australia, Ireland, and the UK.

Civil Law
Civil law, on the other hand, comprises codified, or written laws, which are supplemented by additional written laws and legislation. The most famous "Code" is the Code Napoleon (the French Civil Code of 1804), named after its proud creator, Napoleon Bonaparte.

It has influenced the laws of Belgium, Luxembourg, Netherlands, and the old French colonies. Some countries and states (Louisiana, Scotland, and the Canadian province of Quebec) have "hybrid" systems. Some legal scholars believe that the differences between the two systems are eroding over time. There are, however, still significant differences, especially in the criminal law arena, that are best left to local lawyers to interpret.

Summary
Laws in both civil and common law systems are frequently revised and amended. Because of this, those responsible for legal action must work hard to stay current.



Incident Handling and the Legal System

Introduction
As you can imagine, the security professional needs to take many factors into account when reacting to an incident; for example, whether law enforcement should be advised, whether charges should be filed, or whether a criminal offense has been committed.

Criminal Law
Criminal law was designed to protect the public from conduct considered in conflict with certain societal norms (for example, assault, murder, rape, fraud, and more recently computer crime). Criminal law generally imposes fines and orders the confiscation of assets (for example, the proceeds of crime, or "drug money"), and/or may impose a period of imprisonment. In some countries, the death penalty may be imposed.

Certain acts may have both criminal and civil consequences. A drunk driver may be prosecuted for the crime of drunk driving and sued by the victim for damages for his/her injuries. Computer crime laws, such as the U.S. Computer Fraud & Abuse Act, may contain both civil and criminal law penalties.

Computer crime has proven to be challenging for global law enforcement agencies, because the crimes are often anonymous, hard to trace, and borderless. The criminals might reside in a jurisdiction with inadequate, if any, computer crime laws. As a result, it might be impossible to extradite them. Some computer crimes might even fall between the cracks. The law attempts, with limited success, to keep pace with evolving threats. For example, international treaties, such as the Cybercrime Convention, attempt to ensure that signatories have similar computer crime laws and that international cooperation is rendered more effective.

Civil Law
Civil law deals with adjudicating private disputes between parties, such as neighbors fighting over noise pollution. The Law of Torts is the area of the civil law that deals with many such disputes. A "tort" is simply "a civil wrong." The Law of Negligence forms an integral part of tort law. Generally speaking, in order to be held accountable for negligence, a party must owe "a duty of care" to the injured party; there must be a breach of that duty; and damage must follow as a result of the breach.

In the security arena, damage resulting from a security breach can be hard to prove; so it is important to document the cost of all remedial measures, including the time/number of personnel spent on such efforts.

Sometimes, in certain egregious cases, or where the law allows it, the damages awarded may be punitive in nature—more than is necessary to restore the injured party to the position it was in before the breach.

Recovery in Civil Law

In the event of a malware attack, a denial-of-service attack, or another attack that affects the availability of a system, or where sensitive or valuable information has been stolen, it is important to get legal advice to ascertain whether court orders can be obtained to try to trace and/or recover assets or get compensation from a defendant. Determined insiders might try to move stolen assets offshore. Involving legal counsel and law enforcement agencies in a timely manner might be of the essence in trying to recover them.

Regulations

After the Enron scandal and other such financial and accounting scandals, many governments have adopted tough new laws and regulations to try to prevent similar incidents occurring in the future. For example, the U.S. Sarbanes Oxley Act (SOX) is a legislation intended to reform the accounting practices, financial disclosures, and corporate governance of public companies. Certain regulated sectors, such as the pharmaceutical, healthcare, and financial services sectors, have always been heavily regulated around the world because there is a greater potential for harm to the public if something goes wrong. As an example, the Federal Drug Administration (FDA) in the U.S. regulates the drug companies in an effort to ensure that they only develop, market, and sell "safe" products to the public; the U.S. banking regulators, such as the FDIC and the OCC, are required to protect the public and the safety and soundness of the banking system as a whole. International regulators, such as the Bank of International Settlements (BIS), issue rules and guidance to member banks (for example, The Basel Accord) to protect consumers and global financial markets.

In certain regulated sectors, such as financial services and healthcare, statutes such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm Leach Bliley Act (GLBA) might compel industry participants to adopt security policies and procedures that include incident-handling and business continuity planning. HIPAA protects health insurance coverage, establishes national standards for electronic health care transactions, and addresses the security and privacy of health data. GLBA requires that financial institutions ensure the security and confidentiality of customer personal information against "reasonably foreseeable" internal or external threats.

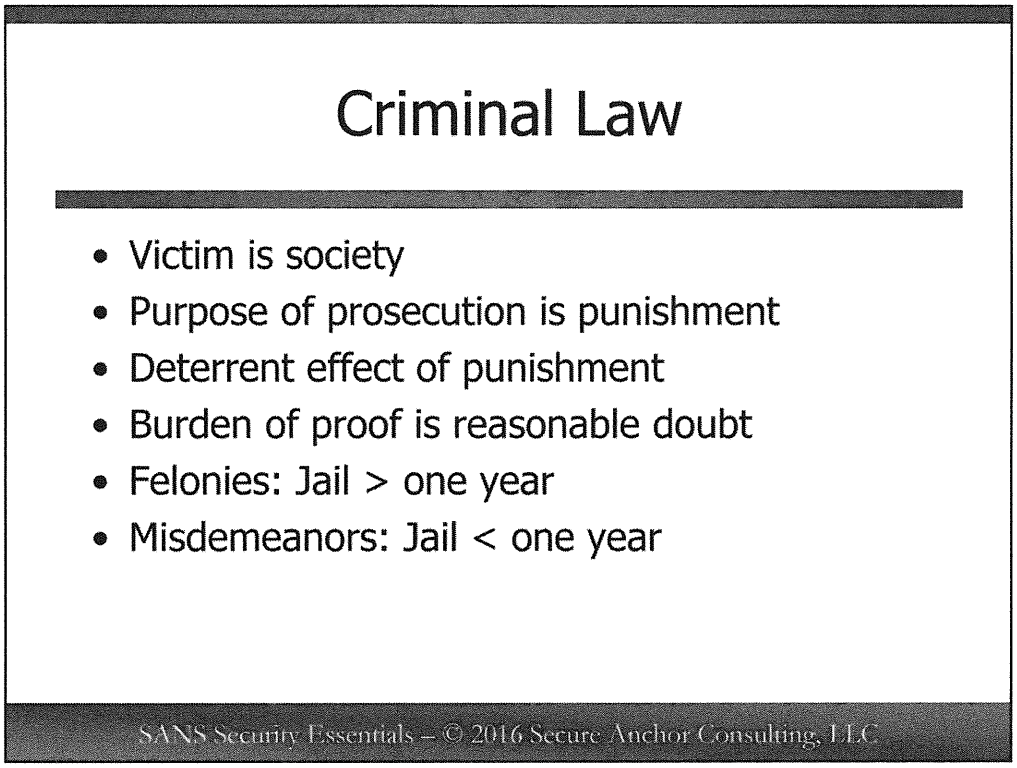
There are modern regulations affecting generic sectors, such as merchants dealing with credit card information. The Payment Card Industry (PCI) Data Security Standard is an industry regulation developed by VISA, MasterCard, and other bank card networks. It requires organizations that handle bank cards to conform to security standards and follow certain requirements for testing and reporting.

Reporting Security Breaches

Traditionally, senior management has been very reluctant to report security breaches for fear of negative publicity and other adverse consequences. However, certain laws in the United States, such as SB1386 in the state of California, mandate that security breaches be reported to consumers in defined circumstances, usually where the exposed or lost data was unencrypted. Under U.S. SEC rules, public companies are under an obligation to report to regulators if an event occurs that might impact the stock price.

Other Relevant Laws, Standards, and Policies

If competitors or foreign governments are implicated in an attack, counter-espionage laws might be relevant. Certain countries, such as Canada, Australia, and the EU member countries, have strong privacy laws that contain security-relevant provisions that must be respected, such as the "Ley Orgonica de Proteccion de Datos de Carocter Personal" (LOPD), Personal Data Privacy Protection Law, available in Spain. Investigations might also reveal illicit employee activity, such as the downloading and storage of illegal software, music, videos, or pornography on company property. Such activity might expose the company to liability and/or severe penalties. Hence, strong e-mail and computer usage polices are essential. All employees must be fully aware of what constitutes appropriate behavior and be aware of the consequences of non-compliance.



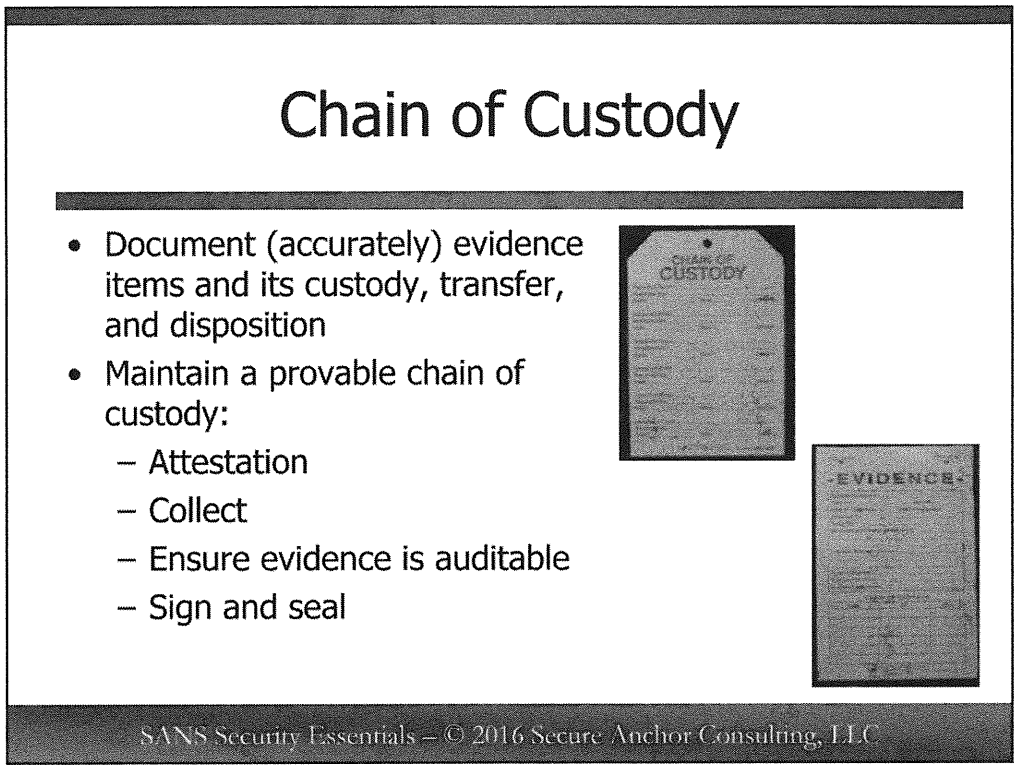
Criminal Law

There are two main categories of law: criminal and civil. With criminal law, the victim is society and to take criminal charges against someone, law enforcement must take the case. An individual or company cannot take criminal charges against someone. Criminal charges are the only laws in which someone can get jail time. With civil laws, you can get monetary restitution, but not jail time.

When dealing with law, there is a criteria that determines whether someone is guilty. With criminal law, the burden of proof says you have to prove beyond a reasonable doubt that someone committed a crime. Depending on the severity of the crime, there are different amounts of jail time one can get for a crime.

Civil Law (Tort Law)

We mentioned previously that there are two types of law: criminal and civil. With civil law, you do not need law enforcement involved to take action against an individual. However, with civil law, a person cannot get jail time. A person can be ordered to pay only monetary damages. Because law enforcement is selective about which "hacker" cases it takes, it is common for a company to take civil action against an attacker if the attacker is known and there is proof the attacker caused damages to the company. In civil cases, because there is no jail time, the cases are generally easier to prove and take less time in the court room.



Chain of Custody

Chain of custody is a concept in jurisprudence that applies to the handling of evidence and its integrity. It also refers to the document or paper trail showing the seizure, custody, control, storage, transfer, and analysis of physical and electronic evidence.

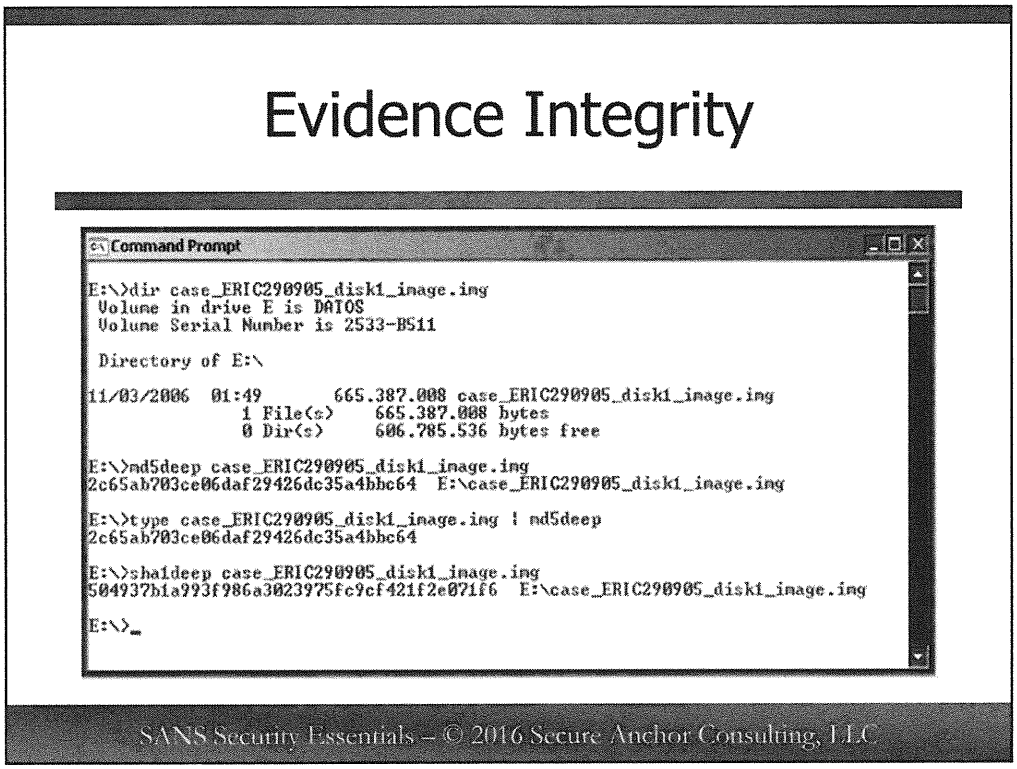
In a criminal trial, it is usually important to prove that the chain of custody has been respected. In other words, it will often be necessary to document the chain of custody for the evidence in question from the time it was seized to the time it is sealed and subsequently presented to the court. Unexplained gaps in the chain of custody can cause serious problems, because the defense might be able to argue that the integrity of the evidence cannot be assured during that time frame. It is prudent to allow law enforcement personnel do their jobs in managing the chain of custody.

When you are required to play a role, document all dates and time stamps on the items seized, and keep a record of serial numbers. Chain of custody is an important application of the Federal rules of evidence. The methods and procedures used can affect the admissibility of the evidence collected and, although this is not generally considered a problem, maintaining good procedures ensures that any evidence gathered will be admissible in a court of law.

The first step in maintaining chain of custody is to establish the basics of the situation: who, what, where, and when. Before you touch the computer, it is a good idea to write down where you are, describe the situation, and note all serial numbers of the machine(s) in question.

After the baseline has been established, the collection phase can begin. If at all possible, a binary backup of the information should be performed to prevent any further steps from possibly weakening your case. However, as previously stated, it is highly advisable that incident plans mandate that only experienced and trained individuals conduct the forensic component of an investigation—and/or that trained law enforcement personnel do so pursuant to a valid search warrant.

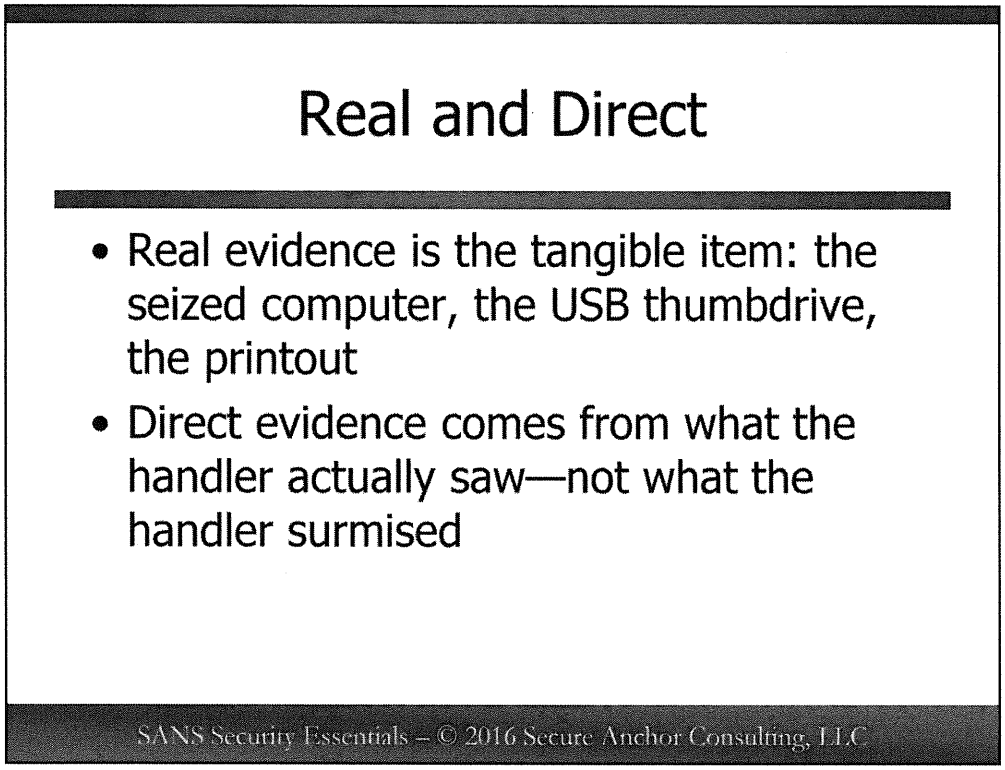
The final step to ensure a proper chain of custody is to sign and seal each piece of evidence as it is collected. If the evidence is transferred to another person, it is imperative to get that person to sign off on an itemized list of all the data collected and transferred.



Evidence Integrity

Trained forensic computer security professionals follow defined procedures that have been approved by the courts in their jurisdictions for preserving computer-based evidence. They generally take a mirror image or snapshot of the targeted media, often using commercially available forensic tools. Subjecting the image to a hashing algorithm, such as MD5, provides a value that represents the imaged data and its integrity. The idea is that if the imaged data subsequently changes in any respect, however minute (if its integrity is not fully preserved), the hash will not compute.

The process must be fully auditable, verifiable, and repeatable. Follow reasonable procedures when examining the evidence; keep good notes; and, if possible, run "script" or some other command-line history tool to demonstrate later the exact steps you took to assess the situation. The basic premise is that if you can repeat what you did during the investigation, then you are in decent shape going into court. Whenever possible, it is a good idea to run checksums to maintain the file integrity of the data being collected. MD5 is one of the most popular methods and can easily be used on both Windows and Unix platforms. To validate a copy, one could obtain the hash, store it in a safe location, possibly with a digital signature, and if necessary, copy the evidence file to external media. To validate the copy was accurate and nothing has changed, run the hash on the external media and compare it to the well-known good hash.

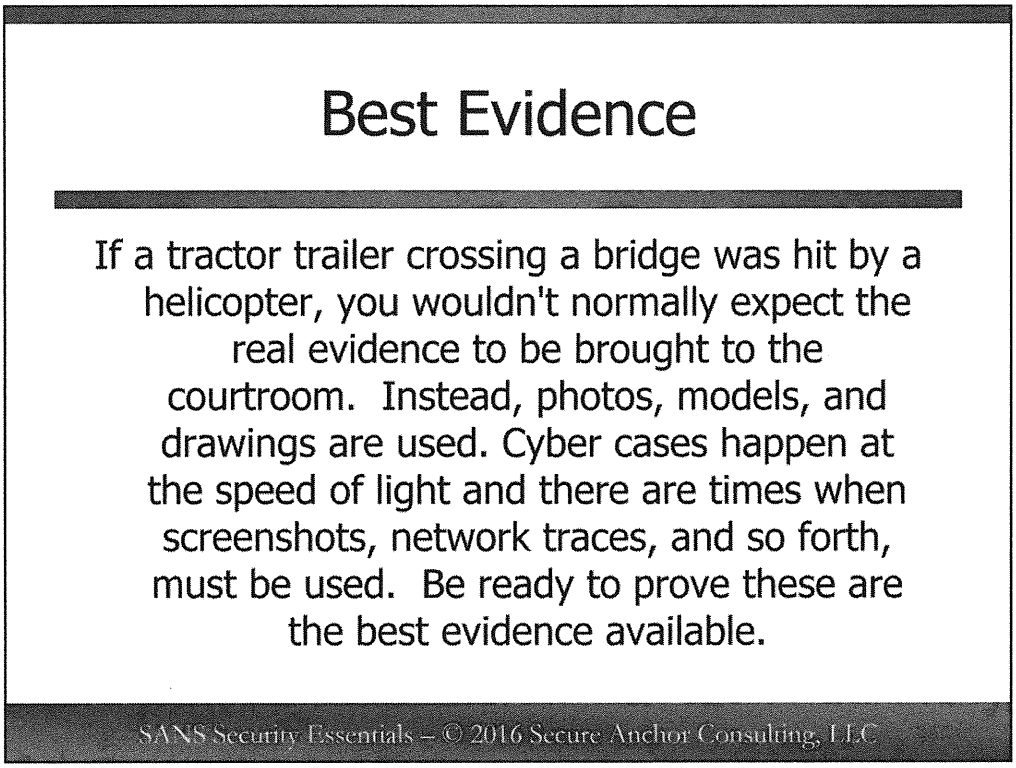


Real and Direct

When it comes to presenting evidence in court, you should present the most compelling or "best evidence" available to you. If you are fortunate enough to have real and direct evidence of the facts in dispute, your chances of success are much better than if you must rely entirely on circumstantial evidence or third-party testimony. Real evidence is often the most compelling type of evidence. It is usually a tangible object, such as the blood-stained murder weapon, evidence on a seized computer disc, or other documentary evidence that in and of itself tends to confirm the facts in issue.

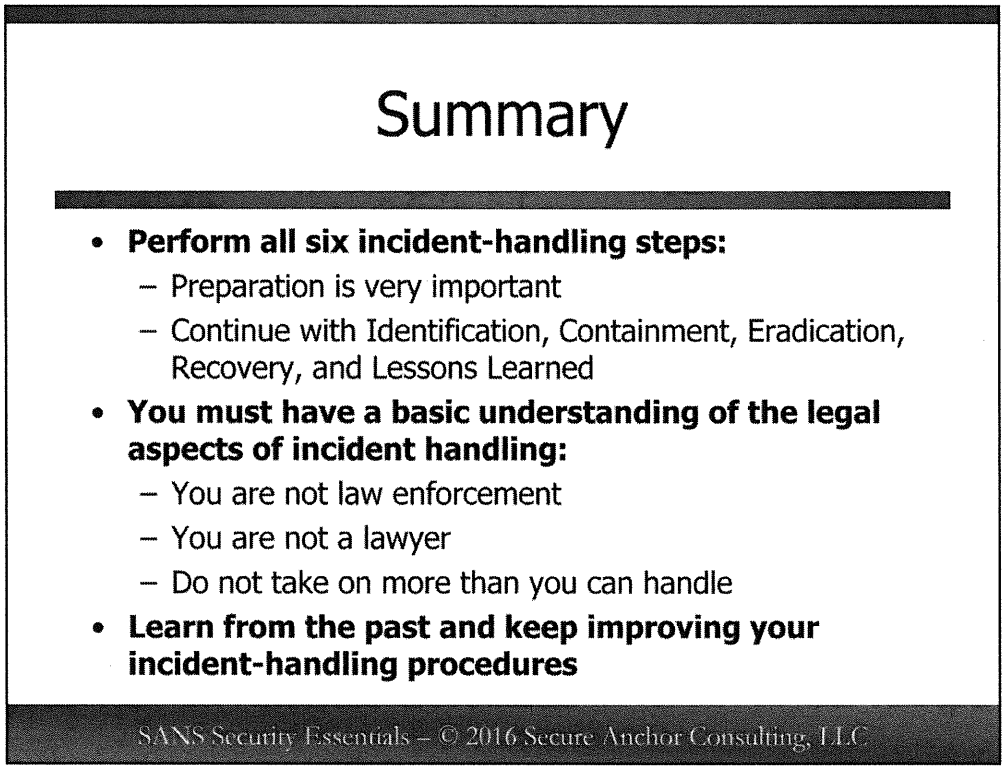
Direct evidence is also a strong form of evidence. It usually refers to evidence gathered from an eye witness or the person who watched or logged an incident as it occurred, not from someone who merely speculates as to what occurred during an attack. In cyber crimes, it can be relatively easy to demonstrate the what, where, and when of a case. It can, however, be difficult to prove the "who" and "why" behind the attack. A person might claim someone else used her password at the time of the attack and so forth. This might be a good reason to adopt a watch-and-learn approach to incident handling. Such an approach allows you to build your case rather than speculate on who the perpetrator might have been during the time of the attack.

The final type of evidence we need to discuss is hearsay or, as it is sometimes called, "third-party evidence." This is evidence that is the opposite of direct evidence. It is one party's testimony to what another party said or did. There are many exceptions to this ancient rule, which has been all but abandoned as outmoded in many jurisdictions. Under the U.S. Federal Rules of Evidence and other statutes, business records, for instance, are expressly admissible under an exception to the hearsay rule.



Best Evidence

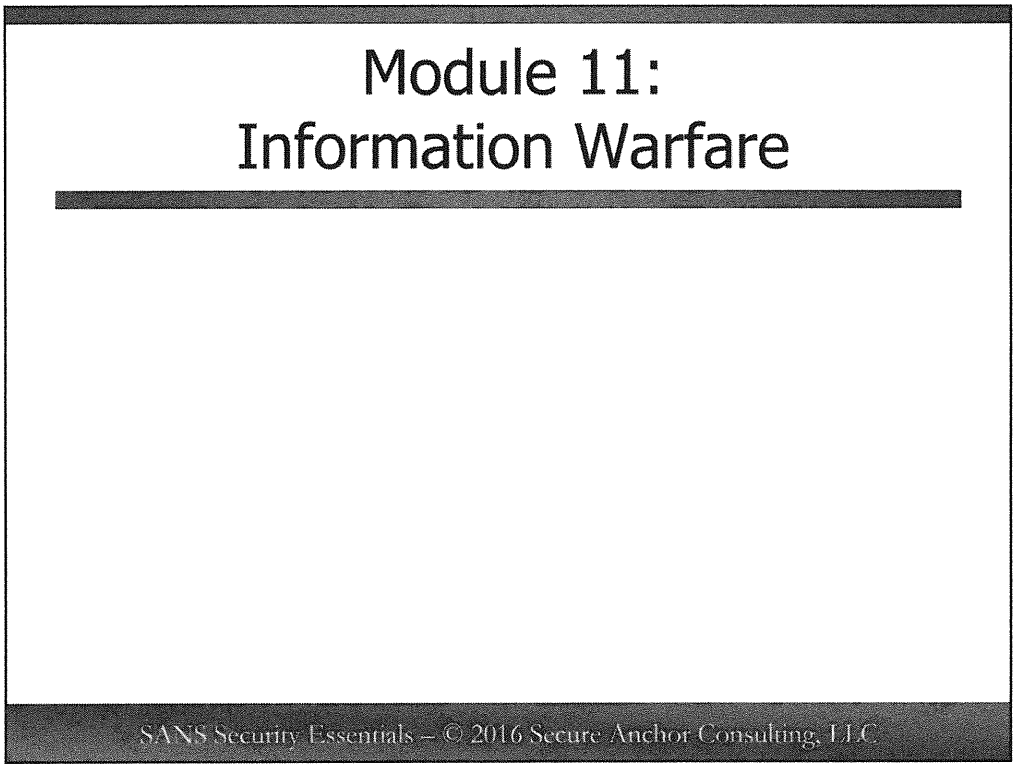
As stated, try to put your best foot forward in presenting evidence at trial. The "best evidence rule" in common law systems usually refers to a requirement to produce the original of a piece of evidence rather than a mere copy. In the context of computer-based evidence, the concept of originality is clearly challenging. However, most legal systems provide for special recognition of computer-based evidence. Printouts and other forms of output, like screenshots, are used to accurately reflect the data in question. Such evidence is often permissible. Therefore, in many cases, a properly constituted snapshot or mirror image of the data in question constitutes the best evidence available and should be admissible. It can be useful to have screenshots of the entire incident-handling process as a visual aid for the court, and in some cases to show context.



Summary

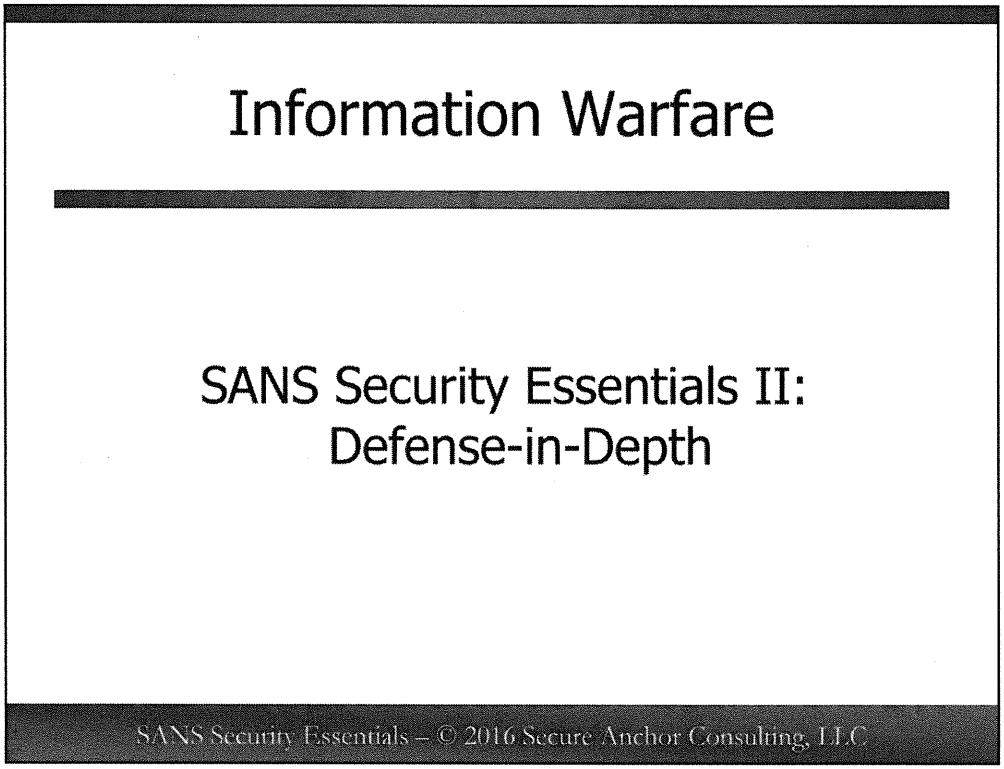
Incident handling can be extremely difficult, and the opportunity to make a mistake that could jeopardize an investigation is decreased if planning and preparation are taken into consideration before an actual attack occurs. Employing a six-step process that covers preparation, identification, containment, eradication, recovery, and lessons learned will aid in creating an incident-handling team that is capable of reacting quickly and accurately to any attack that might occur during its tenure. There are several laws that pertain to incident handling, and the organization must keep these laws, especially the Computer Fraud and Abuse Act, in mind when developing incident-handling policy and procedures.

Evidence collected must satisfy the minimum requirements of being able to prove what, where, why, and, if possible, who conducted the attack. Maintaining a chain of custody is considered crucial, and having pre-defined checklists and deploying a standard of sealing and signing evidence will help ensure evidence is not corrupted during the course of the investigation. Mistakes will happen, but when they are made, it is important to learn from them and change your plans accordingly, so the mistake does not happen again. Finally, being part of an incident-handling team is a high-pressure job where mistakes can be costly. Being able to respond quickly but accurately is considered vital, and those who are willing to make that commitment should be rewarded accordingly.



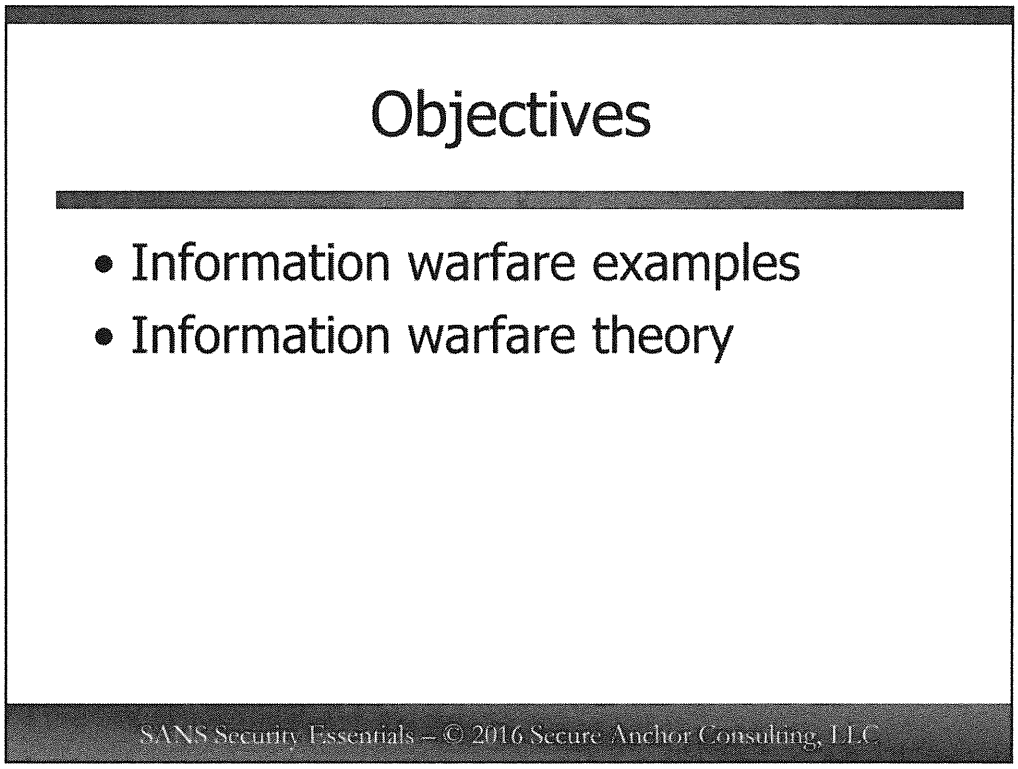
Module 11: Information Warfare

This section intentionally left blank.



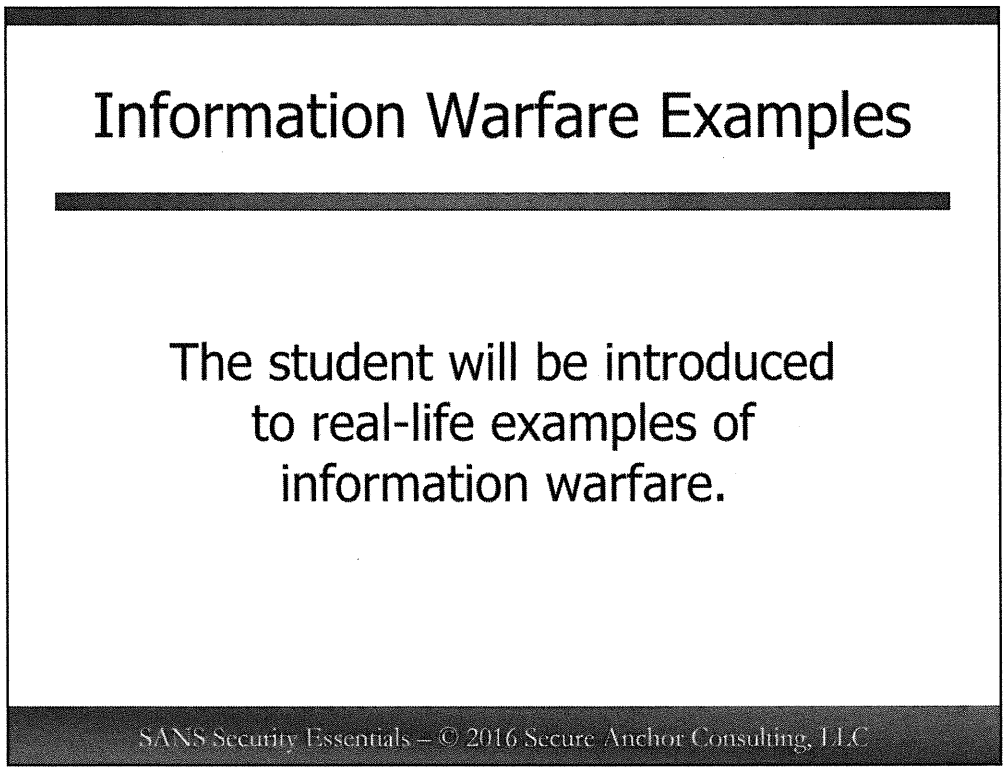
Introduction—Information Warfare

This section intentionally left blank.



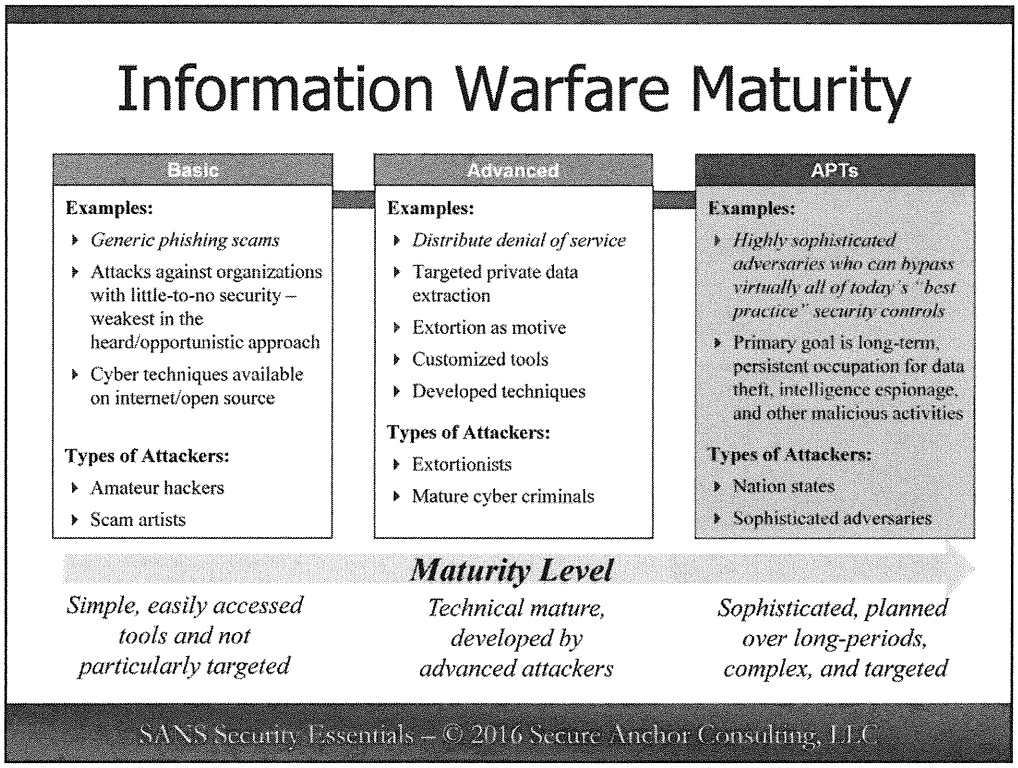
Objectives

Information warfare is competition between offensive and defensive players on a game board of information resources including computers and networks. In this chapter, we discuss a range of techniques from information warfare (IW) applicable to business, including essential tools of information warfare, perception management, malicious code, and predictable response. By the end of the chapter, we hope you realize that there is not a lot of difference between business and information warfare.

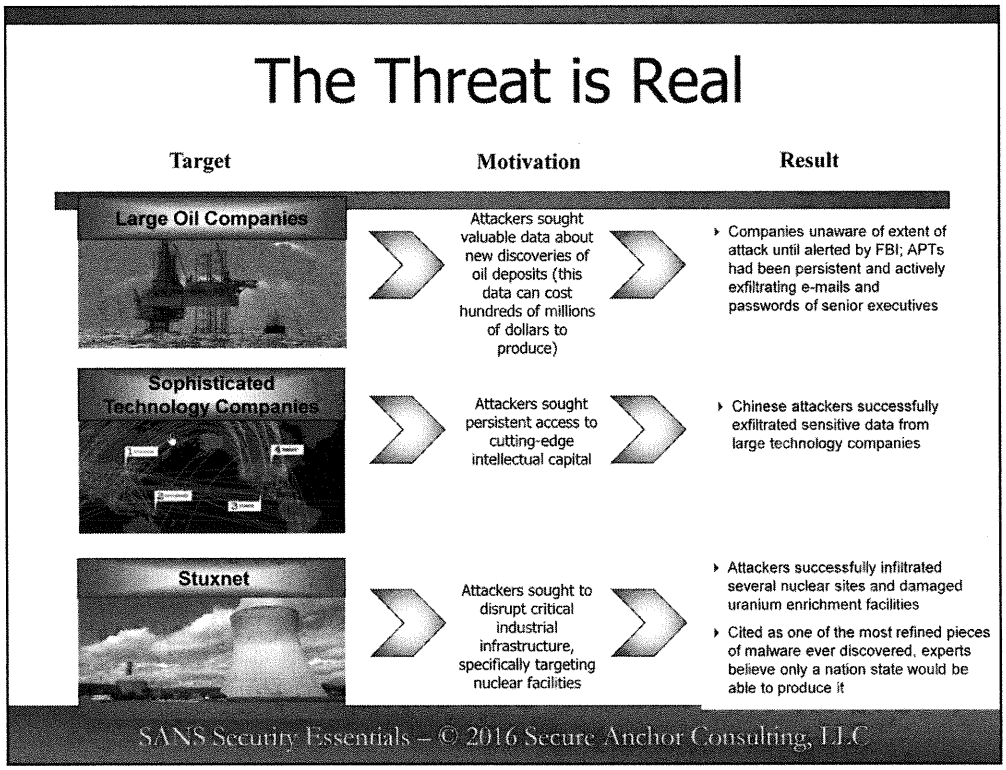


Information Warfare Examples

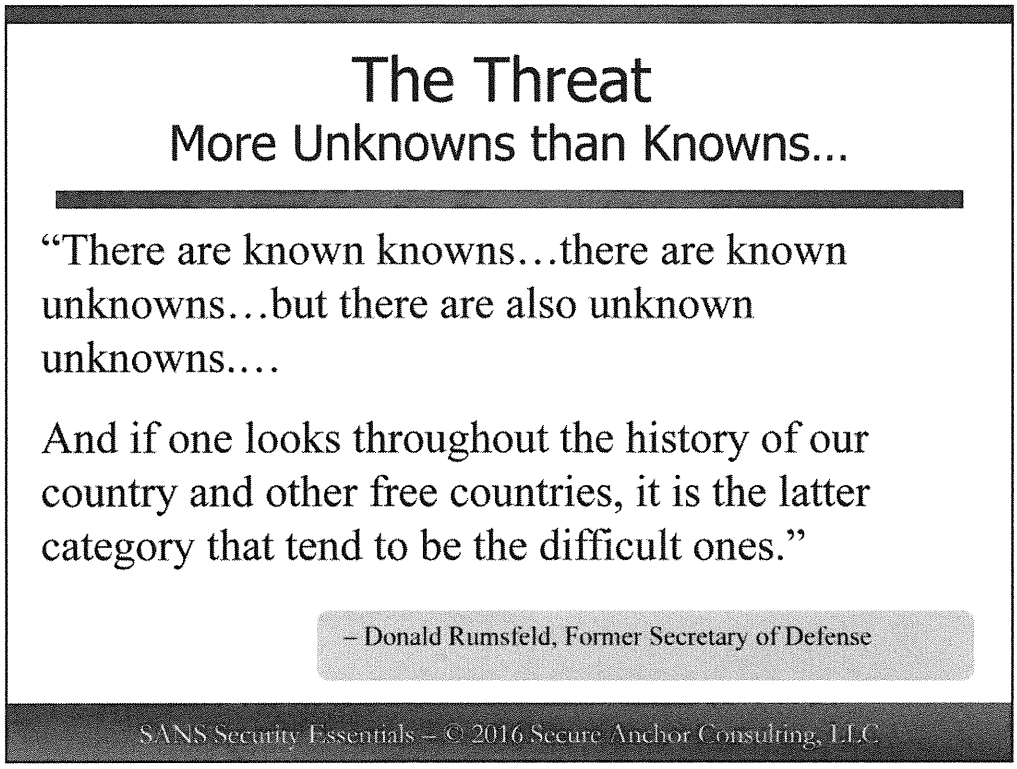
This section intentionally left blank.



Information warfare is always changing and adapting. Today's adversary is persistent, which means he will continue to keep attacking an organization until he is successful. Therefore, understanding how the adversary is changing is critical to building a secure organization.

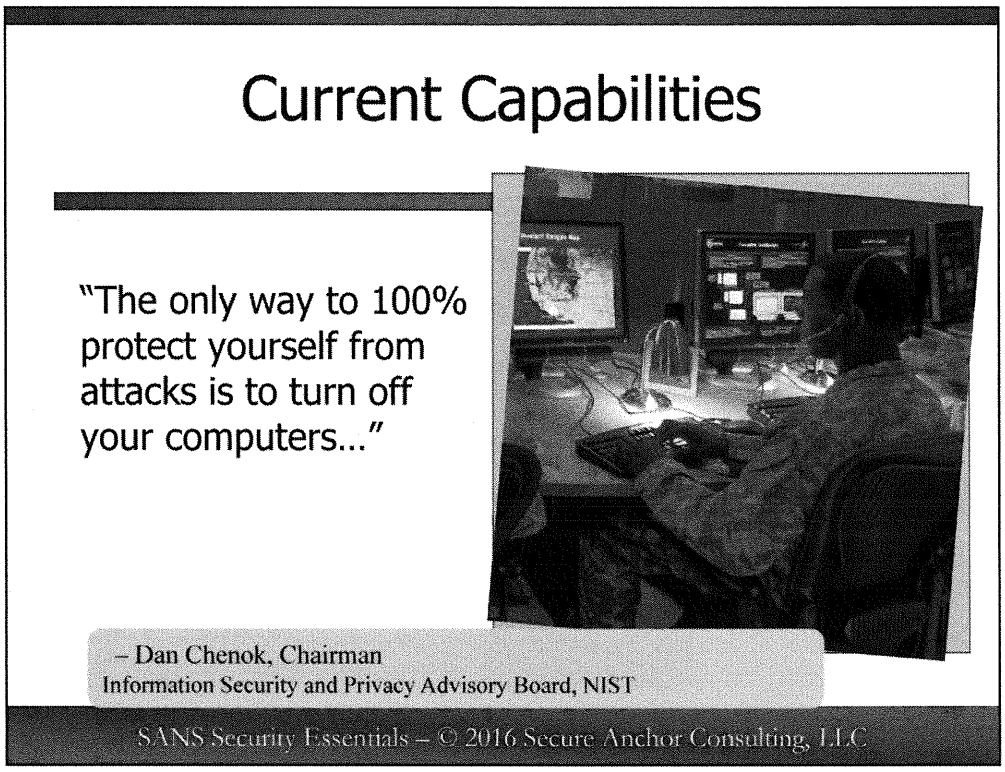


Cyber attacks are not science fiction or hypothetical examples meant to scare people; they are real. The threats are actually breaking in and causing harm, and we have real data to show how damaging the adversary can be.



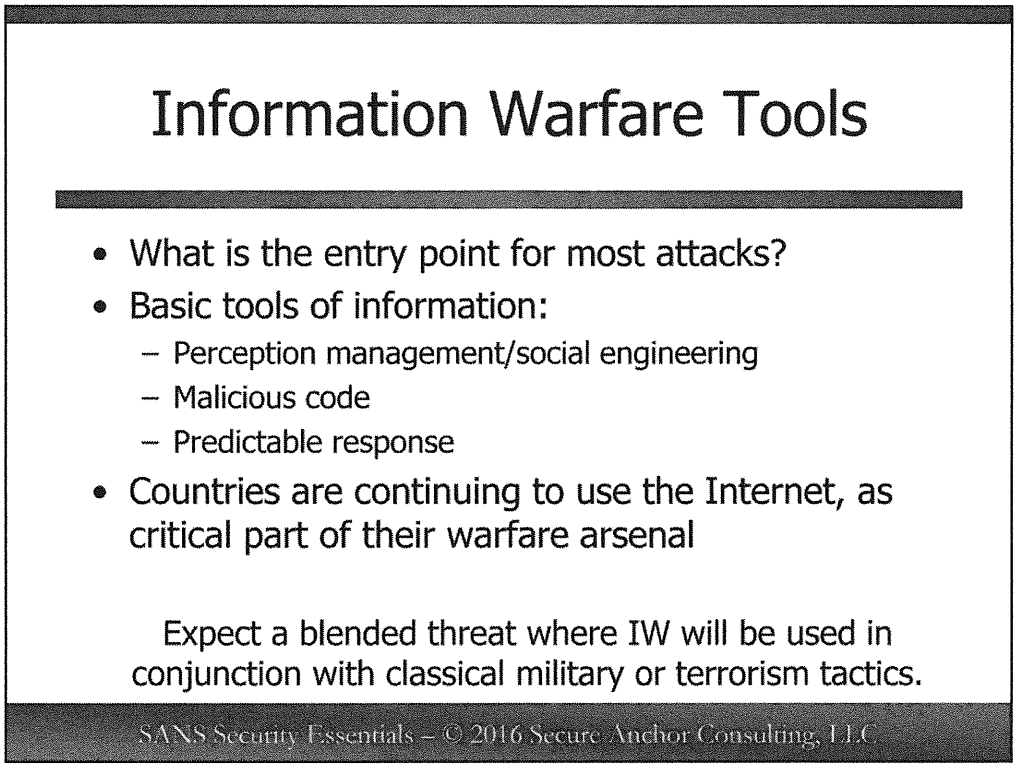
The Threat More Unknowns than Knowns…

One of the things that makes information warfare so difficult is the high number of unknowns. One of the key tenets of cyber security is "know thy system." If you do not know what the threats and/or vulnerabilities are, it is hard to calculate the risk.



Current Capabilities

It is important to remember that every organization and computer is a potential target. After a system is plugged into a network, there is the potential for compromise.



Information Warfare Tools

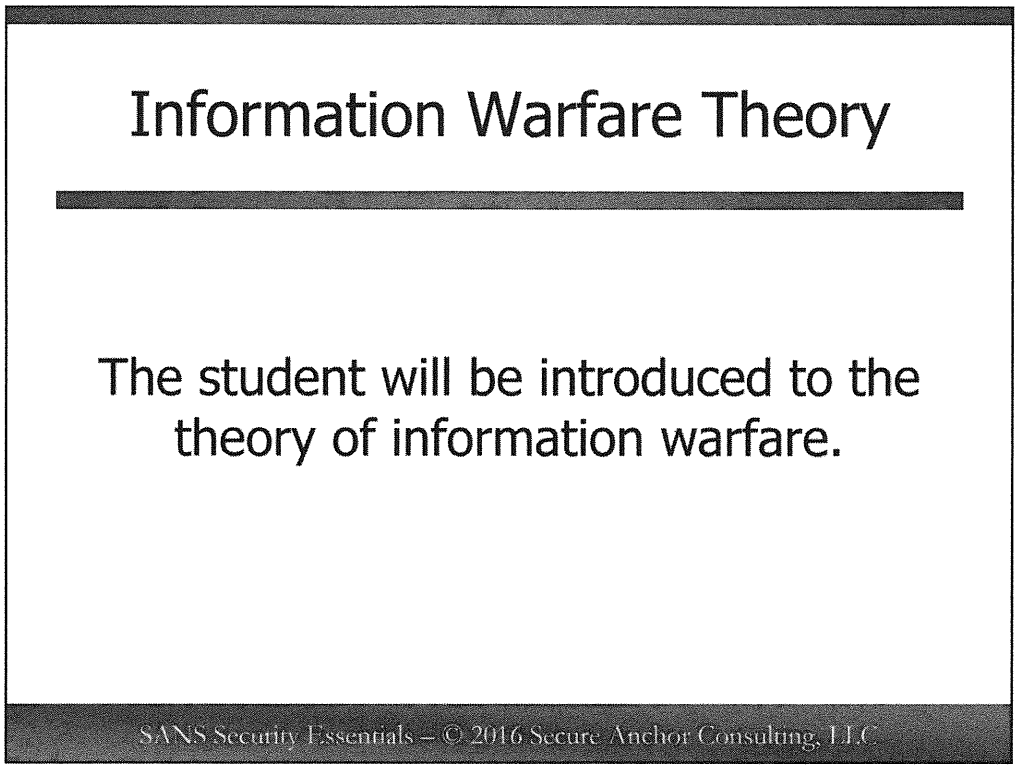
Three basic tools of information warfare are perception management, malicious code, and predictable response. They can be used separately or together. Two of the three, perception management and predictable response, are core techniques of business competition. In information warfare, the focus shifts to technological implementations of these techniques.

As we develop greater skills in perception management, use of malware, and predicable response, the infrastructure to use these tools is increasing. Industry experts estimate 60% of all spam is sent from zombie PCs whose owners have no idea their PCs are being used for such purposes. That implies a strong command and control network as well as a large Army of foot soldier desktop computers.

The focus for information warfare over the past decade seems to be, at its heart, economic. Countries like France have vigorously pursued intelligence to help their businesses. Perhaps you remember the widely told story of microphones in Air France jets to listen to the discussion of business travelers. There really isn't such a thing as a neutral country when it comes to cyberwar. Information warfare operations are as likely to travel through neutral countries as any others before reaching the belligerent target. User awareness training can help with both perception management and predictable response. Standard defense-in-depth approaches can reduce the attack surface for malicious code.

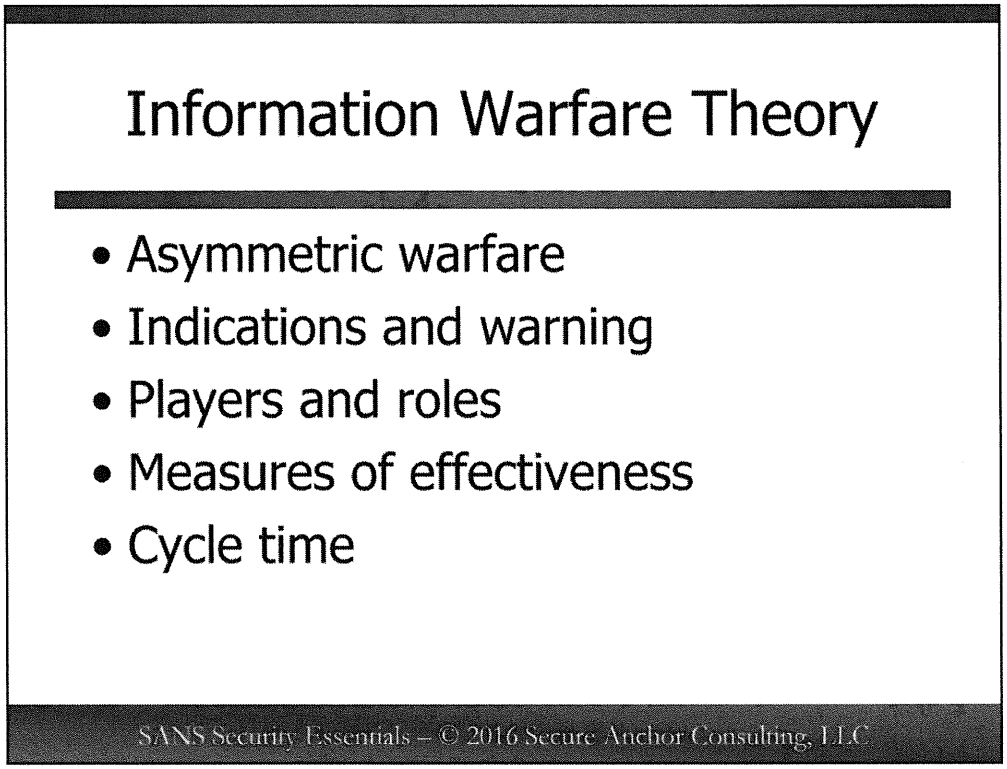
Offshore Coding and Software Engineering

Even though the trend to outsource is going strong, it appears that the warnings are finally being heard. Gartner stated in a recent press release the following: "The top two inhibitors to IT outsourcing focused on data security or privacy issues and concerns around the potentially high costs of outsourcing." Only time will tell what the future holds, but there is evidence of a shift in the culture and understanding of the significance of the threat to outsourcing. When Terrill Maynard published his article, both he and Michael Vatis were met with mixed emotions from the security community. Some understood and supported their concerns with the outsourcing that was taking place whereas others laughed at the whole idea. It seems the winds of change have begun to blow when it comes to understanding the security issues involved with who has access to source code.



Information Warfare Theory

This section intentionally left blank.



Information Warfare Theory

One resource used for the next section of the information warfare module is *Information Warfare and Security*, by Dorothy Denning. Ms. Denning has written the best schema for information warfare we have seen, and we use this as the basis to explain what we see in practice. The book is highly recommended for anyone involved in information operations or responsible for security for any Internet or e-commerce-based business.

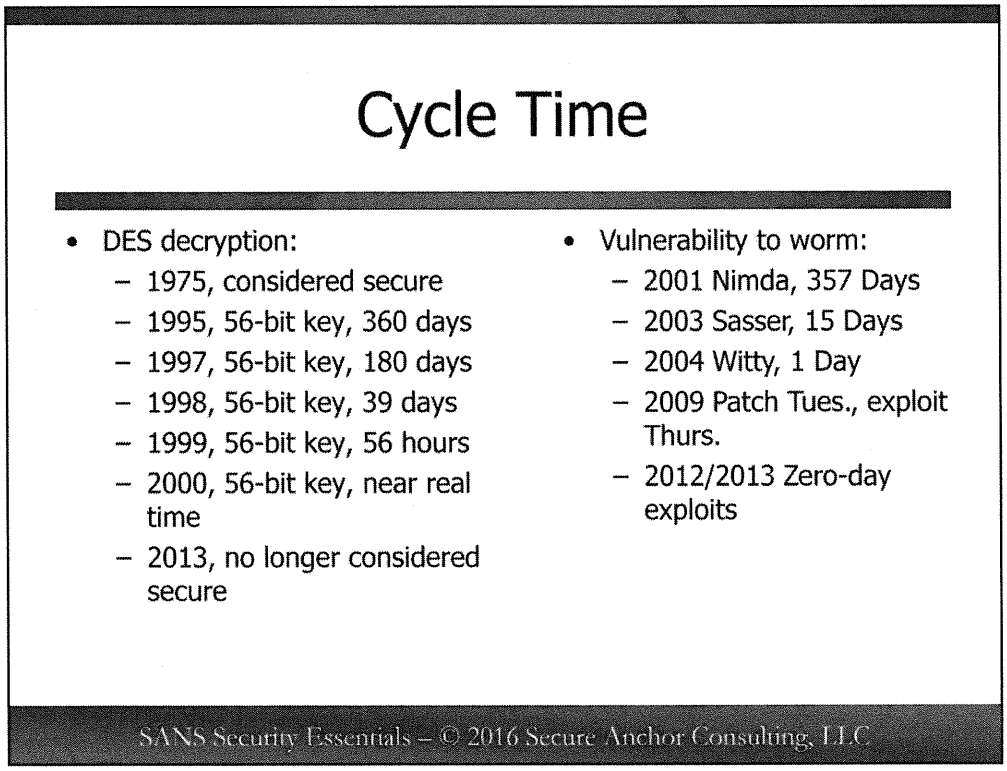
The most important concept to internalize as either the attacker or defender is *asymmetry* that information warfare is not a zero-sum game. If we add up the wins and losses in a game, treating losses as negatives, wins as corresponding positives, and the sum is zero, then we have a zero-sum game. This is how we open our discussion on theory. The competitive mind rails against the zero-sum game, and intense competition is a hallmark of business, economic warfare, and information warfare. The landmark book on this observation is John Macdonald's *Strategy in Poker, Business and War*, published in 1950.

Even though poker is zero-sum, do poker players aim at the end of the evening to each walk home with exactly what they brought to the table, no richer, no poorer? No chance—even though what one wins another loses, they are each focused on optimizing their winnings. What is the most basic technique they use? They attempt to avoid predictable responses, using techniques like bluffing to help them improve the odds that the game will favor their position.

In information warfare, we look for asymmetry to avoid zero-sum gamesmanship. Business and warfare are non-zero-sum, not closed systems; there isn't a fixed pie, and anything is possible.

Asymmetry is not the only approach – we also discuss cycle time to achieve non-zero-sum results.

Asymmetry essentially is where a fairly small investment or input has a very large effect. There are times when this is built into the protocol level of the Internet. As an example, the smurf attack takes advantage of ICMP being allowed to operate as a broadcast protocol. Of course, you have already been exposed to smurf, so this is a review. So let's focus on the asymmetry. To implement a smurf attack, the attacker spoofs the source IP address of the potential victim. He sends the packet to a smurf amplifier site, a place on the Internet that allows broadcast ICMP echo requests to the internal site from the Internet. Then the hosts that receive the broadcasted ICMP echo request answer back with an echo reply. For a Class B-sized network, one spoofed echo request might elicit 65,535 echo replies to the victim computer—now that is asymmetry! The www.netscantools.com/ site lists known smurf amplifier sites.

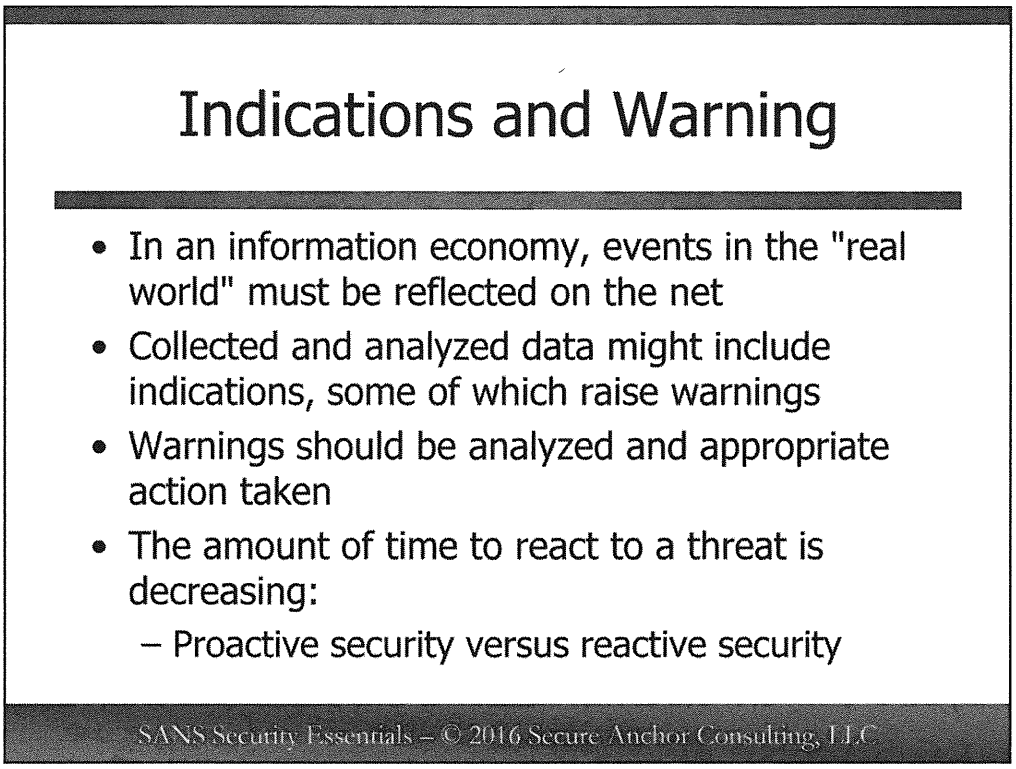


Cycle Time

For years, DES (data encryption standard) was the standard for most encryption, and this included financial transactions. Its impending demise became evident as early as 1998, though triple-DES was released as a partial workaround. In the cryptography section of this course, you look at DES in detail, but the basic problem was a short-key length. This cipher system has a fixed key length, 64 bits, but 8 of those bits are parity, so the effective key space is only 56 bits. If computing power doubles every 18 months, as Moore's law claims, then once the 56-bit key fell, it was game over—and every 18 months, it would be game over in half the time.

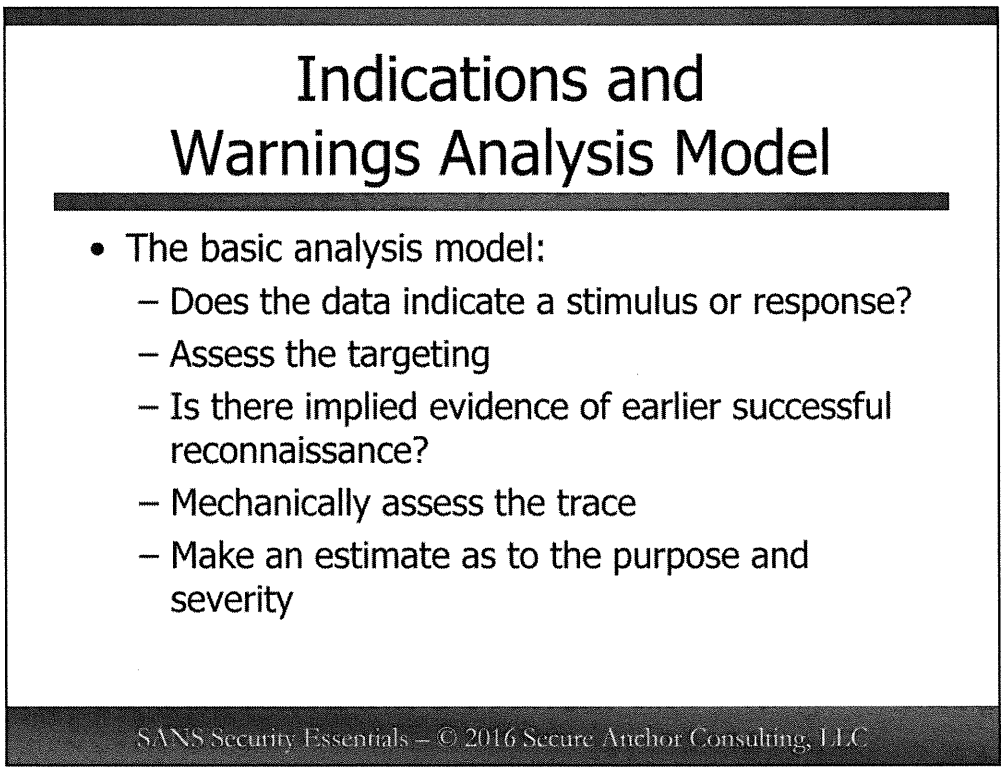
Well-funded groups probably have been able to read DES-encrypted notes with no more than a couple days delay for several years and certainly are in the realm of real time today. By decreasing the cycle time for decryption, this drastically lowers the value of information resources to the financial community that has a lot of DES hardware in place.

Another example of cycle time is the decreasing amount of time between a vulnerability announcement, patch availability, and the release of a worm taking advantage of the vulnerability.



Indications and Warnings

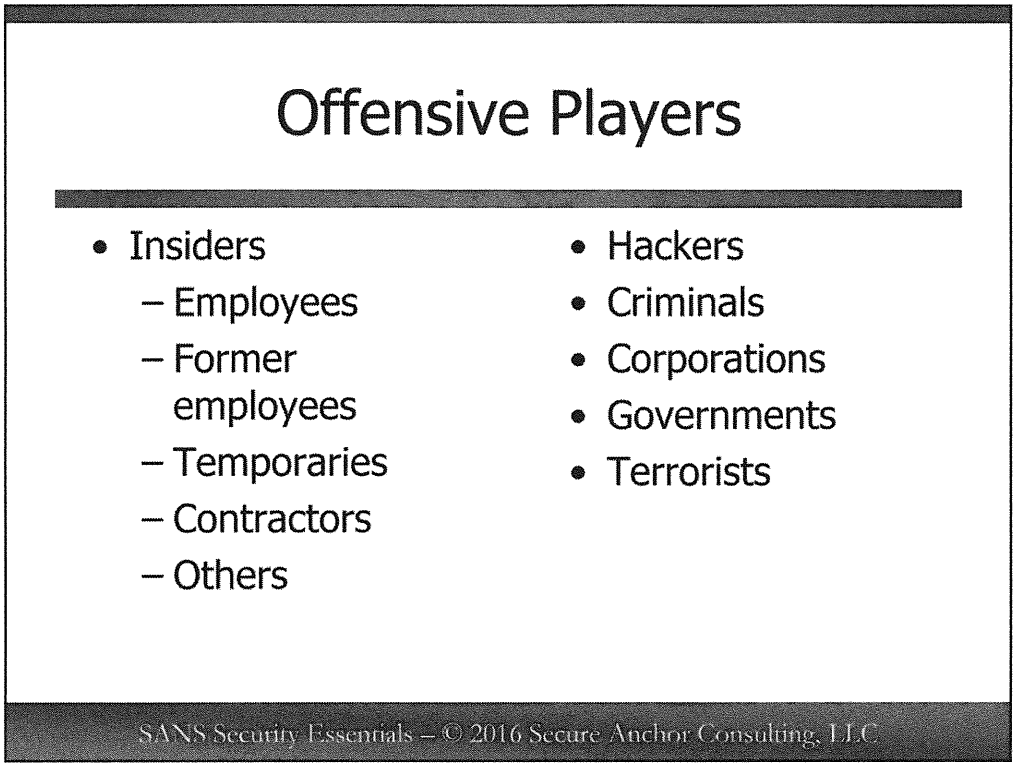
Indications and Warnings is a military analysis tool. Reconnaissance might be an indication because successful reconnaissance often is followed by the attack. Consider the following sequence: A Windows computer has a vulnerability for privilege escalation, but it is not normally Internet-facing. You might first see a port scan; Windows systems would have particular ports open, such as 137 and 139 TCP and UDP. The port scan is an indication, but you might not want to raise a warning just yet, because port scans happen all the time. Perhaps you next detect a null session. A *null session* is an anonymous connection to the Windows box that can be used to look for account and other system information. At this point, you might raise a warning and alert the security officer or take other action. In the meantime, the attacker might have moved on to trying to use the account information to do password guessing. As you see, with these additional indications, there is very little doubt at this point that the Windows computer is under attack.



Indications and Warnings Analysis Model

The basic model for assessing collected data is as follows:

- **Does the data indicate a stimulus or response?** To interpret data correctly, we must be able to make this determination. Echo requests and SYN packets are examples of stimulus packets; Echo replies and SYN/ACK packets would be responses.
- **Assess the targeting.** Are they probing everyone or focusing on our system?
- **Is there implied evidence of earlier successful reconnaissance?** Worms notwithstanding, the more accurate and focused the targeting, the higher the threat level.
- **Mechanically assess the trace.** Find out what is going on at the network and system level.
- **Make an estimate as to the purpose and severity.** At this point, you can decide whether to raise a warning or, if necessary, begin battle damage assessments.

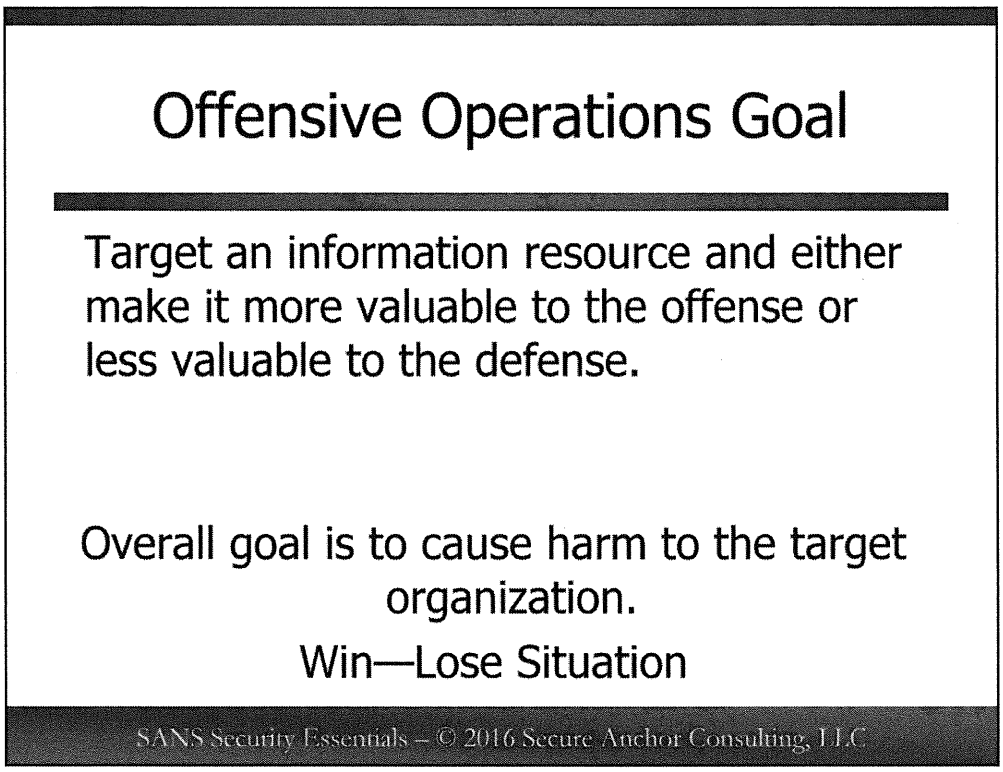


Offensive Players

Hackers are the most visible offensive players in terms of the consciousness of the public and press. An ICSA (http://www.icsalabs.com/) yearly report claimed that insider attacks account for 90% of the financial damage that occurs. However, insider attacks are on the rise.

Whether hackers or insider fraud, the criminal world is fully engaged in the Internet and in information operations. Your organization's FTP server might be a drop box for illicit materials completely without your knowledge. Your mail server might be forwarding the attacker's e-mail without even knowing it. Your finances might become co-mingled with those of the underground as part of a money-laundering scheme.

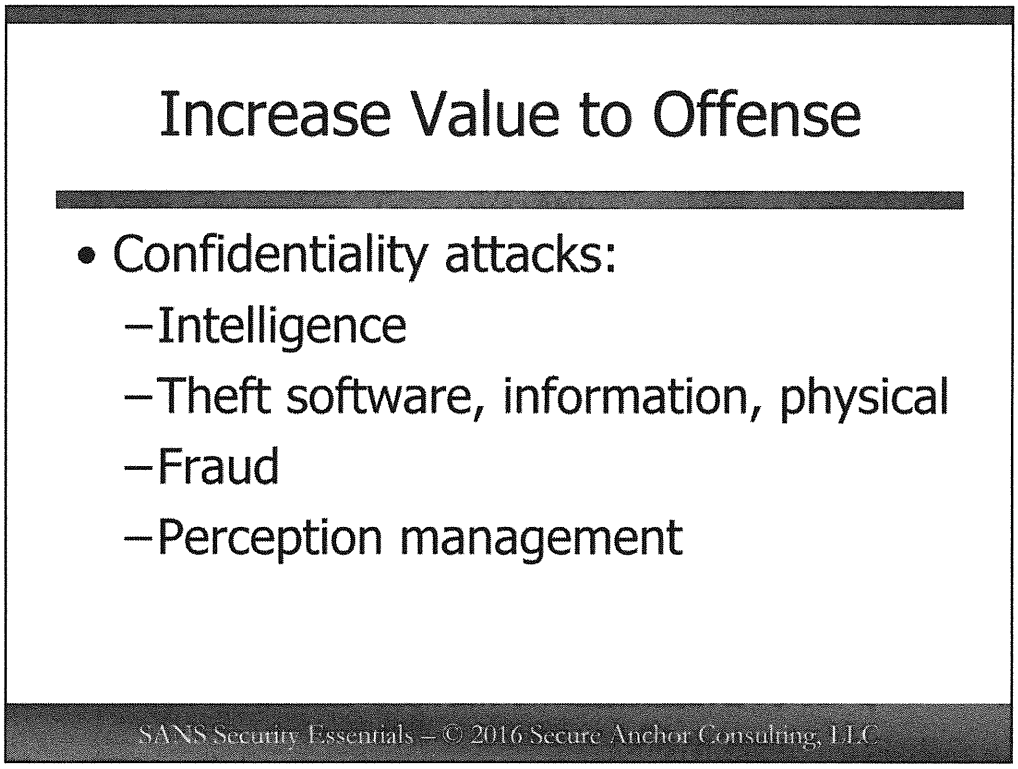
The important thing to consider about the previous issues is that a remarkable number of organizations do not realize they are being used or stolen from. Whatever crime your organization is or is not experiencing, you can be certain that due to the intense competition from globalization, unless your organization is the laughingstock of the industry, it is experiencing competitive intelligence gathering.



Offensive Operations Goal

Although poker fits nicely into the notion of zero-sum—whatever one poker player wins, another (or the rest) loses the exact same amount—this doesn't apply so neatly to business, economic warfare, and information warfare. As we have already pointed out, intense competition drives us to search for non-zero-sum solutions. In the book, *NONZERO: The Logic of Human Destiny*, the author Robert Wright leads the reader to believe that all parties will seek win-win situations. This is *not* the mantra of the information operations worker. We have the opposite perspective. We win; you lose, but perhaps not in zero-sum fashion. In our world, you size up your opponent with reconnaissance, and you find an information resource opportunity with the goal of making it more valuable to you, less valuable to them, or both.

You do (or better said, "you ought to do") the same thing if you are on defense. You size up your own organization, you find the targets of opportunity, and you determine how as an attacker you could co-opt these. Then, of course, you design and implement the countermeasures to make such an attack harder to accomplish.

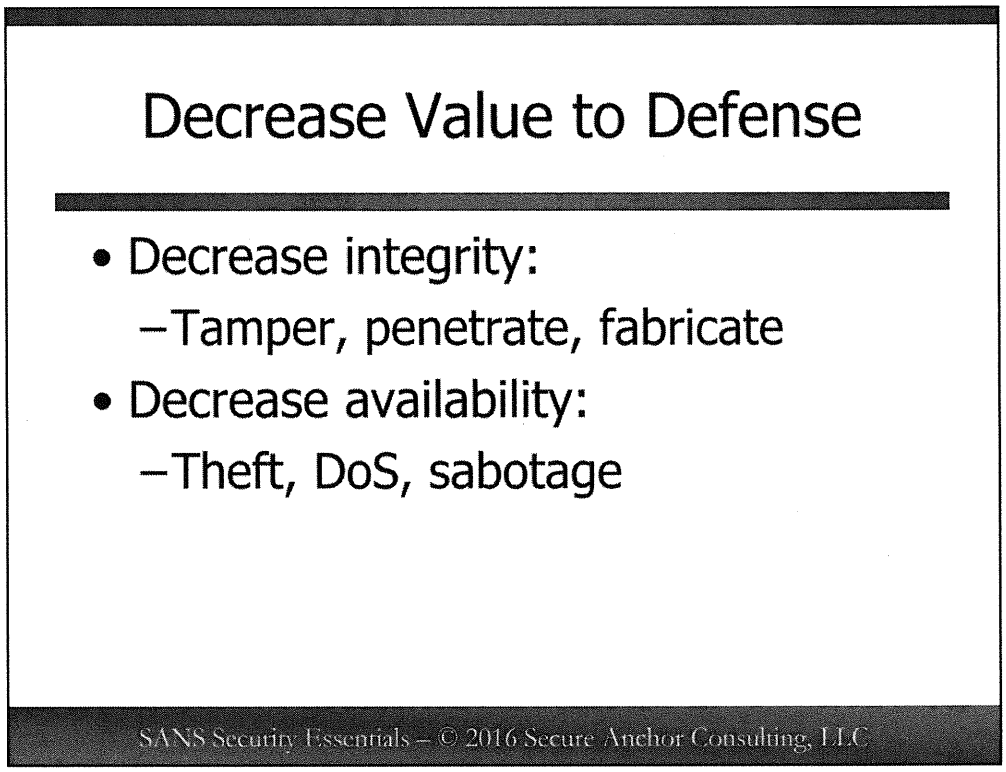


Increase Value to Offense

The attacker is the offense. One of the goals of attackers is to take a piece of data/intellectual property your organization owns and steal it. When the attackers steal a piece of sensitive data, they are increasing the value of the information to themselves. However, in doing so, they have decreased the confidentiality of your information so they can obtain benefit from it.

What is a pound of information worth? This phrase was coined by Perry Luzwick, a DoD information warfighter in the military at the time he said it. He used this phrase to entitle a paper that SANS helped research and develop. If you had a dollar in your wallet and I took it, you would not have the dollar, so you would know that it was missing. If the offense is able to acquire access to the opposition's information, he might or might not know that it is missing. If he does not know the information has been accessed, there are a number of possibilities that could occur.

It turns out that there are very few models to accurately quantify the value of information, and yet, intuitively, we know information is at the heart of the value of a company. In the SANS incident-handling course, we teach the notion of critical program information—the crown jewels of information that make or break a company. This is a central concept in information operations as well.

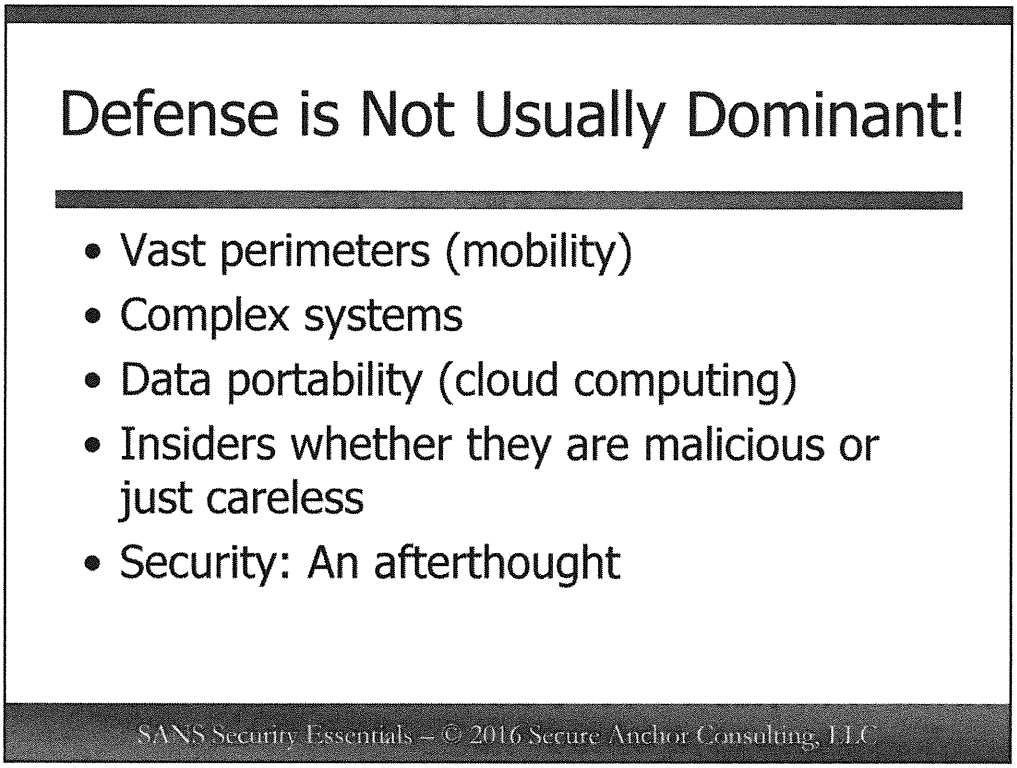


Decrease Value to Defense

On the defensive side, the challenge is to protect the value of our information and hopefully reduce the value of the offense's assets. The most common approach to defense is to establish effective perimeters and seals to maintain and to be able to validate integrity. If the information containers are breached, then a number of things can happen—most of them bad.

Availability has been a whole new ballgame officially starting in the year 2000, with DDoS attacks against yahoo.com and cnn.com; however, there is evidence of serious denial-of-service activity going back to at least 1997. Availability is a simple way to decrease the value of the defense's assets and requires fairly little sophistication. On the other hand, to defend against a large denial-of-service attack is challenging.

If the attacker can penetrate system integrity and copy the information, the defender might never know a breach has occurred. This leaves the defense open to make predictable moves. The attacker might also modify the information, perhaps leading to mistakes that also decrease the value of the defensive player's own information assets.



Defense is Not Usually Dominant!

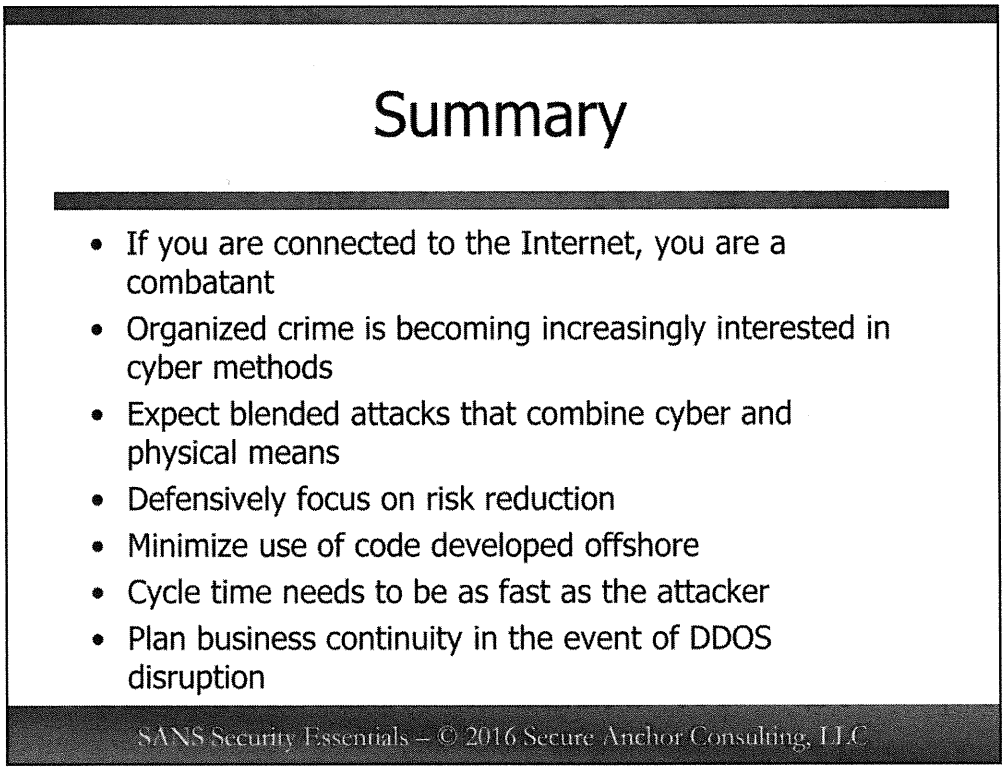
Although defensive dominance, like win-win non-zero-sum games, is a nice thought, it seems unlikely in the real world. The advantage belongs to the attacker. We are all sitting ducks in some senses. If you think in terms of each IP address potentially having 65,536 TCP ports and the same number of UDP ports to defend, you realize the perimeter we have to defend is really big!

Defense-in-depth is one of our most important tools as a defender. If we can learn to think like the attackers, to see through their eyes even just a bit, we can be alert to the opportunities to attack us asymmetrically and attempt to place defenses to prevent those. Sometimes we might be able to use tools like honeypots to understand their attacks and motives, and we might be able, on occasion, to release some incorrect information and cause the attackers to make a predictable response.

However, if the odds are in the favor of the attacker, then we need to learn to attack. Clearly we want to stay well within the law and ethics of our country and profession, but hopefully this module has given you the foundation to think about competitive intelligence. The principles will apply nicely.

Dumpster diving *per se* might be beyond the pale of legal, but companies post tremendous amounts of information on their Web pages and in their papers. Perhaps the best way to gather information on a company, though, is to pose as a potential customer; salespeople seem to feel obligated to share everything they know.

Asymmetry and non-zero-sum game opportunities are out there. One last book to consider is Malcolm Gladwell's *Tipping Point: How Little Things Can Make a Big Difference*. It is not a book on information warfare, but it is an interesting twist on asymmetry. No one has yet solved the problem of teaching people to think in a non-zero-sum fashion step-by-step, but it can be done; there are masters at the game, like George Soros. If you can engineer an asymmetric win, we promise it will be one of the biggest thrills of your life. As a final parting thought, please remember that you are an information warfare player whether you want to be or not. We hope you will strive to be the best that you can be.

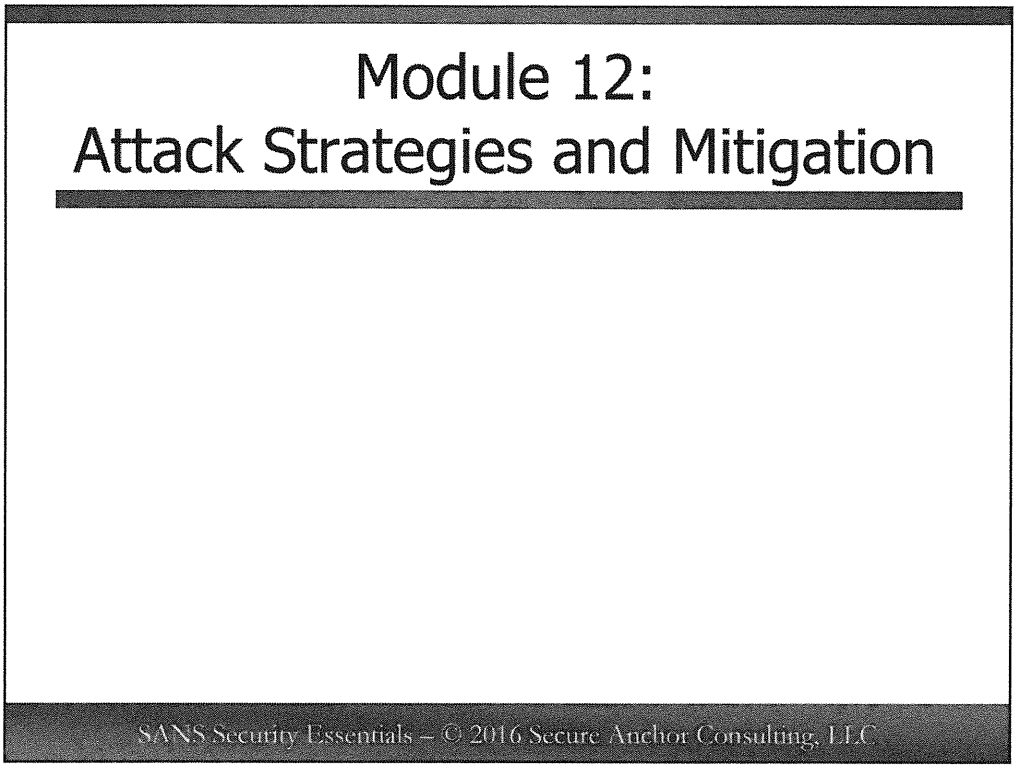


Summary

As demonstrated by the examples and anecdotes provided in this chapter, the threat of being a victim of information warfare is real. Whether a direct target of attack, a target of opportunity, or randomly chosen, everyone is at risk. Further, the level of risk is expected to increase in coming years.

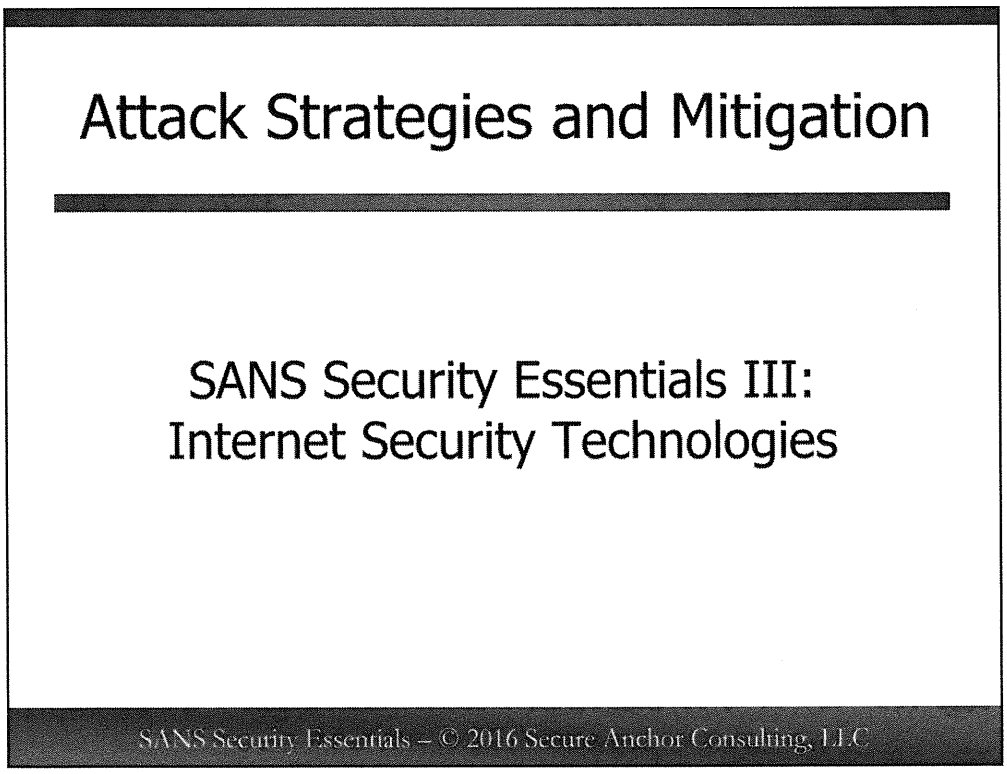
The aggressor's goal is to achieve the greatest impact from the least investment possible (asymmetrical result). Fortunately for him, the ongoing expansion of network perimeters and increasing system complexity aids greatly in the impact available from a relatively minor investment.

The defender's goal is to increase the required investment to a level where the target is no longer attractive. Although achieving a true zero-sum result is unreasonable to expect, there are tools and techniques that can be used to begin approaching balance. Your task is to understand and use these tools to your best ability.



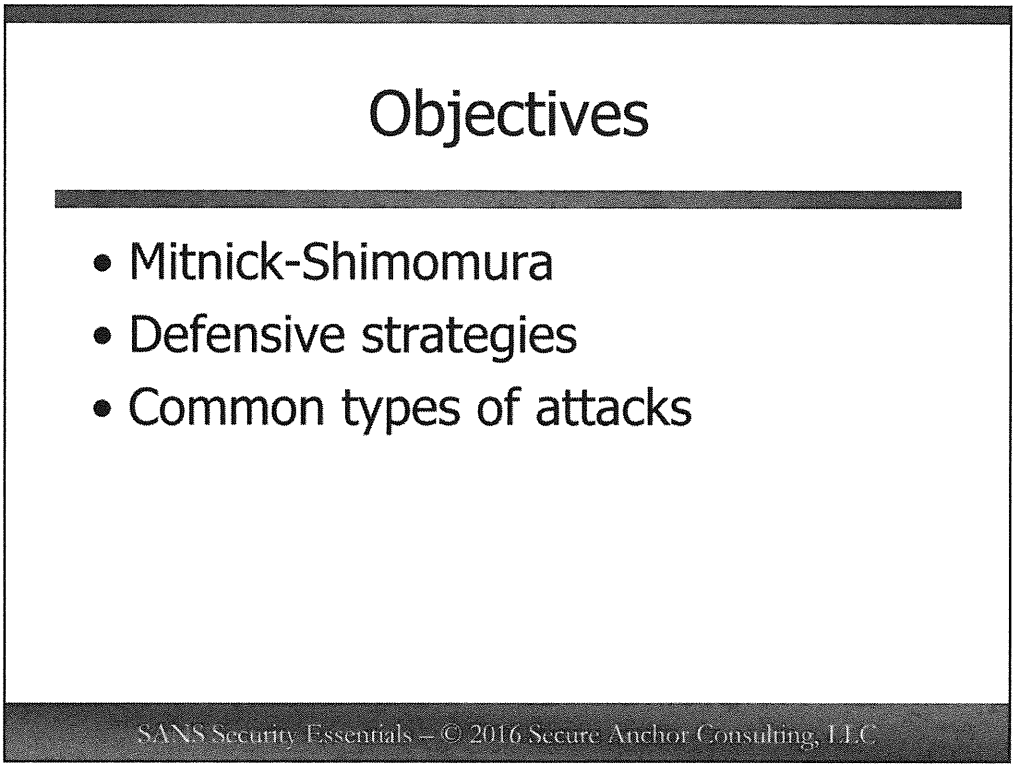
Module 12: Attack Strategies and Mitigation

This section intentionally left blank.



Introduction—Attack Strategies and Mitigation

This section intentionally left blank.

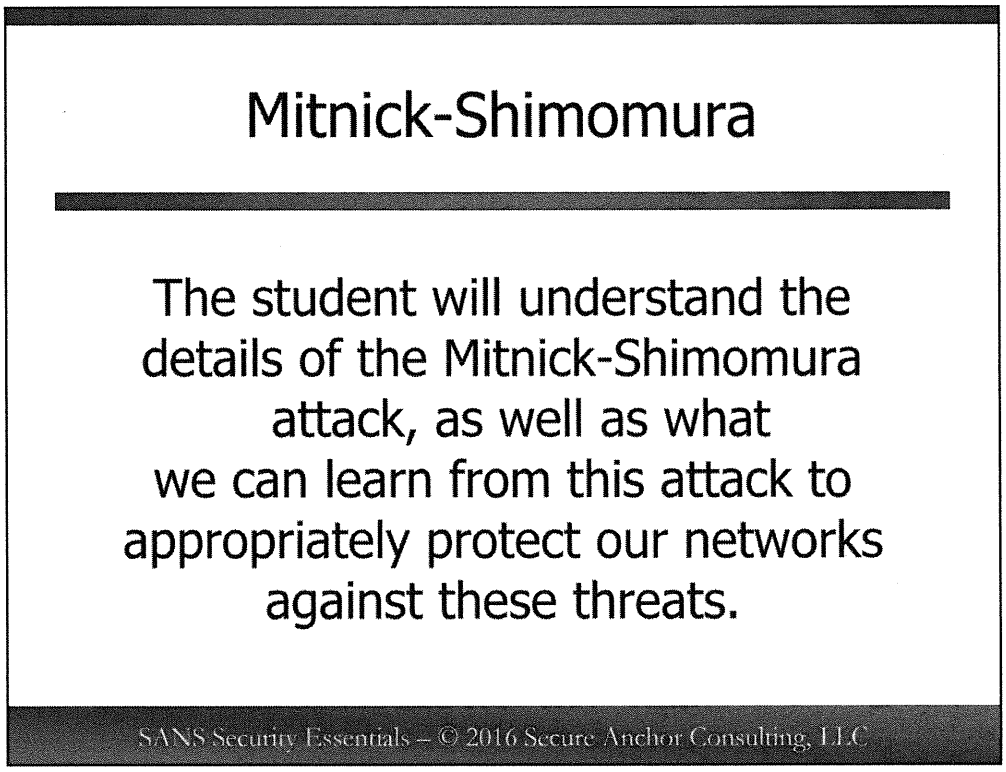


Objectives

The key to the future is to understand the past. The Mitnick attack is historic and can teach us a lot about modern-day security and provides us with key lessons learned.

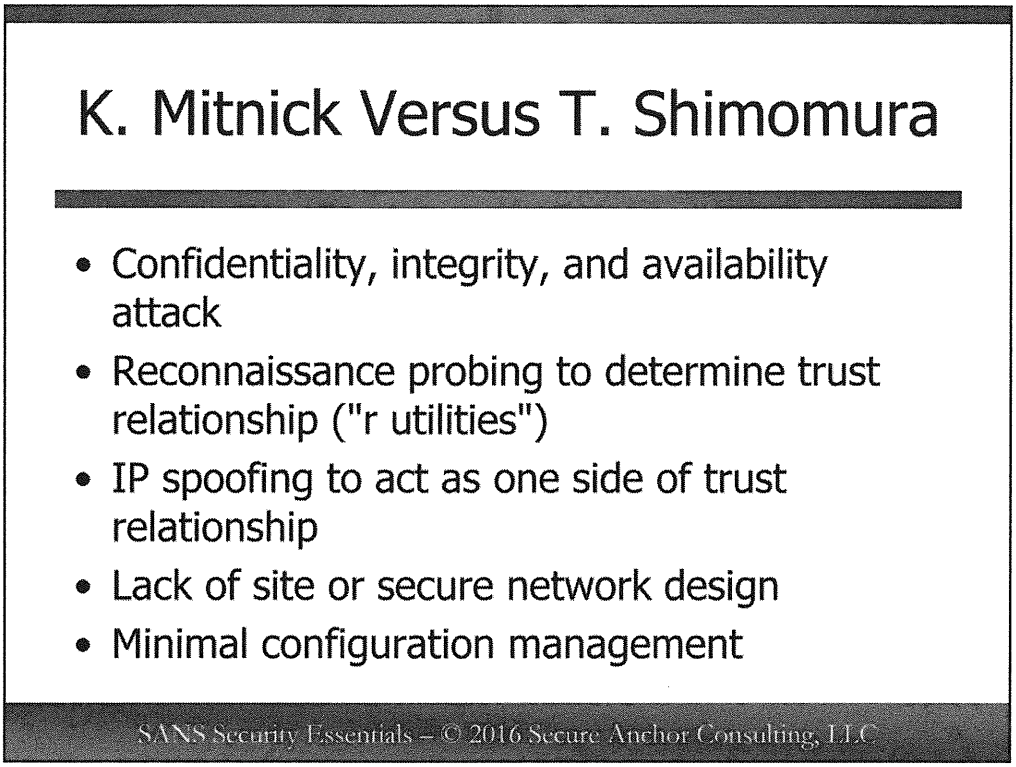
It was Christmas 1994 when Kevin Mitnick executed his now-famous attack against security researcher Tsutomu Shimomura's home network. Using just a few basic attack strategies, Mitnick was able to effect a root-level compromise on systems owned by a skilled information security professional. The sad fact is that many systems still remain vulnerable to these kinds of attacks; thus, this incident isn't so much ancient history as it is a modern-case study. In this chapter, we describe the attack in detail, discussing not only the conditions that made it possible but also some strategies that you can use to help manage the risks associated with these sorts of attacks.

Focusing on individual exploits used in real-world situations is useful, but it doesn't give you a very good look at the bigger risk landscape. The techniques Mitnick used certainly are not the only ones available to knowledgeable attackers. There are almost as many ways to abuse a system as there are ways to use it legitimately. To protect your network, you have to know it. In the second part of this chapter, we present a short list of some important classes of information security threats. We talk about their goals, their distinguishing characteristics, and some strategies you can use to mitigate or remove your vulnerability to these risks.



Mitnick-Shimomura

This section intentionally left blank.



K. Mitnick Versus T. Shimomura

One of the reasons Kevin Mitnick's famous attack on Shimomura's network is a good example for analysis is its scope. The three major tenets of information security—the C-I-A triad of confidentiality, integrity, and availability—were all breached in this attack. By accessing files that were not his, Mitnick compromised the confidentiality of those files, and by penetrating network resources to which he was not granted explicit access, he compromised the integrity of that network. Finally, by executing a SYN flood as a denial of service against one of Shimomura's servers, Mitnick effectively rendered that machine unavailable. Another reason this scenario is an excellent demonstration is the step-by-step execution of Mitnick's attack. First, he scouted the resources he wanted to compromise using the Finger utility. From this, he determined his method of compromise, the Unix r utilities. Then, he used a denial of service to silence a trusted machine and imitated it by way of IP spoofing to gain access to the desired computer. From this system, Mitnick could obtain the files he wanted. Overall, the attack demonstrates the importance of perimeter defenses in planning an effective prevention strategy.

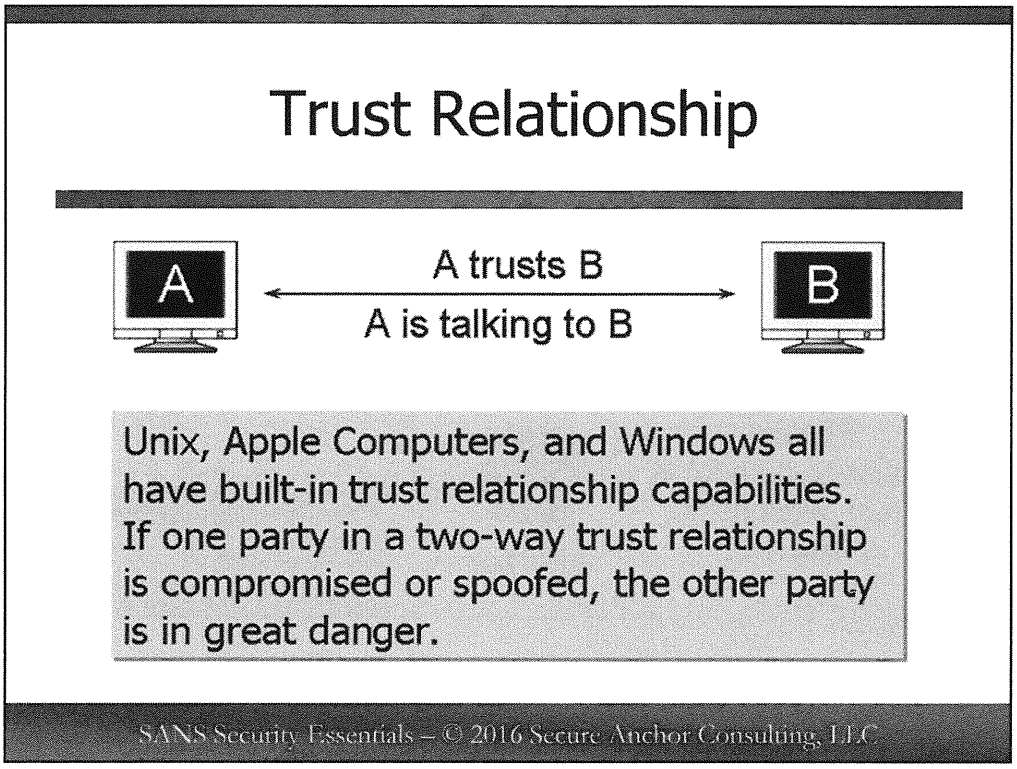
When Toads Attack!

No, this isn't the name of a new reality-based TV show about violent amphibians. Rather, it is the starting point of our story. Shimomura, a respected computer security researcher, was relaxing and spending the Christmas holiday at the home of a friend in San Francisco. The home, known affectionately as Toad Hall, was a social experiment of sorts, exploring what it meant to live online all the time.

There was a substantial amount of computing equipment scattered around the various rooms, all networked and connected via a permanent link to the Internet. It even had its own domain name, toad.com. It might have been sheer coincidence, but at the very time Shimomura was relaxing upstairs, the servers in the basement downstairs were being used as a staging point to launch an attack on his own network, some 500 miles to the south.

According to a posting Shimomura made to a firewall mailing list the following month, one of the servers at Toad Hall had been compromised. The attacker, whose identity was unknown at the time, used the server to send a series of probes to Shimomura's home LAN in San Diego. Shimomura's network consisted of a workstation, a server, and another machine. The workstation, diskless, served as an X terminal. Both the workstation and the server ran Solaris 1, also known as "SunOS 4." The additional machine housed Mitnick's ultimate target—some cellular telephone software Shimomura had reverse-engineered. For simplicity's sake, we refer to these three systems simply as the terminal, the server, and the target.

As you read through the rest of this analysis, keep in mind that Mitnick didn't invent this attack himself. The techniques involved had been known for some time, and he introduced no new innovations, except to perform it in the real world. Still, it makes a great case study because it is so easily understood. We particularly like the fact that there's a clear information-gathering phase that we can use to examine how clever intruders can map out trust relationships to abuse. In fact, the attack easily can be broken up into several clear phases, which we now examine.



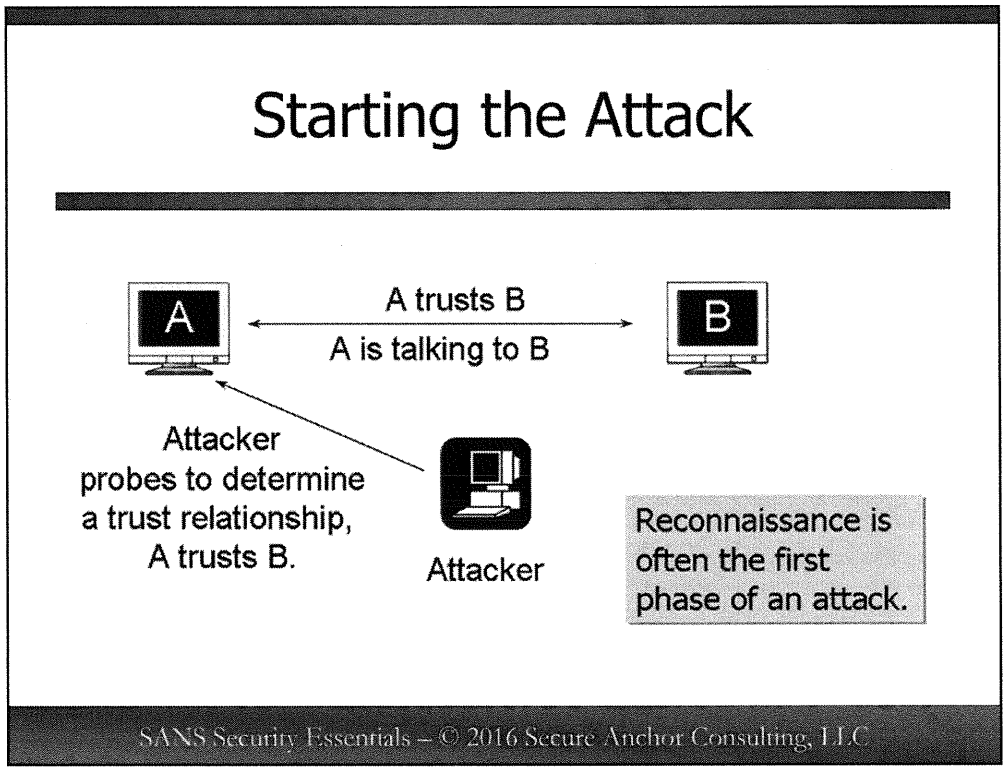
Two Systems, Trust Relationship

Phase 1: Scouting the Coast
If you're going to stealthily infiltrate an island fortress in the dead of night, the first thing you need to do is to scout the coast. You need to know the location and composition of the structures on the island, both man-made and natural. You want to discover all the bays, lagoons, rivers, and beaches to see where you can most easily come ashore without being detected, preferably someplace close to your eventual objective. Shimomura's LAN was this island fortress, so Mitnick's operation commenced with a short reconnaissance of the virtual landscape.

Like most well-planned attacks, this one started with a series of gentle probes designed to gather information about the victim's systems. Mitnick knew that he'd likely be unsuccessful in obtaining the cellular software via a frontal attack. The target was too well-secured for that. But if that machine could not be attacked directly, perhaps some other system could be compromised instead, and its trust relationship with the target could be abused.

According to Shimomura's analysis in the days after the attack, Mitnick started by sending out several feelers to determine just this sort of information. Their purpose was to discover possible trust relationships among the three computers on Shimomura's LAN. These trust relationships simply means that a computer is familiar with another computer and trusts the information that is coming from it. If one of these hosts is compromised, the other is much more susceptible to attack.

Complex attacks against specific targets usually start with a reconnaissance phase in which the attacker maps out the lay of the LAN, so to speak, determining which hosts are present and gathering as much information about them as possible. After all, if the attacker hopes to achieve a specific goal, he'll want to know what resources are available for his use.



Starting the Attack

Finger is a service that can return information about users on a particular system and is used for reconnaissance. If you supply a username, finger happily will tell you a lot of information about that user. However, what if you don't know the names of any valid users on that system? No problem! Finger will be happy to tell you all about anyone who's currently logged on to any host you name. The output can be quite informative, containing valid user names, along with their home directories and shells. It can even tell the last time a particular account logged in and from which machine. To someone trying to gather intelligence about your network, this is a treasure trove of information. Because it can be so helpful to attackers, many system administrators routinely disable the finger service on machines for which they're responsible.

The finger probes Mitnick used were designed to look for patterns of remote logins between the three computers. The assumption is that if a particular user regularly logs on from a certain machine, there's a good chance that a trust relationship exists with that machine. In other words, the attacker hopes that the act of having to type an account's password each time someone logs in from one machine to another might have become so annoying that the user set up a .rhosts file to direct one computer to accept incoming login connections from the other computer on trust and not prompt for a password. If the user successfully logged in to the first computer, then tried to remotely access the second one, that machine would then accept the first computer's word about the user's identity and log him in without challenge.

Users often set up these trust relationships on their own accounts to make it easier for them to work on many machines at once. A system administrator might also do the same thing for all users on a machine by creating a hosts.equiv file. Although this sort of trust is convenient for users, it is usually a gaping security hole. Unfortunately, this is still common in many Unix environments.

Filesystem Information from showmount and rpcinfo

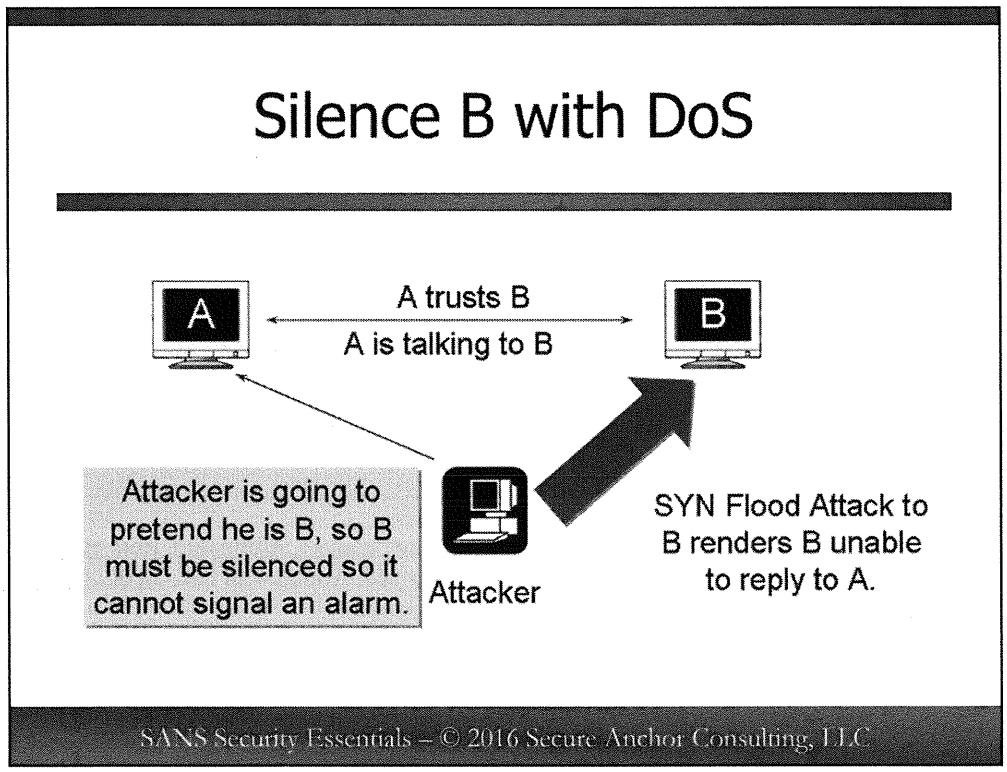
Now take a look at the other two commands Mitnick used during his reconnaissance, showmount and rpcinfo. showmount lists the filesystems exported by an NFS file server, which this machine was not.

For its part, rpcinfo tries to enumerate the various RPC-based services on a remote machine. Most of the services listed would be related to NFS or to Sun's Network Information Service (NIS), the popular network user database included with most varieties of Unix. Based on the nature of his later attacks, and from the fact that we know the X terminal was a diskless workstation without the capability to act as a fileserver, we can guess that these commands probably didn't turn up much useful information.

Tracing the Trust

It is particularly interesting to note the progression of these probes, from the actual target of the attack to the eventual diskless workstation that later would become Mitnick's initial point of entry. Although we don't know his thoughts for sure, we can feel fairly confident that he was tracing a series of possible trust relationships between the three machines. Perhaps while fingering the target, he discovered an idle login session from the X terminal (there was one, it turned out). Maybe he also noticed a repeated pattern of logins from the server to the X terminal and inferred that there might be some trust between the two. Whatever his reasoning, he eventually decided that the diskless node would be a good foothold in Shimomura's network. That could be why he used rpcinfo and showmount against it but not the other hosts.

No matter what his exact thought process was, he was able to gather enough information from all three machines to make a pretty good map of the relationships they enjoyed. Armed with the output of his initial probes, Mitnick progressed to the next stage of his attack.



Silence B with DoS

Phase 2: Cutting the Phone Lines

Back to our island infiltration mission! After you manage to slip onto the island undetected, you have to get through the compound's front gate. The rudimentary electronic door lock is connected to a computer system elsewhere in the compound, and it uses this connection to verify your identity. If you're not authorized for entry, you have to trick this lock into thinking otherwise. Fortunately, the designers of this building were builders and not cops. They run the lock's connection through a set of wires in a fairly obvious conduit along the outside wall of the building. It is a fairly simple matter to cut this connection so the remote system won't be able to communicate with the lock. This isn't enough to get you in the door yet, but it is a necessary first step. You've now removed the possibility of someone rejecting your false ID. This is exactly what Mitnick did for his next step, too.

Mitnick decided to try an IP address spoofing attack against Shimomura's X terminal. IP spoofing is just what it sounds like: You send packets to a remote computer but lie about your source IP address. Mitnick guessed that the X terminal might have a trust relationship with the server, allowing the server's users to log in locally without having to type a password. Therefore, his first goal was to open a TCP connection to the terminal's remote login service (port 513) that would appear to come from the server, so he wouldn't be challenged for a password. But first, he had to overcome a significant hurdle: The server machine had to be silenced.

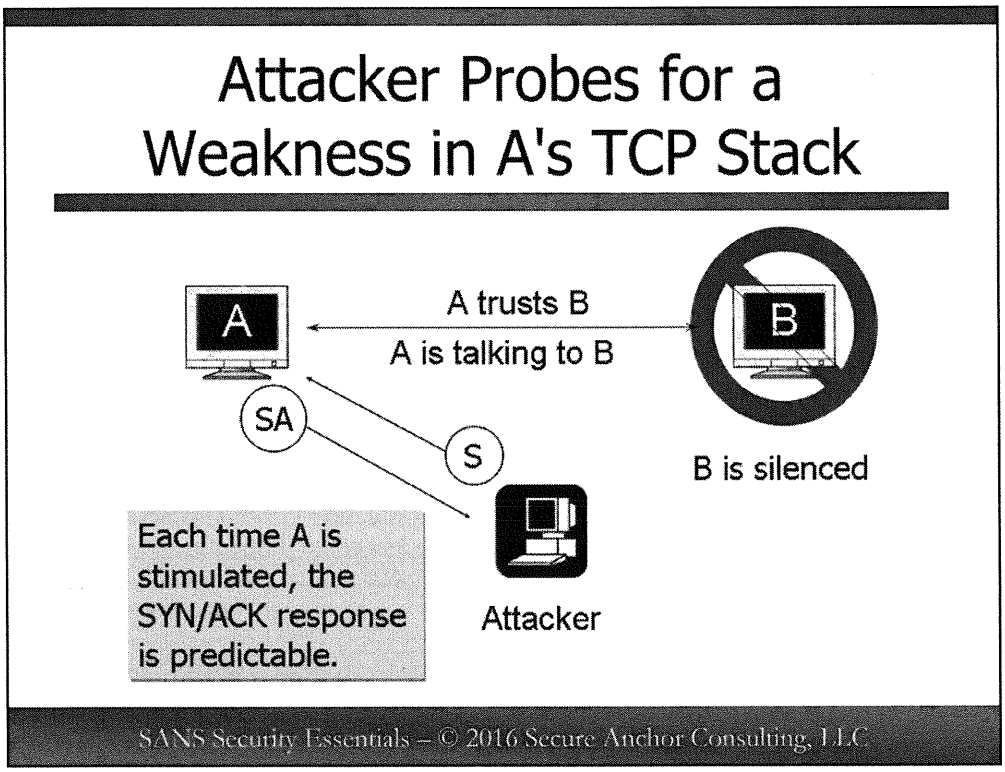
Do you remember the three-way handshake TCP uses to open connections? The initiator of a connection is supposed to send a SYN packet to the destination, along with a randomized initial sequence number (ISN). The destination host ACKs that SYN with the initiator's ISN, and also includes its own SYN and another randomized ISN for the second half of the connection (the half that sends data back to the initiator). The destination expects to receive a final ACK from the initiator along with its randomized ISN (actually, the ISN + 1, but who's counting?). If the ISN in the ACK for either side of the connection is wrong, the whole process terminates and no connection is made. If you hope to be able to complete the handshake and make a new connection, you need to know the ISN that the destination host chose for its SYN.

Here's the trick: If you're spoofing packets to the server, you're giving it the IP address of some other machine as the originator of the connection. Therefore, the ACKs will go to that other machine, and not you. What happens if a machine gets an ACK to a SYN it never sent? You guessed it! It sends a reset (RST) packet in response to the ACK, which causes the intended victim to close the whole connection attempt right there, foiling your evil plot. Thus, if you're trying to spoof the address of a specific machine in order to exploit a trust relationship, you have to find some way to silence it before you can proceed.

Conceptually, the easiest way of silencing a machine is to crash it. Although there are numerous tools that can do this, especially for older OS releases, crashing a machine is pretty obvious, because it is easy to notice when a machine is entirely down. To help minimize the chances of his attack being detected, Mitnick chose a stealthier way to temporarily silence the server without actually bringing it down.

The technique he chose, known as "SYN flooding," involves sending numerous SYN packets to the machine to be silenced but never completing the TCP handshake protocol. A lot of OSes allocate a fixed-size buffer to handle TCP handshakes while they are being negotiated. This attack sends so many SYNs that it fills the buffer and prevents any other handshake processing. While the buffer is full, the OS won't respond to any incoming connection attempts, not even the ACKs that are sent as a by-product of the IP spoofing attack. These buffers usually have timeouts associated with them, so the half-open connections don't stay around forever, but the timeout usually is pretty long, maybe even a few minutes, which is long enough for an attacker to do his job.

Let's pause for a moment to review the situation so far. The server received an overwhelming number of requests to create new TCP connections, but none of these faux attempts continued the protocol past its first step. The server kept hoping to complete the handshakes, though, so it placed these SYN packets in a buffer it could refer to when their ACKs eventually arrived. After this buffer filled up, the server started ignoring all other TCP protocol processing. The SYN flood attack effectively silenced the server temporarily, so that Mitnick could proceed to the IP spoofing phase of his plan without fear that the server would receive the ACKs to his false SYNs and cause the server to reject his spoofed connection attempts.



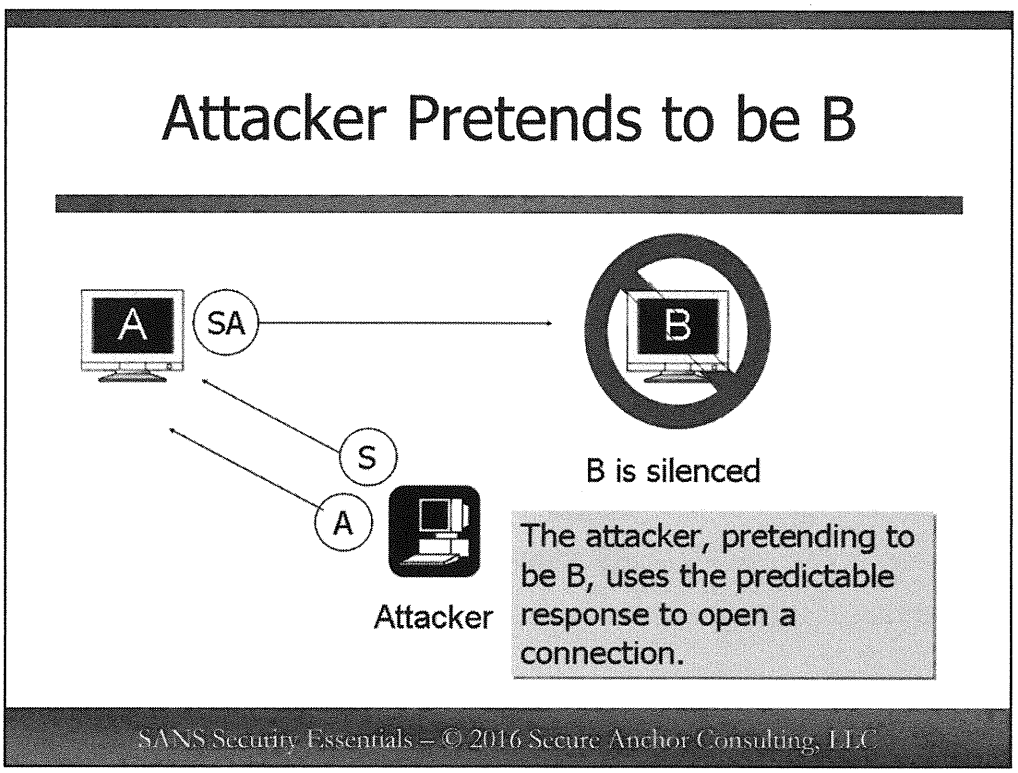
Attacker Probes for a Weakness in A's TCP Stack

Phase 3: Analyzing the Lock

Okay, you're back on the island again. In the previous section, you prevented the nifty electronic lock from communicating with the central system that verifies IDs. This isn't enough to get you through the door, though. You still need to convince the lock that the ID you're going to present is a good one. Probably the easiest way is to hook up your own "central system" to the communication cables you just cut. You know the basic protocol, but there are a few little details of the communication you can figure out only "at run time," so you hook up your palm top to the cables and experiment a little to find just the right combination. That's what Mitnick did next, too.

Think back for a moment when we were studying IP concepts. Do you remember the role of the initial sequence number in negotiating a TCP connection? When one computer sends a SYN packet, it generates an ISN and includes it in the packet. It expects that the other end of the connection will ACK with the same ISN (plus 1, remember, because the ACK indicates the next byte the ACKer expects to receive). If you think about it, this can be a problem for someone trying an IP spoofing attack. Because the attacker is sending her packets with the IP address of a different machine as the source, the receiver naturally will attempt to send the ISN to the spoofed address. Not only that, but if the attacker ACKs with the wrong ISN, the victim simply will reject the connection attempt. If the attacker never sees the victim computer's ISN, how can he complete the connection?

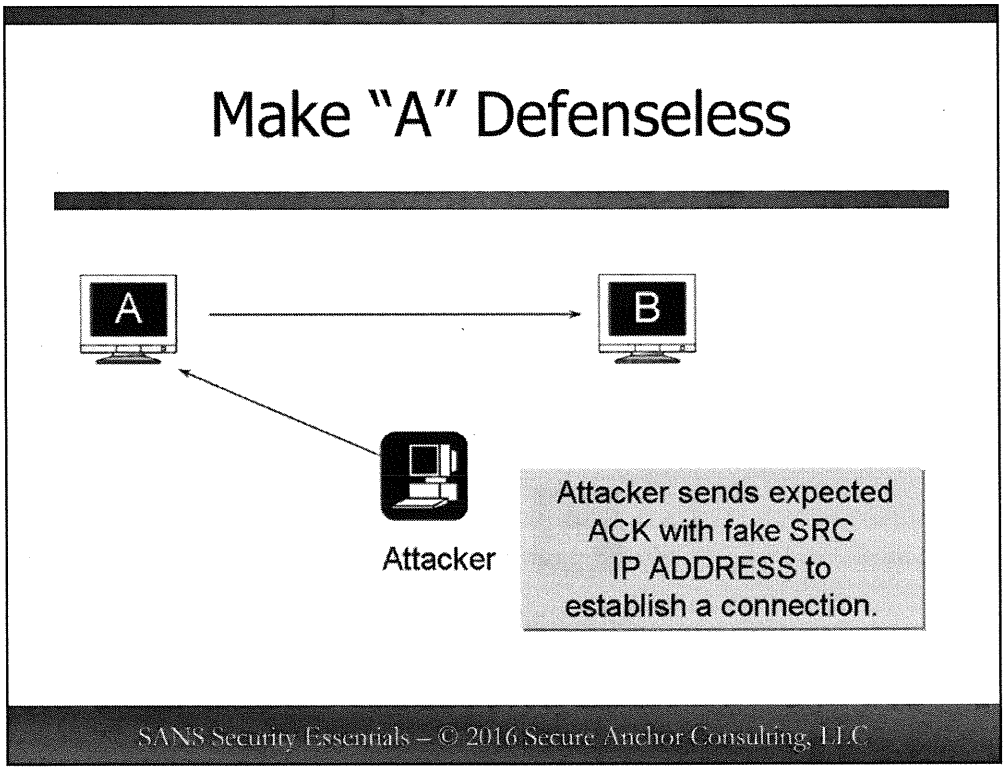
Like all really good ideas, the answer is obvious (after a lot of thought). The attacker has to predict the ISN that his target will use. It turns out that, although the TCP specification has pretty good protection against ISN guessing, many implementations either ignore the specifications or implement them improperly. The problem is guessing an ISN is supposed to be hard to do. At the time of the attack, however, most TCP implementations followed fairly static rules about how to generate ISNs, and most were vulnerable to simple guessing attacks. This remains the case today, but to a somewhat lesser degree.



Attacker Pretends to be B

Phase 4: Picking the Lock
From the previous slide, you can see what is happening at this stage of the attack. The attacker sends a SYN packet from an outside connection that is being spoofed as a trusted source (B). The target machine (A) replies with a SYN/ACK that the attacker never sees—it goes to the real B. However, armed with a predictable sequence number, the attacker completes the three-way handshake by sending an ACK to the target.

Now, back to our analogy. Your attack on the island fortress' bunker is nearly complete! You've figured out the inner workings of the lock and how it communicates with its back-end server. You've even connected your own phony server to the lock's communication lines, so you can trick it into allowing you access. All that's left is to present your fake ID, and to hope it works. You swipe it through the reader ... Success! The ponderous steel door slowly creeps open, and you're in!

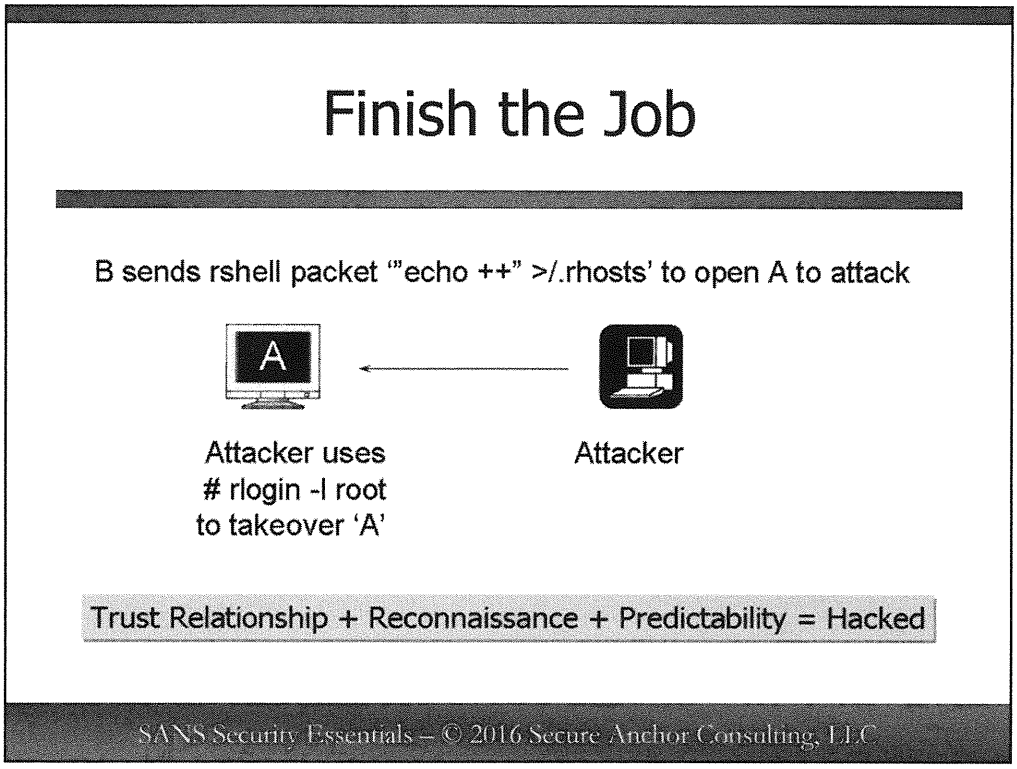


Make "A" Defenseless

Having gotten this far, Mitnick's next move was simple. Using the sequence number data gained from his probes, Mitnick was able to make a good guess as to the next ISN the X terminal would try to use. All he had to do then was to fake a connection to the X terminal's remote shell service that appeared to originate from the server. After waiting a short time, he could fake an ACK to the X terminal's SYN, too, using the guessed ISN. With a little bit of luck, he'd manage to convince the X terminal that the connection attempt had been successful. Unfortunately for Shimomura, luck was with Mitnick that night.

After connected to the X terminal's remote shell service, Mitnick mimicked the rsh protocol, lying to the terminal that his connection was coming from the server's root user. He hoped that the terminal trusted root to log on automatically from the server without having to supply a password. Apparently, this was indeed the case.

ISN guessing has one slight drawback: It only works against idle servers. If the machine you're trying to access is busy accepting other connections while you're attacking it, you can't guarantee that the next ISN it presents you with will be the one you guessed, since some other connection(s) might have been processed between the time you probed and the time you tried to use your guessed ISN.



Finish the Job

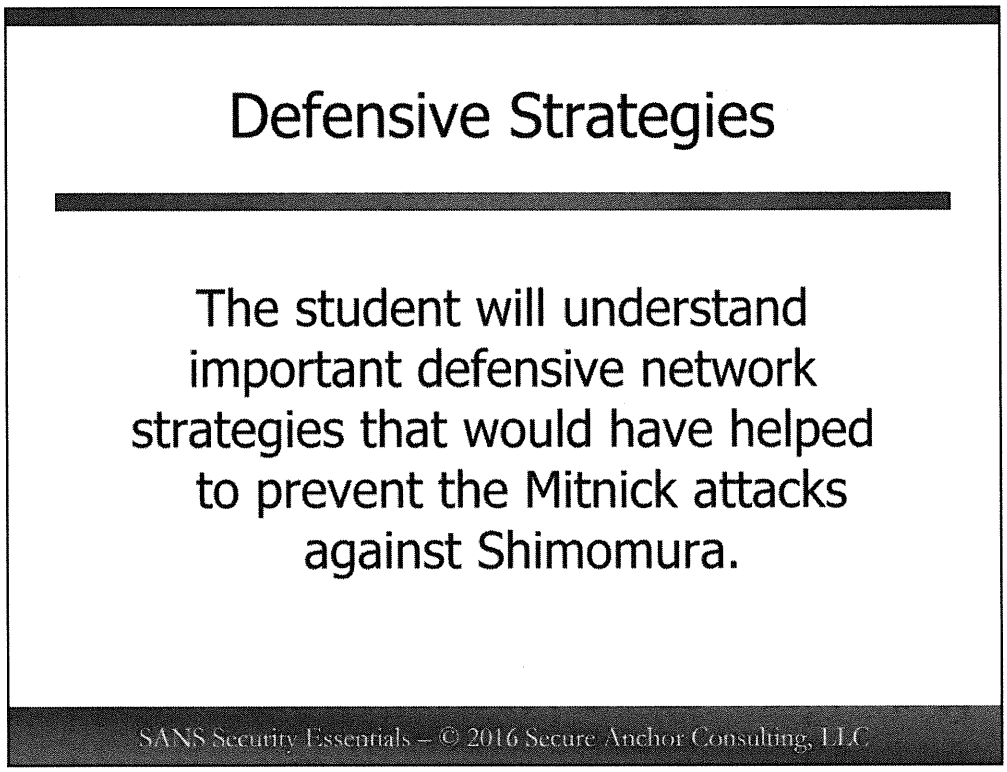
After successfully logged in as root, Mitnick had a simple goal: to preserve his root access and make it simpler to log in the future. To accomplish this, he modified root's .rhosts file to ensure that any user from any other machine on the Internet could log in as root on the X terminal without being challenged for a password. The command he issued was echo + + >> /.rhosts. This adds ++ to the file, allowing any user from any host to connect to the machine.

After this, it was a simple matter for Mitnick to clean up after himself. First, he shut down the spoofed remote shell connection and sent a flurry of RST packets to the server to clear up the temporary block he'd placed on it. After it was fully operational and responding to connection requests again, Mitnick hoped his slight modification to root's .rhosts file would have a better chance of going unnoticed.

What Mitnick did then was fairly straightforward. Using his new root backdoor to the X terminal, he logged on (using the command rlogin -l root, as seen in the slide) and brought over a nifty kernel module capable of tapping into an existing login session. Then he used the module to take control of a session Shimomura already had established from the X terminal to the target machine. Mitnick then simply browsed around until he found what he wanted and copied it for his own later use.

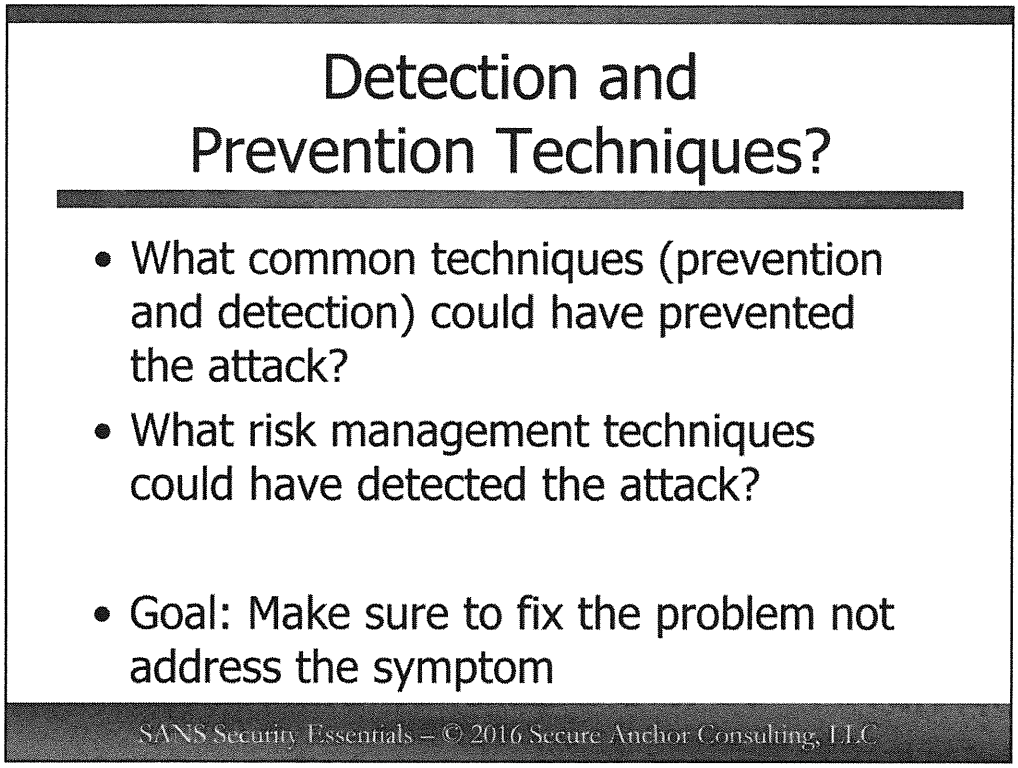
Fortunately, Shimomura's network was configured to keep good audit logs of Internet-based logins. An associate of his was watching over the network while Shimomura was out of town and noticed that the audit trail suddenly had gotten smaller! The only way for that to happen was for someone to manually edit the logs, probably as a way to help cover his tracks. Once alerted, Shimomura was able to analyze the attack and start the recovery process.

What followed was a long, arduous odyssey during which Shimomura partnered with the FBI and other law enforcement agencies to track down and arrest Mitnick. You can read more about this fascinating story, told in the Shimomura's own words, in his book, *Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw - By the Man Who Did It* (ISBN 0786862106). Published by Hyperion in February 1996, this is a clear and interesting account of the whole process.



Defensive Strategies

This section intentionally left blank.



Detection and Prevention Techniques?

It is certainly the case that an intrusion into your network can cause you no end of aggravation. Simply identifying the scope of the compromise can be a large task sometimes, so whatever energy and resources you can devote to protecting yourself ahead of time will almost certainly have large paybacks in the future. Let's look at a few techniques that you could use to help mitigate the risks associated with attacks like Mitnick's.

Electronic Toad Remover

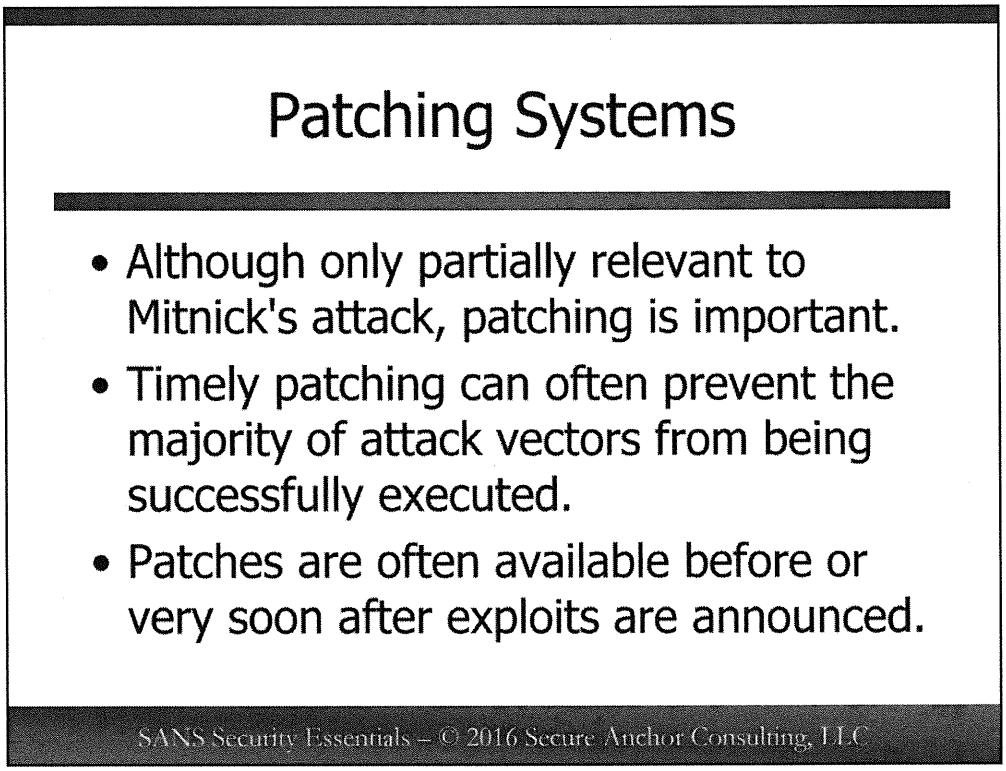
The information security cycle consists of three parts: prevention, detection, and response. The earlier you insert your countermeasures in this cycle, the more cost-effective they will be. Following that rule, the biggest return often comes with techniques designed to prevent attacks from occurring.

Prevention Isn't Enough

Not only is prevention usually the most cost effective way to deploy security resources, but it is probably also the most obvious. Sometimes it is too obvious. Organizations that rely solely on prevention often get taken for a bad ride when an attack eventually manages to get through all their preventative measures.

At a minimum, you should have well-defined incident response procedures in place to help you figure out what to do when your defenses are finally breached. And they will be breached eventually; it is just a question of when and to what extent.

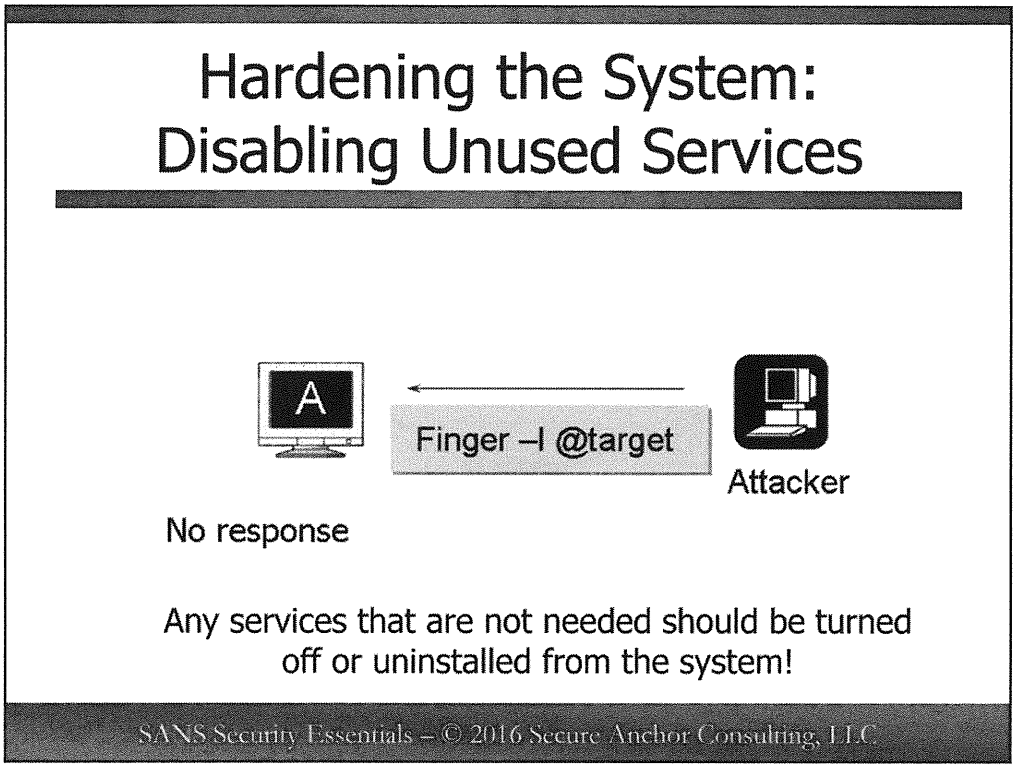
An ounce of prevention is worth a pound of cure.
- Benjamin Franklin



Patching Systems

In executing his attack against Shimomura, Kevin Mitnick did not use unpatched systems directly. The inherent lack of security in the r* commands themselves enabled this compromise. However, in modern-day information security, patching of systems and applications is often considered to be one of the most important tasks for system administrators and security professionals alike.

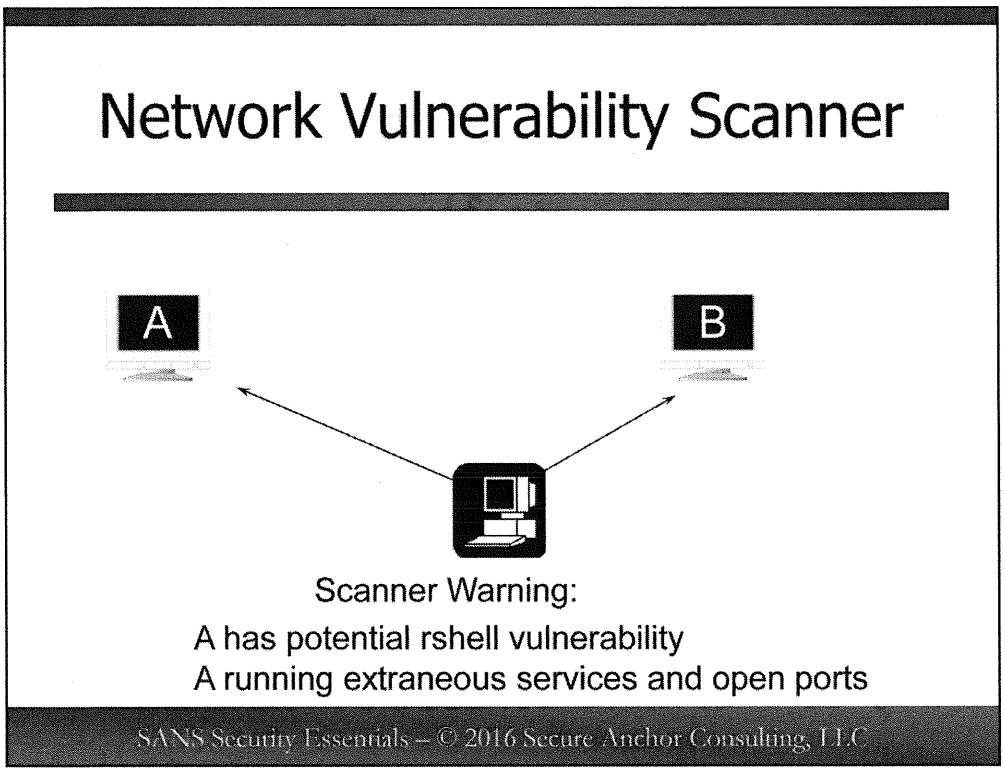
When a vulnerability is discovered in an operating system's code, or an application is found to have a flaw, the software vendor usually releases a patch of some sort to plug the leak. Many times, these patches are available to administrators before, or shortly after, someone has actually created exploit code to take advantage of this vulnerability. By effectively managing the process of patching systems and applications, many attacks can be prevented before they really even get started.



Hardening the System: Disabling Unused Services

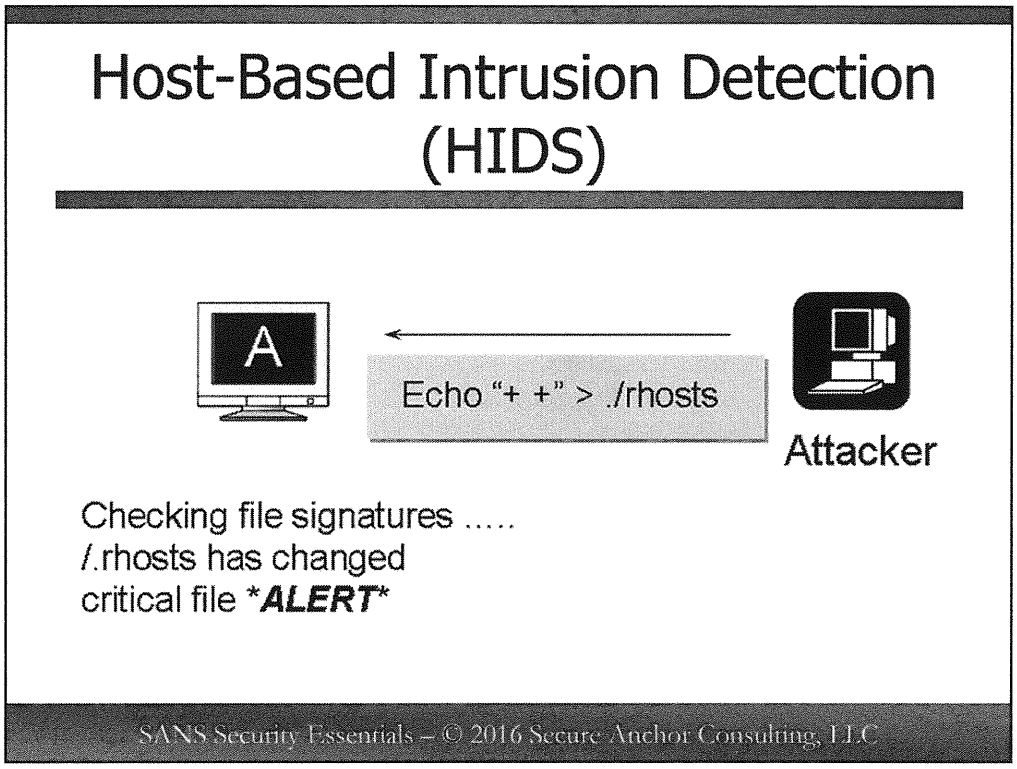
What if Shimomura's systems had not responded to the finger command Mitnick used for reconnaissance? This command, in its various forms, was instrumental in providing Mitnick with the necessary information to successfully stage his attack against Shimomura's network. Without this information, it would have been considerably more difficult to discover trust relationships, or anything else for that matter.

What about the r* commands? If Shimomura had disabled both finger and the various r commands, how would Mitnick have broken into the network? This all serves to illustrate a simple point: You should disable any and all services that are not essential to the system's function. Many operating systems come with unnecessary services installed and enabled out of the box, which can place a system at immediate risk once connected to a network. By carefully selecting options during installation and routinely auditing systems for unnecessary services or applications, you can significantly reduce the risk of a threat exploiting a vulnerability.



Network Vulnerability Scanner

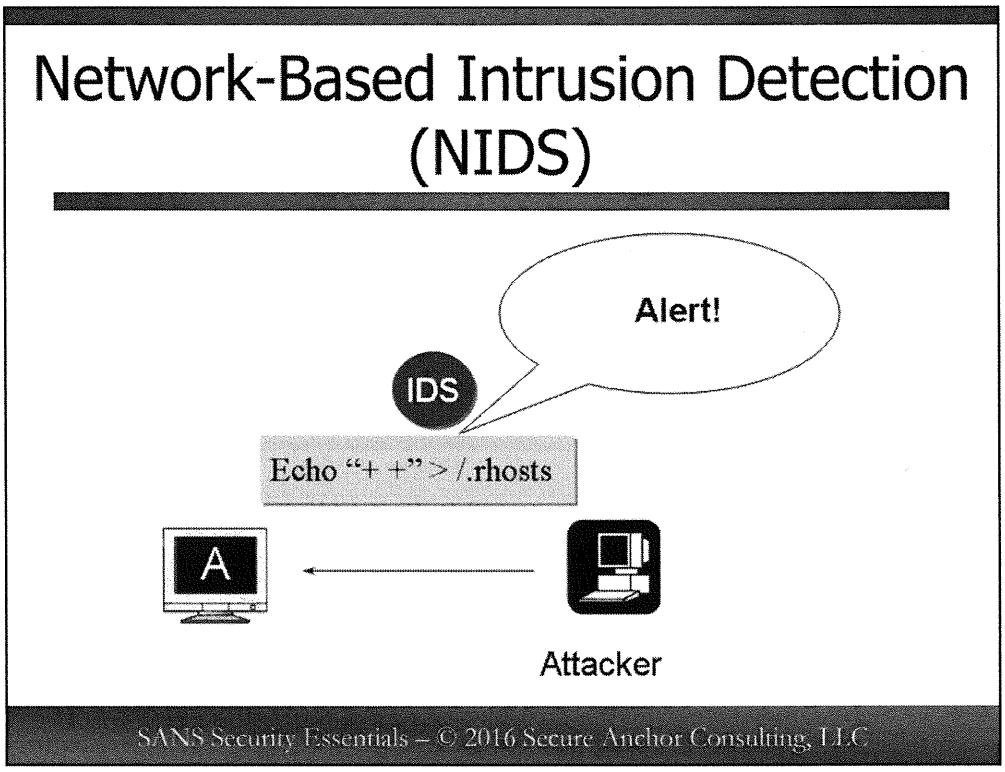
We're going to cover vulnerability scanning in more detail, but we felt it was important enough to merit at least a brief mention here. If you scan your own networks regularly, you stand an excellent chance of finding and closing vulnerabilities before attackers can exploit them.



Host-Based Intrusion Detection

Host-based intrusion detection systems (HIDSs) usually consist of software that resides on a host machine and monitors the traffic in and out, as well as the integrity of the host's files. HIDS can be trained to record a system's initial baseline of users, running applications and services, and particular files to monitor, and it can then alert an administrator when any one of these elements changes unexpectedly.

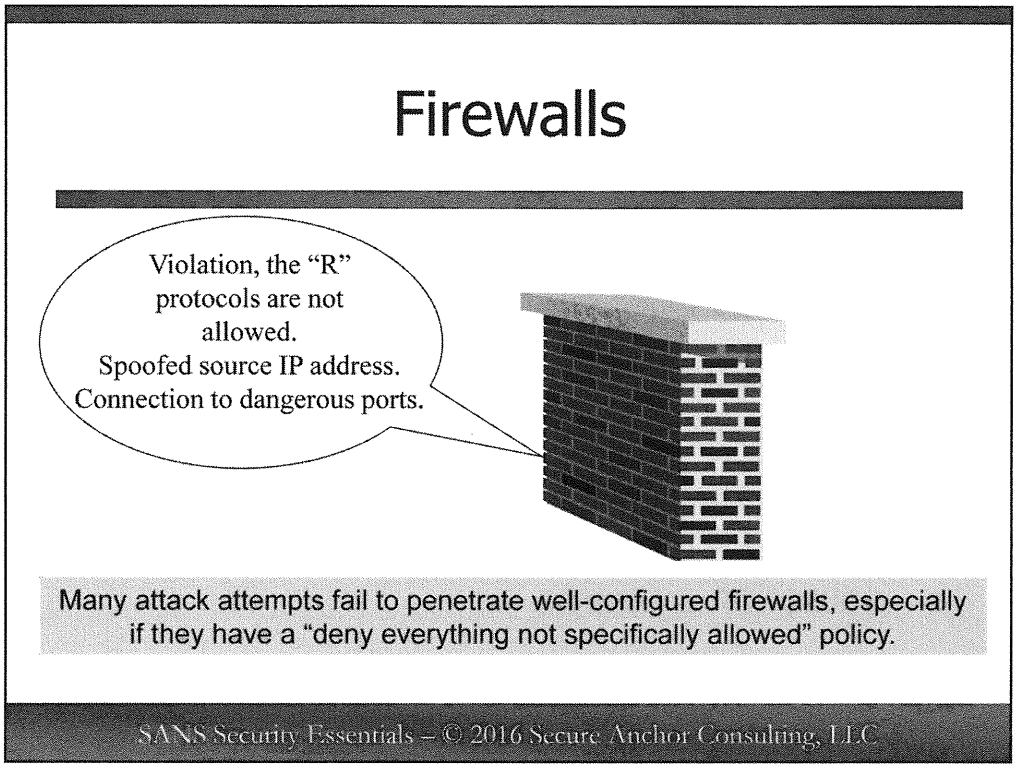
HIDSs are often considered in the same class as host-based firewalls, and several current products can adequately perform the functions of both. Other types of HIDSs are distributed as software agents that can then be monitored from a central console; this type is more practical and suitable for a large enterprise. Typically, HIDS solutions are implemented with robust logging capabilities enabled on the host as well.



Network-Based Intrusion Detection

What if a successful attack isn't blindingly obvious? How will you know when it is time to put your incident response plan into action? More than likely, you'll also want some sort of detection mechanism to fill the gap in your security cycle.

> Although we highly recommend starting with good preventive techniques, it is important to keep in mind that they are only a third of the total cycle of risk management. Be sure to follow them up with good detection and response mechanisms. That's security-in-depth. You will thank yourself later.



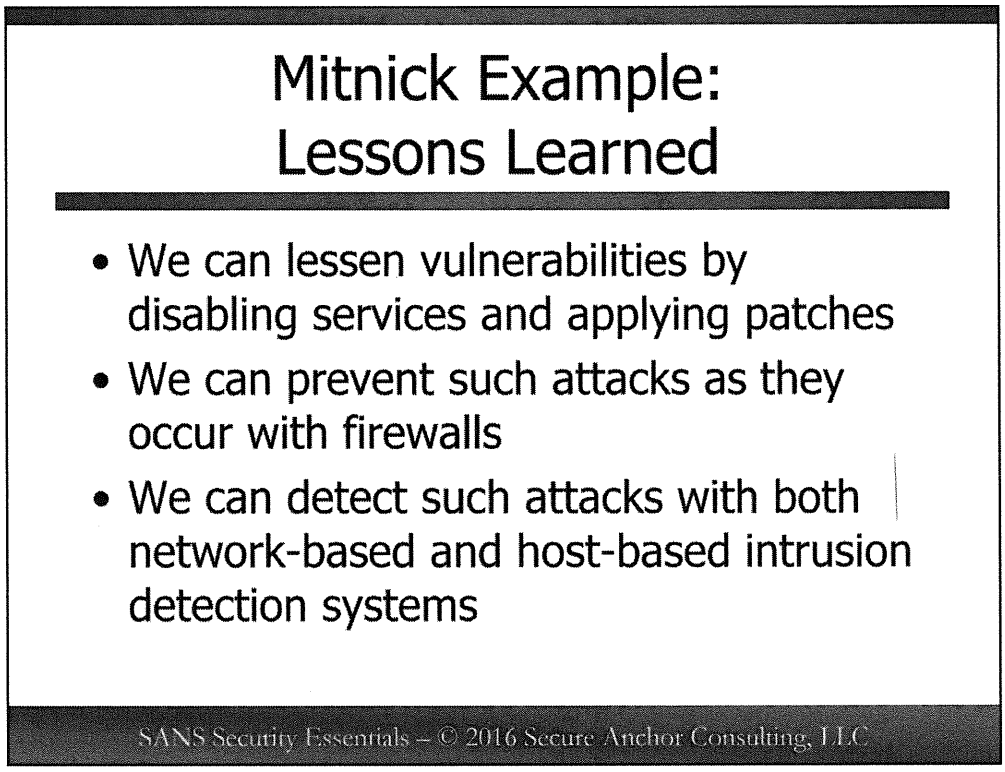
Firewalls

Probably the first thing any security analyst does when he designs a network these days is to plan for a firewall. It is almost impossible to have any kind of good internal security control without first establishing a secure network perimeter. In fact, the principle of security-in-depth practically demands that you be able to control the traffic entering and leaving your network. Fortunately, firewalls are very visible components of today's information security scene. They're usually the first thing management thinks of when it writes the security budget.

A good firewall (or at least a filtering router) can help prevent a variety of different types of attacks. In our scenario, it provides two very helpful functions: It prevents outsiders from accessing internal network services and from using spoofed IP addresses that should appear only inside your own network.

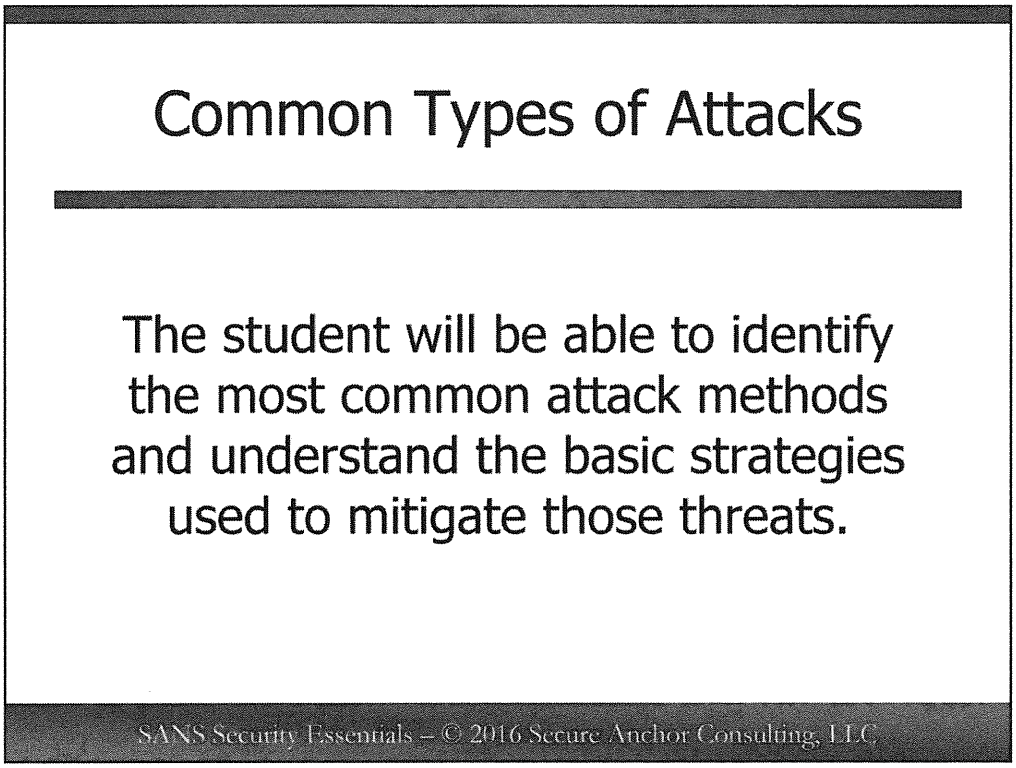
Blocking access to non-critical services probably is the single biggest benefit of any of the risk management techniques we're going to discuss. Why offer to the entire Internet every service that's running on your internal LAN? Offering such an environment provides what the military would call a target-rich environment. If you narrow down to a select few range of services you offer, you can concentrate on configuring those services in as secure a manner as possible, while simultaneously denying an attacker any possibility of using poorly managed secondary services against you. In this case, had Shimomura deployed a firewall to block outside access to the r* command suite, finger, rpcinfo, and showmount, Mitnick would have had to search for some other way in.

So your firewall can keep outsiders from accessing your internal network services, but what if those outsiders are pretending to be insiders? Most routers and firewalls these days are smart enough to know which IP addresses should appear only inside your network and to reject external traffic bearing those addresses. Had this sort of protection been in place for Mitnick's attack, the spoofing would have failed because the firewall would have rejected Mitnick's spoofed packets, which originated from outside the network yet appeared to use an internal IP address. This kind of protection is pretty much the standard these days, and there's probably no excuse for not implementing it.



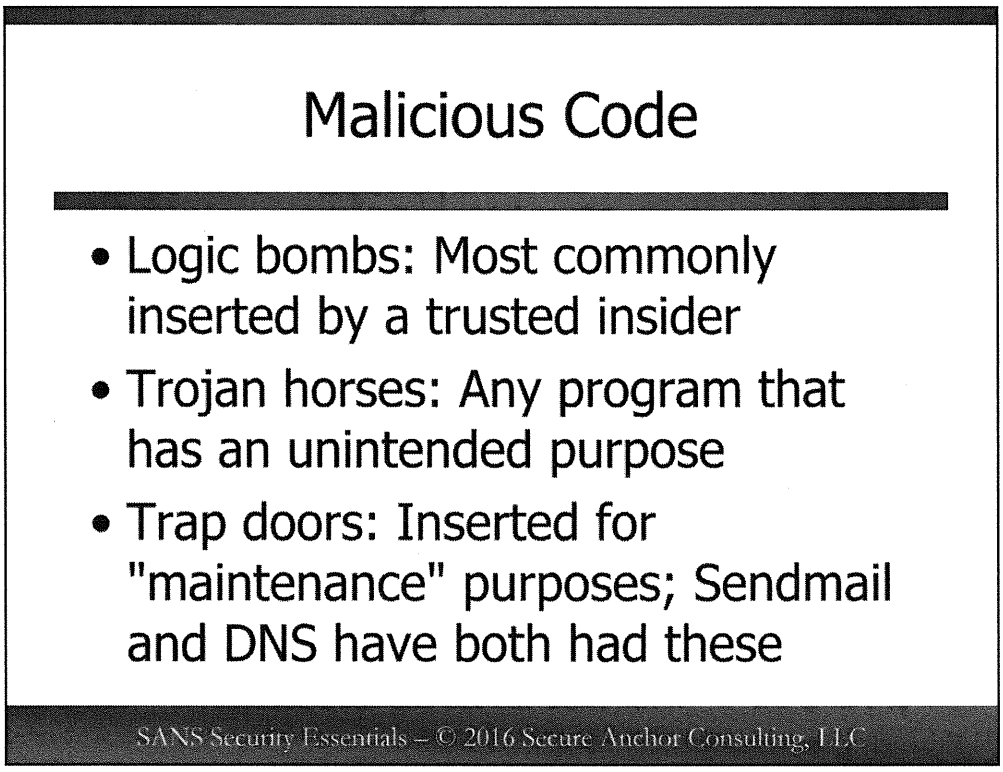
Mitnick Example: Lessons Learned

Having just described some basic prevention and detection techniques, you should now see some overall concepts and guidelines that can easily apply in your environment. First, you can significantly reduce your vulnerability exposure by patching systems as soon as possible, running scanners to detect vulnerable systems in your environment, and disabling services you don't need. You can also add a major element of perimeter (and possibly intra-network) security that can help prevent such attacks with a properly configured firewall. Finally, deploying host-based and network intrusion detections systems can alert you to any possible breaches in security or system anomalies much sooner.



Common Types of Attacks

This section intentionally left blank.



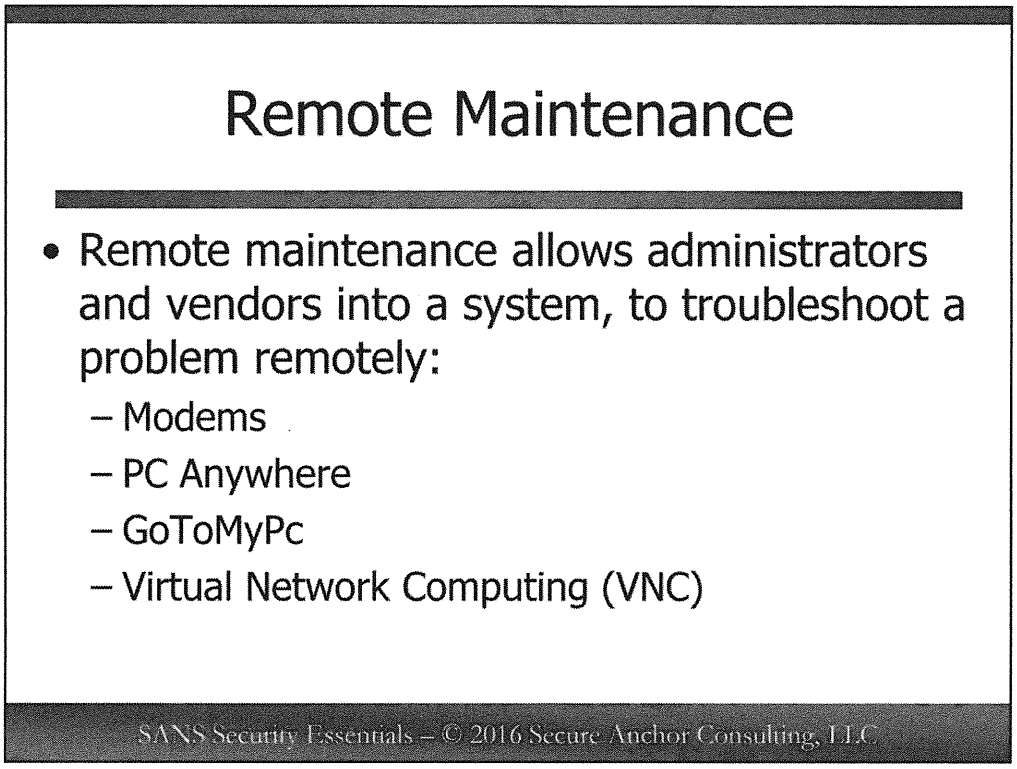
Malicious Code

There are many types of malicious code in the wild today. Logic bombs, Trojan horses, and trap doors are only a small subset of these, but they are fairly common. Logic bombs are small programs or sections of a program triggered by some event, such as a certain date or time, a certain percentage of disk space filled, the removal of a file, and so on. For example, a programmer could establish a logic bomb to delete critical sections of code if she is terminated from the company.

Trojan horses (often just called "Trojans") are often disguised as helpful or entertaining programs (such as operating system patches, Linux packages, or games). Once executed, however, Trojans perform actions the user did not intend, such as opening certain ports for later intruder access, replacing certain files with other malicious files, and so on.

Trap doors, also referred to as "backdoors," are bits of code embedded in programs by the programmer(s) to quickly gain access at a later time, often during the debugging phase. If an unscrupulous programmer purposely leaves this code in or simply forgets to remove it, a potential security hole is introduced. Trap doors can be almost impossible to reliably remove, and often reformatting of the system is the only sure way.

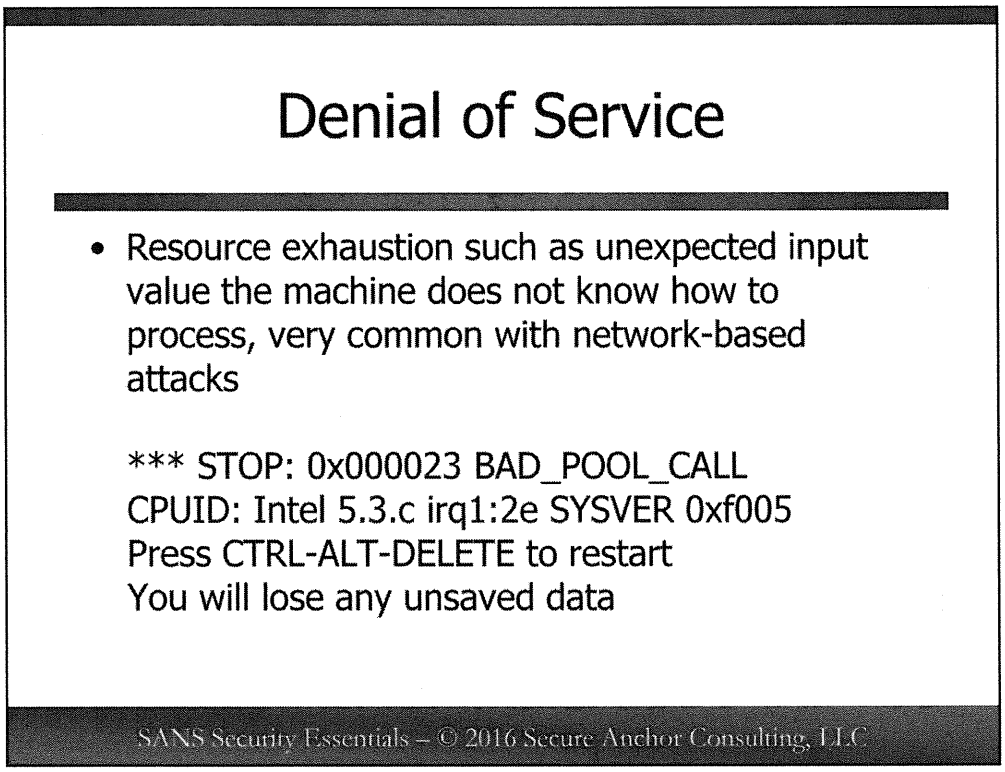
As you read through this section, keep in mind that some of these techniques also can be used against attackers, in a devious sort of way. Administrators sometimes intentionally deploy pseudo flaws, things that look vulnerable to attack but really act as alarms or trigger automatic actions if an intruder attempts to exploit the "flaw." Do not confuse the single pseudo flaw with the concept of a pseudo flaw extended to encompass an entire host or network—often referred to as a "honeypot" or a "honeynet": Neither of these terms properly refers to a single pseudo flaw.



Remote Maintenance

When we hear the term "remote maintenance," we typically think of authorized administrators with the ability to log in from systems while on the road or at home for support reasons. Remote maintenance can also extend to vendors and support technicians who need access to the device to assist in configuration or troubleshooting. In many cases, the remote maintenance tools that are authorized to vendors and support technicians grant a higher level of privilege to the operator than is granted to the administrator, often granting unrestricted access to the operating system versus a standard configuration interface.

The support of remote maintenance is a requirement for many organizations, as well as the support organizations they rely on. Unfortunately, it can reveal weaknesses in the overall security of the network as well. If your support organization has access to the server or appliance platform for maintenance and troubleshooting, what prevents your attacker from accessing the same resource? Some support organizations might require static password authentication to access the remote maintenance services on your devices, and some might even require the use of public key cryptography to restrict access to only authorized individuals. Few support the ability to remotely maintain and expire support passwords, or utilize key revocation mechanisms that would stop a disgruntled employee from abusing his previously authorized access.

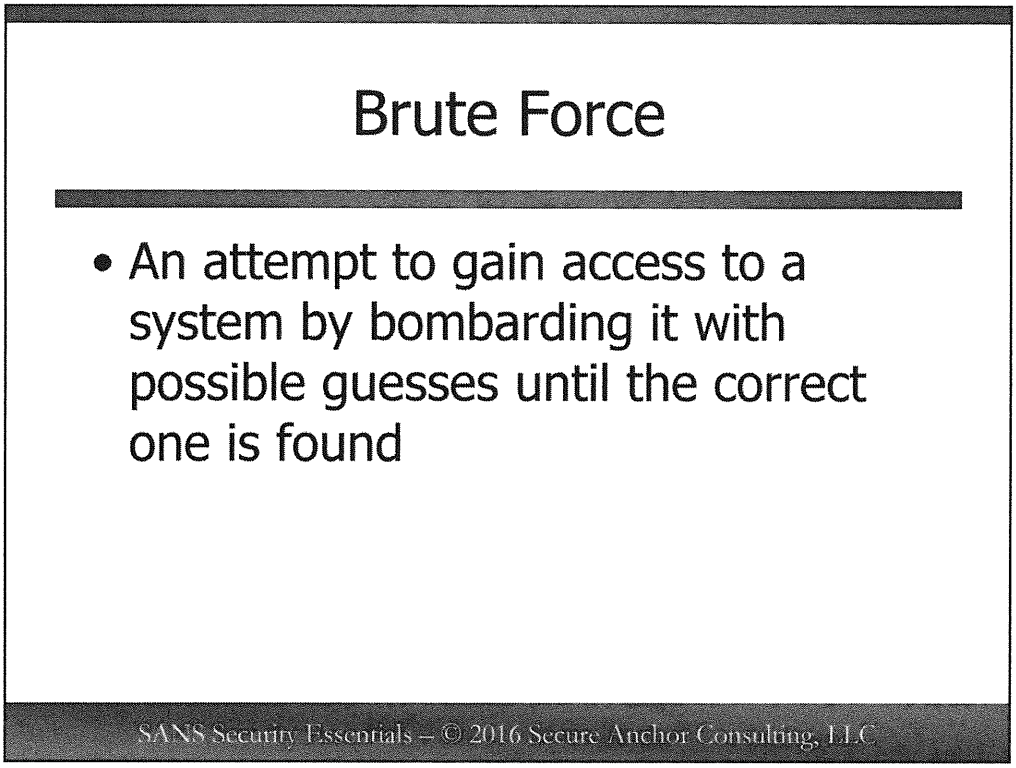


Denial of Service

Denial-of-service (DoS) attacks became quite popular in the early 2000s, but they've probably been around almost as long as computers themselves. As the name implies, a DoS attack occurs when a user is deprived of the use of some data, computing resource, or service due to malicious actions on the part of an attacker. DoS-like conditions also sometimes occur due to a simple error on someone's part, but they usually aren't referred to as "attacks" unless the outcome is intentional.

There are many different types of DoS attacks. The following is a list of some of the more interesting ones, but it's by no means complete:

- **Smurf:** In the Smurf DoS attack, an attacker spoofs the victim's IP address and sends an ICMP ECHO request to the broadcast address of an arbitrary network. When every system on this network responds to the victim, a DoS occurs.
- **SYN floods:** This was used in the Mitnick attack described previously in the chapter. A spoofed IP address is used to send a SYN packet to the target. It then responds with a SYN/ACK that never receives the final ACK to complete the three-way handshake. This hanging connection occupies a portion of the target's pre-established buffer for TCP connections. By filling this buffer to capacity with fake SYN packets, an attacker can effectively prevent the target system from accepting legitimate requests.
- **DDoS attacks:** In distributed denial-of-service attacks, an attacker recruits zombie systems ahead of time to simultaneously release a flood of traffic at a specific target. Several common tools for accomplishing this are TFN, Trinoo, Stacheldracht, and TFN2K. These tools are often found installed on compromised systems, waiting for later use by the attacker.



Brute Force

Imagine that you have a very simple electronic keypad lock on a bank vault. Before you can open the door, you have to type a secret number to pull back the 6-inch (15 cm) diameter deadbolts holding the door closed. If you know the code, there's no problem. You walk up, key it in, and the lock opens, all in a total time of only a few seconds.

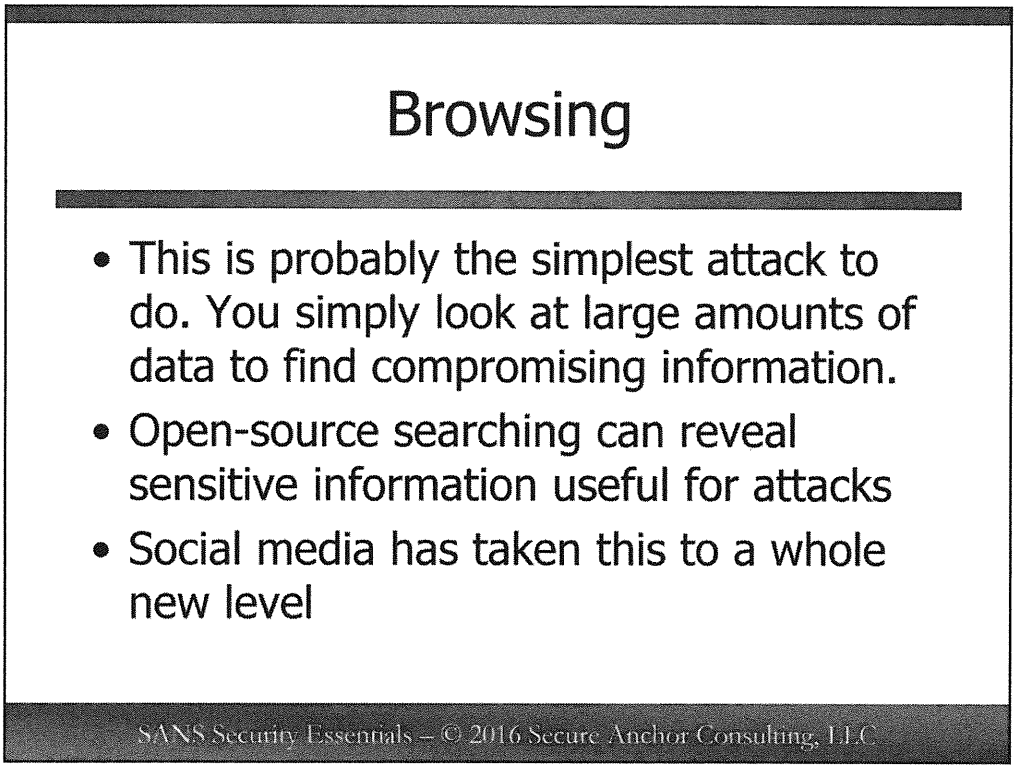
What if you don't have the number? No one is likely to tell you the number just for the asking. Furthermore, you have absolutely no idea what that security code might look like, except that it's entirely numeric (there aren't any letters on the keypad). You don't even know how long it is. It could be a single digit (we hope not!), or it could be 5, 10, 20 digits, or longer. To make things more difficult, let's assume perfect physical security of the lock, door, and the vault itself (an impossibility, we know). Opening the lock is the only way in. Just about the only thing you can do at this point is to walk up to the lock and start punching in numbers, hoping you'll eventually hit on the right one.

Congratulations! You've just mounted a brute-force attack.

Brute-force attacks are probably the least sophisticated available. Their goal is to guess a secret of some sort, like a password or an encryption key. The problem is that they're often nothing more than undirected searches that try every possible combination of inputs until they happen to get lucky. As you can probably imagine, this takes an extremely long time, making it probably the least efficient attack possible.

It's not uncommon for an attacker to connect to some service and repeatedly guess valid usernames and passwords until she gets in. This is a classic, if usually not a very successful, brute-force attack. Retrying a lot of failed logins takes time, though, and most systems impose some sort of limit on failed login attempts. Some even automatically lock the account associated with the failed attempts, preventing the attacker from trying again.

The problem with brute forcing most encryption schemes is that they typically involve extremely long keys. The more bits you add to the key (the key length), the more possibilities an attacker would have to guess on average until she hits the right one. In fact, every bit you add to the key length doubles the number of possible keys. The number of possible keys isn't an absolute predictor of how long it would take to guess the right one, because you might get lucky and get it very early, but it is a good measure to compare the relative strength of different encryption algorithms.

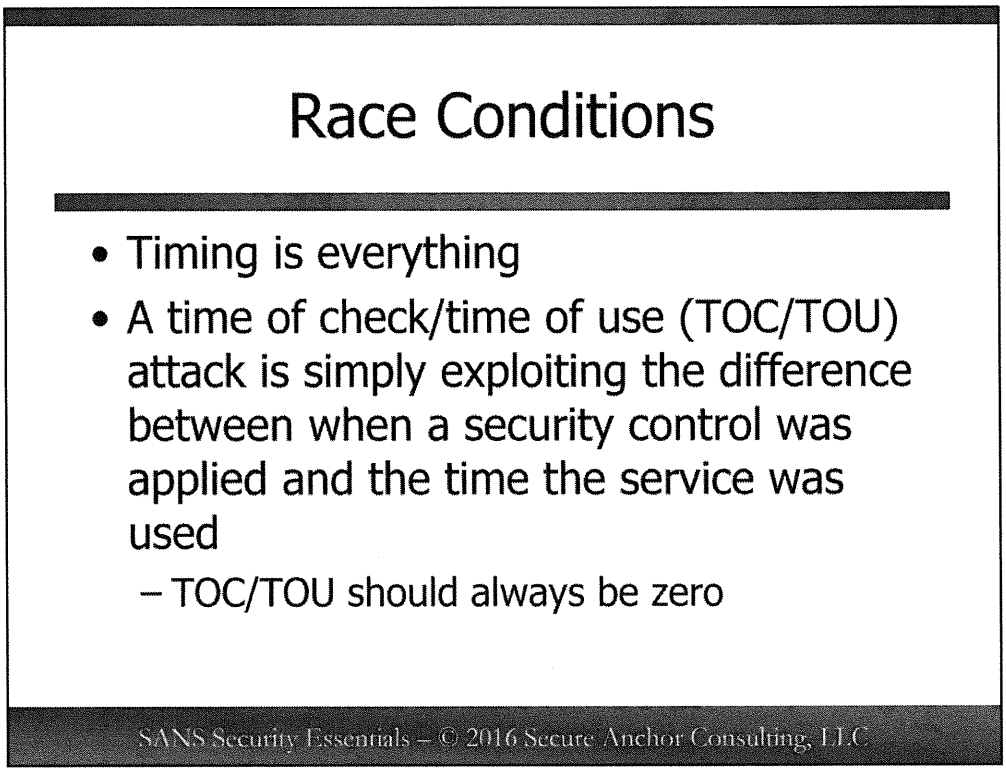


Browsing

An intruder perpetrating a browse attack simply uses the access she already has to look around and see what's "out there." An attacker might not even need to be a local user, especially in the case where a machine offers information to the world through web, FTP, or other public Internet services.

You might think that browsing doesn't really count as an attack because it doesn't involve much in the way of technical knowledge or skill. We hope to convince you otherwise. There's a lot for an intruder to see on a typical system. If he is really lucky, he might find your company's secret business plans or the name and phone number of the CEO's mistress, but even less obviously critical information can be useful.

Most Windows machines let you browse the network to discover file servers, domain controllers, and printers you might be able to access. Under Unix, normal user accounts usually can get a lot of information about printers, file servers, and NIS servers. An attacker might be able to use all this information to map out possible trust relationships. In fact, the first phase of Mitnick's attack began with some simple browsing to see what information Shimomura's systems were willing to freely give away.



Race Conditions

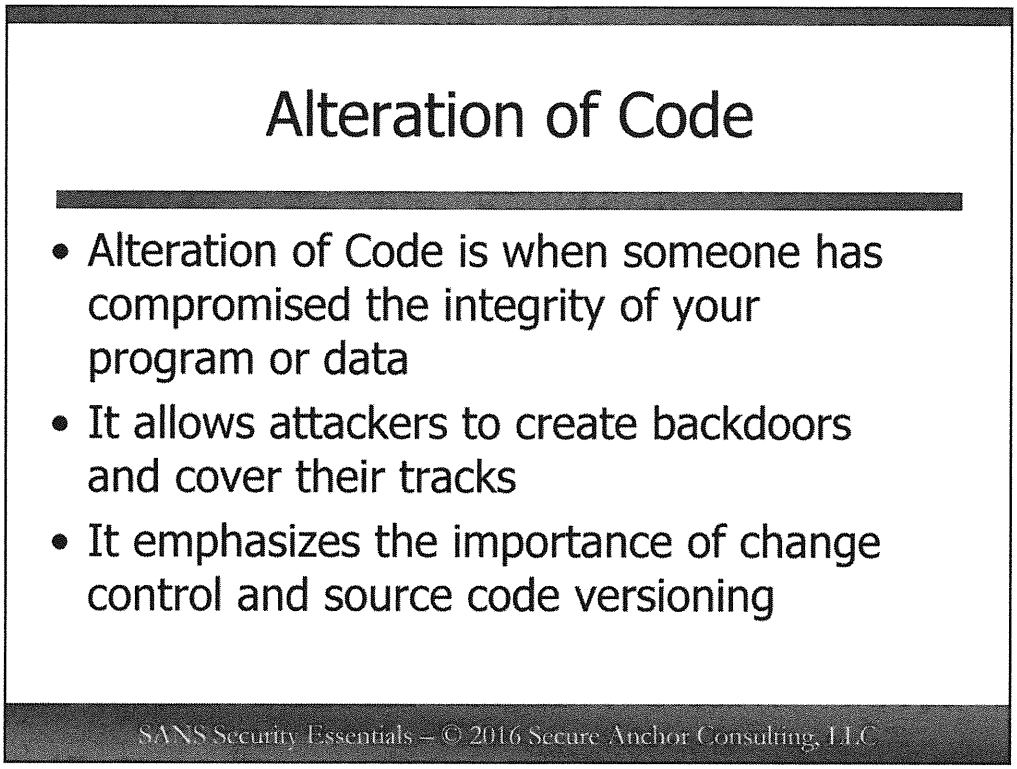
Race conditions exploit that small window of time between when a security control is applied and when the service is used. These usually are very tricky and relatively difficult to pull off. To understand why, think of a common example, such as a script that rebuilds access control lists from a database each night to control who gets to access the payroll system.

Race conditions are also sometimes known as Time of Check/Time of Use or TOC/TOU attacks.

Updating the ACLs nightly is a laudable goal. It keeps the access list up to date, removing employees who no longer need access to the data and adding new ones who suddenly do. The list never gets stale because it is completely rewritten each night from scratch. Paradoxically, this actually can cause a problem if the designer of the system is not careful.

One way to update the list would be to remove the existing ACL, and then go through all the entries in the employee database to find out who has the special "payroll access" flag on their account. Those who do are added to the list. After the list is complete, a new ACL is written.

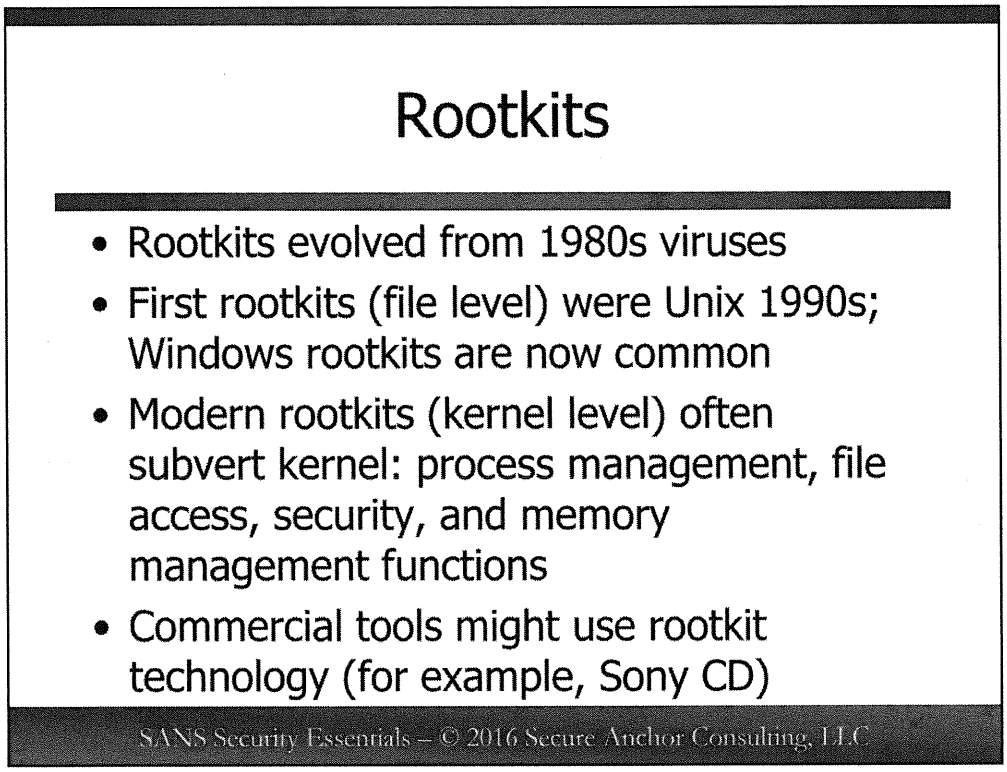
You might have just spotted an obvious point of concern: What happens between the time the ACL is removed and the time when the database search is finished and the new ACL is created? That payroll system is probably wide open for anyone to use. If your employee database is large and takes some time to search, the window of opportunity for an attack could be as much as several seconds, or even a minute or more. That is plenty of time for an attacker to make queries while the database temporarily is unprotected.



Alteration of Code

Alteration attacks are just what they sound like: Someone makes unauthorized modifications to code or data, attacking its integrity. These attacks can take many different forms and have a variety of consequences. Viruses and worms are definitely alteration attacks because they have to modify something on your system in order to work.

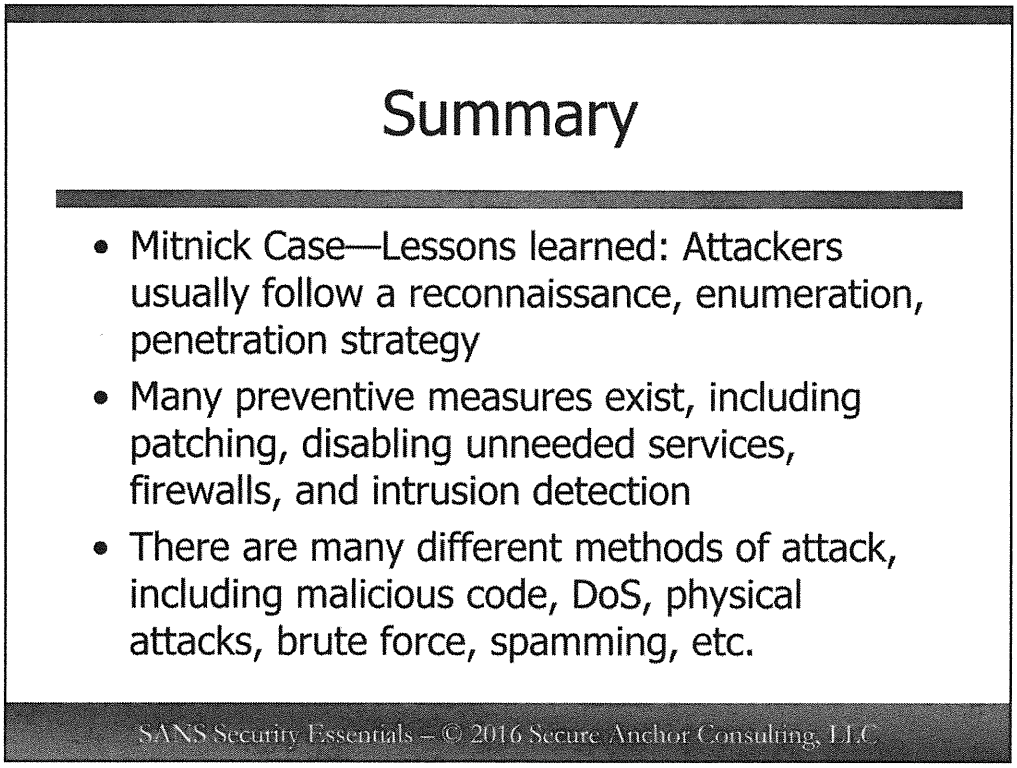
The place where code is altered and not managed is one of the biggest non-attacker driven problems. Maintenance code is often not subjected to scrutiny. Failures to check inputs, removal of security design measures can happen during this period.



Rootkits

One especially troublesome form of alteration attack is the rootkit. A rootkit is a cracker tool meant to be stealthily inserted into the local OS and subvert it so that it does only what the attacker wants it to do. This can be a devastatingly effective form of attack. Think about it: Everything you do on a computer has to go through the OS, including logging in, tracking down rogue files or processes, and performing forensic analysis. If an attacker installs a rootkit, she has total control over anything the OS says or does. She can grant all her processes automatic root privilege, and there's nothing you can do about it. She can make the OS lie to you and hide her processes and files, so you don't even know you have been attacked. Done well, rootkits are bad news.

Rootkits are usually implemented by means of loadable kernel modules (also called "device drivers") under Unix, Linux, and 2000/XP/2003. After code is running the kernel, the rootkit process has access to privileged kernel memory. To build a rootkit for Windows, you will probably need the Microsoft Driver Development Kit for Windows 2003, which allows you to build rootkit drivers for Windows 2000, XP, and also Server 2003. For more information, you can visit http://www.rootkit.com.



Summary

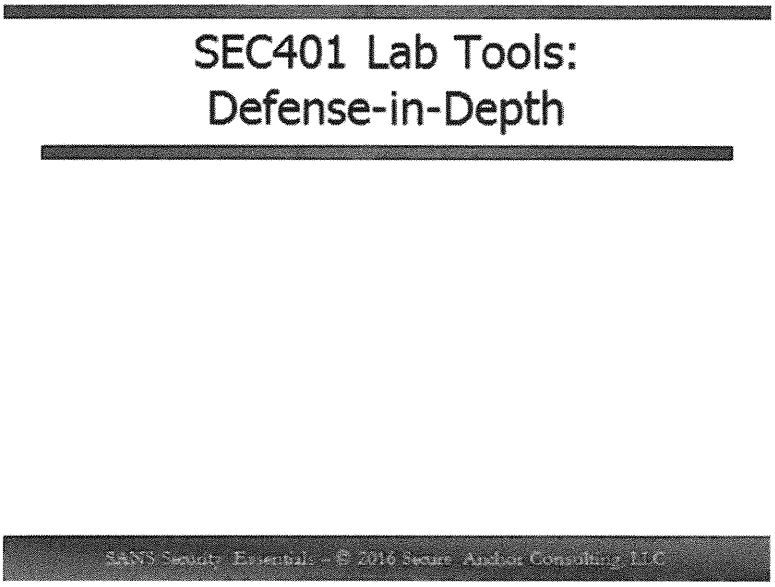
We started this module with a case study, and for good reason. To secure your systems, you have to understand against what types of threats you need to secure them. Thanks mostly to Tsutomu Shimomura's efforts, Mitnick's attack is a well-documented and thoroughly analyzed incident. Studying it allows you to virtually look over the shoulder of a skilled attacker and observe him as he does his dirty work, a valuable experience for any security professional.

His attack started, as most well-planned attacks do, with some basic intelligence gathering techniques. In this case, Mitnick's goal was to map out the patterns of user logins between the systems on Shimomura's LAN. With that information in hand, Mitnick was able to infer the trust relationships between the various machines and exploit them using a previously theoretical technique known as "IP spoofing." The purpose of his spoofing attack was to create on Shimomura's X terminal a backdoor, which allowed Mitnick to log in as root and to carry out an alteration attack by installing a kernel module that allowed him to hijack an existing login session to the target system. Once there, he browsed until he found the cellular phone software he was looking for and downloaded it to his own system. Along the way, he used his access to modify the system logs to try to hide his activity, which ironically was Shimomura's first tip-off that something was wrong.

Of course, there are a number of risk management techniques Shimomura could have used to help prevent attacks like this one. For example, had Shimomura deployed a firewall to protect the perimeter of his network, Mitnick's attack probably would have been foiled easily. Other techniques available today include disabling the Berkeley r* commands and replacing them with secure SSH equivalents, deploying some sort of host- or network-based intrusion detection system, and using periodic vulnerability scans of our own networks to identify and close potential problems before attackers like Mitnick can take advantage of them.

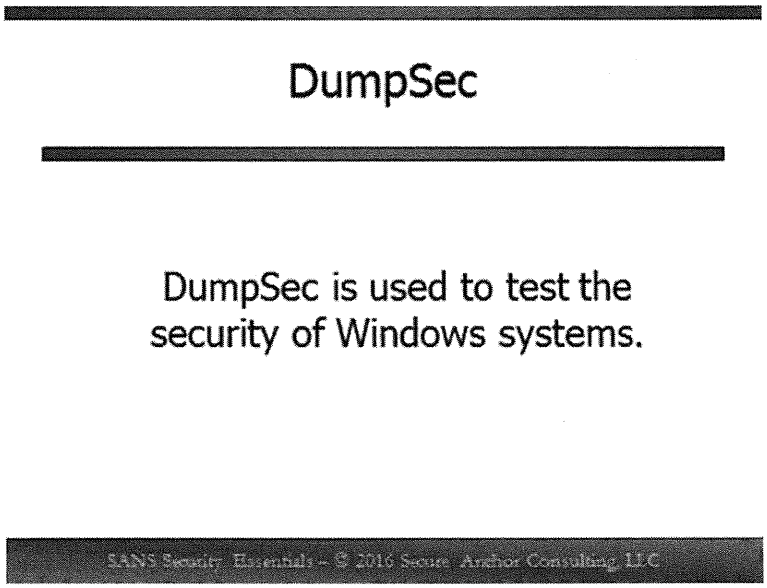
To give you a better understanding of some of the other risks your systems face everyday, we ended this chapter with our own informal list of some of the more common categories of attacks. Its purpose is just to give you a taste of the possibilities and to drive home the fact that most systems have many different points of attack.

There are so many attack methods, in fact, that you really can't rely on just one or two preventive measures to protect yourself. You need to follow the entire risk management cycle, from prevention, through detection, all the way to response, and back again to the beginning. You should use several distinct countermeasures and address each phase of the cycle somehow. Only in this way can you hope to manage the risks associated with sophisticated attacks like Mitnick's. If it can happen to someone like Shimomura, it can happen to you.



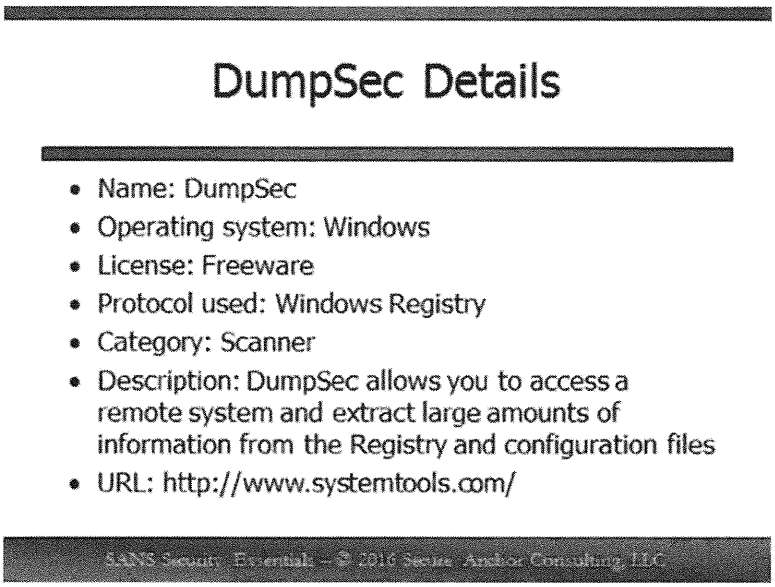
SEC401 Lab Tools – Defense-in-Depth

This section intentionally left blank.



DumpSec

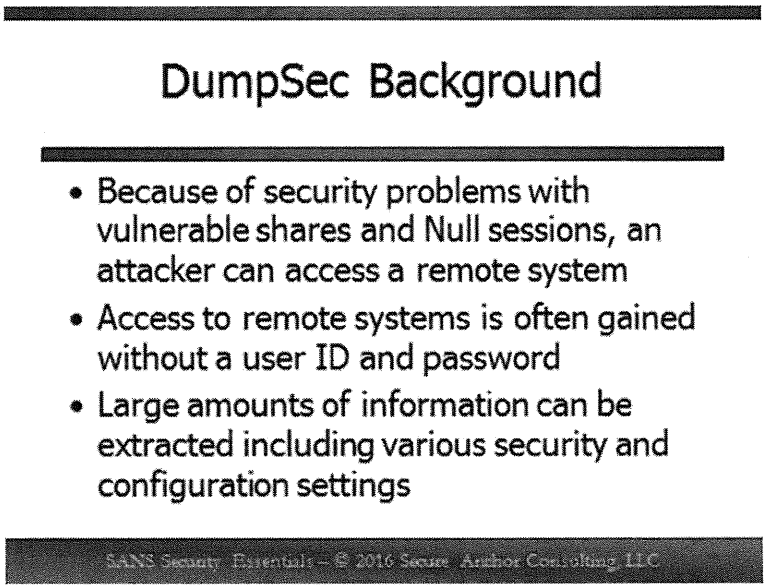
When auditing a large number of Windows boxes, it is critical to have automated tools to continuous check the security of a system and make sure there are no signs of a compromise. Therefore, security personnel require tools to reduce the amount of manual information gathering. DumpSec provides a graphical interface to gather some of this information. Included in DumpSec is the capability to dump configuration statistics from remote hosts, such as access control lists (ACLs) on the registry, shares, users, groups, rights, and services. In addition, DumpSec is free, and the installation and operation of it is effortless.



DumpSec Details

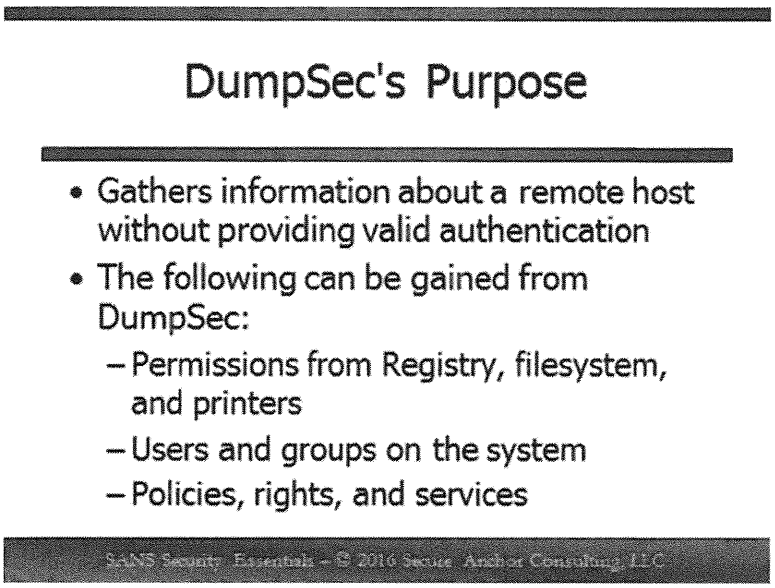
The following topics and action items are covered in this chapter:

- Installing DumpSec
- Running and using DumpSec



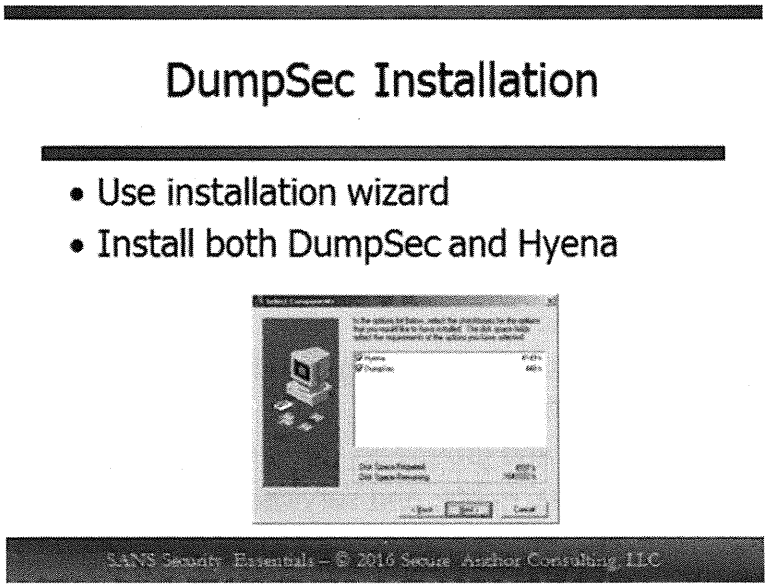
DumpSec Background

This section intentionally left blank.



DumpSec's Purpose

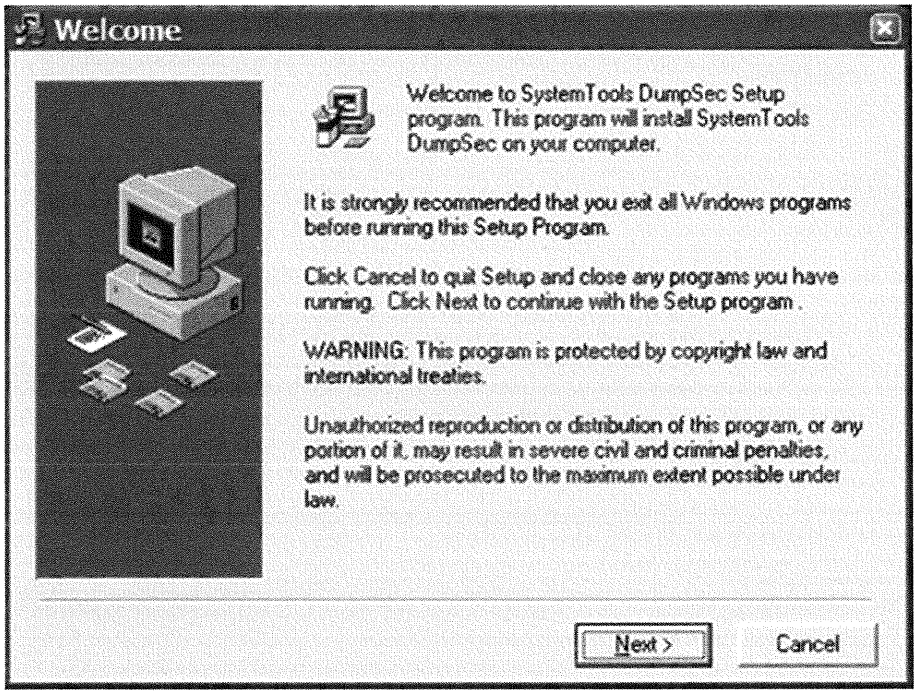
This section intentionally left blank.



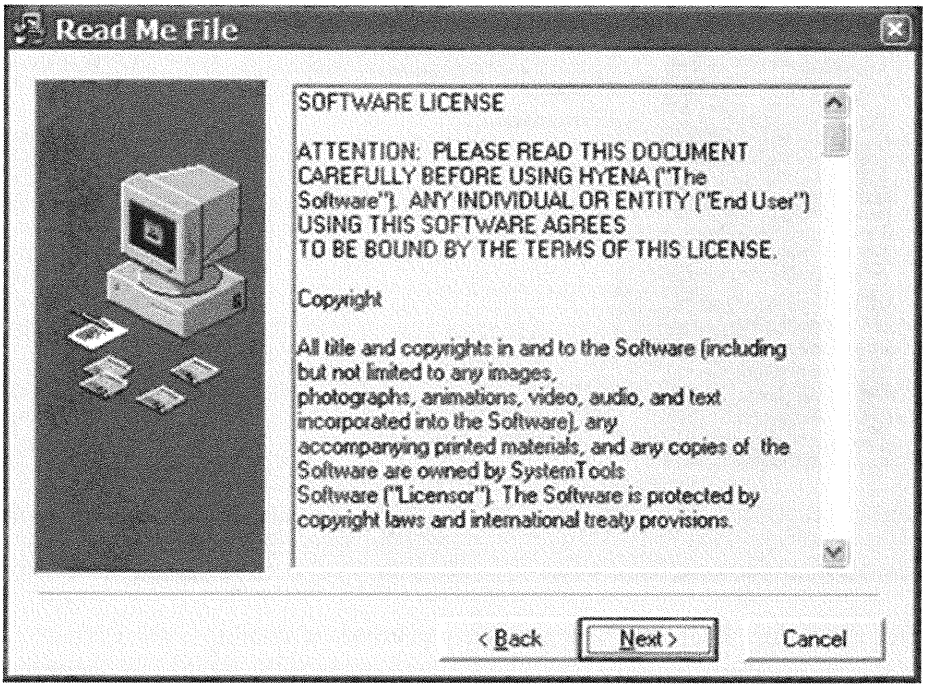
DumpSec Installation

The following steps explain how to install DumpSec:

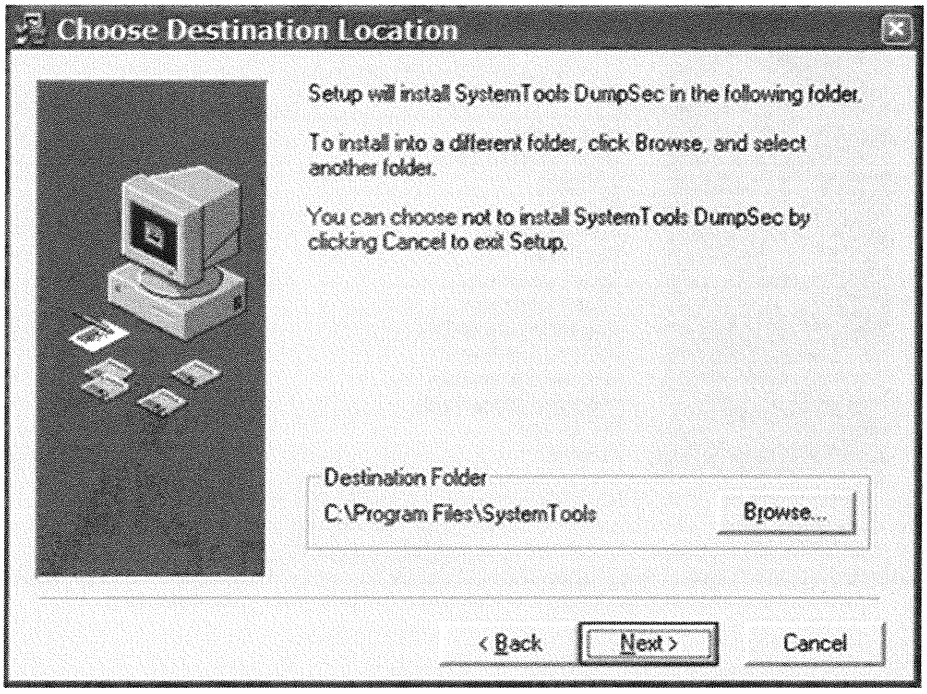
1. After unzipping the installation executable, double-click *SystemTools.exe* to start the installation program. After the Welcome window displays, click *Next*.



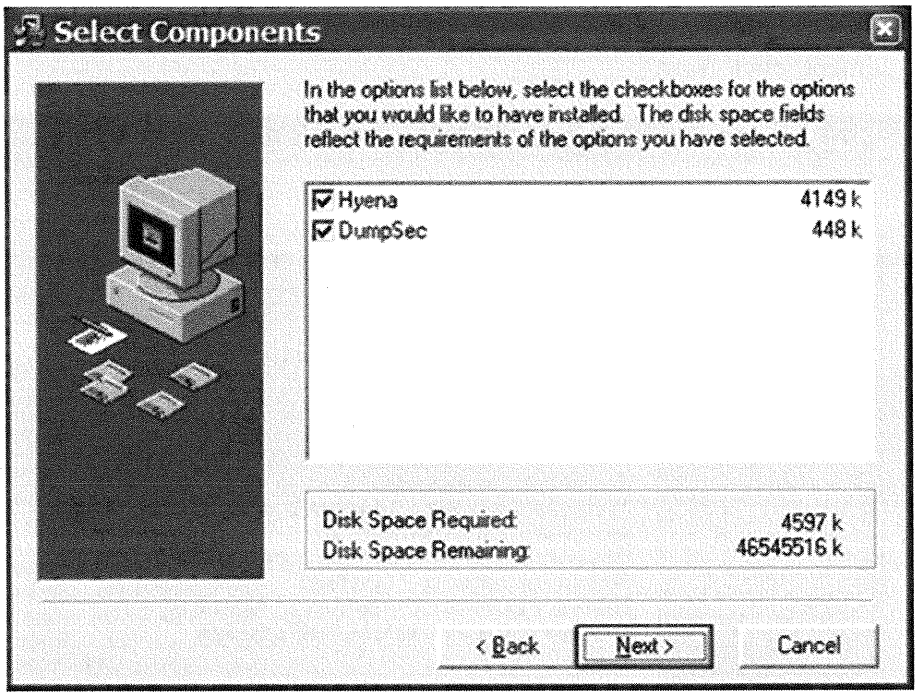
2. The Read Me File window displays. After reading the Software License agreement, click *Next*.



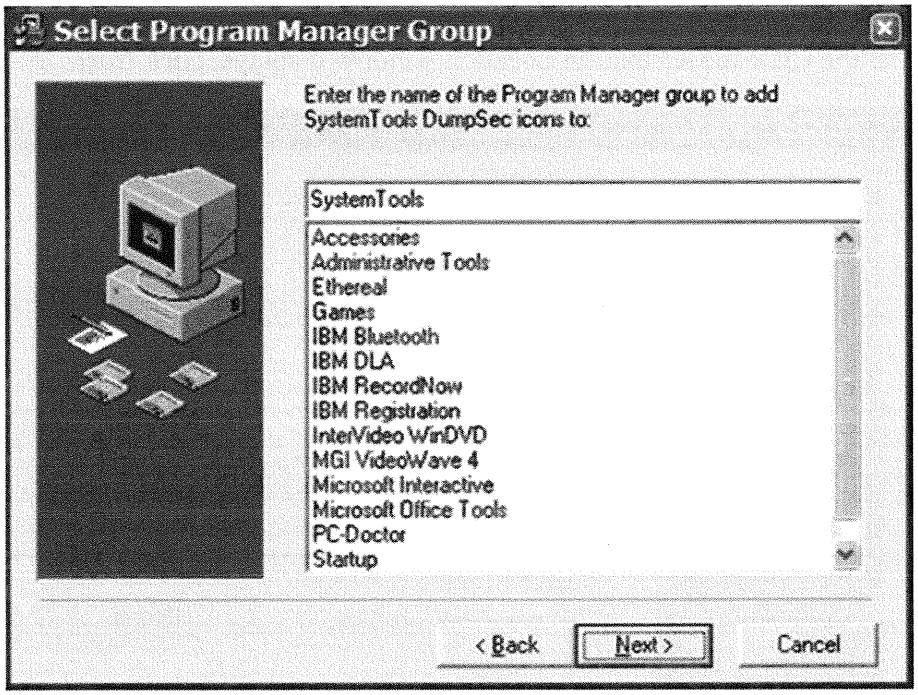
3. After the Choose Destination Location window displays, click *Next*.



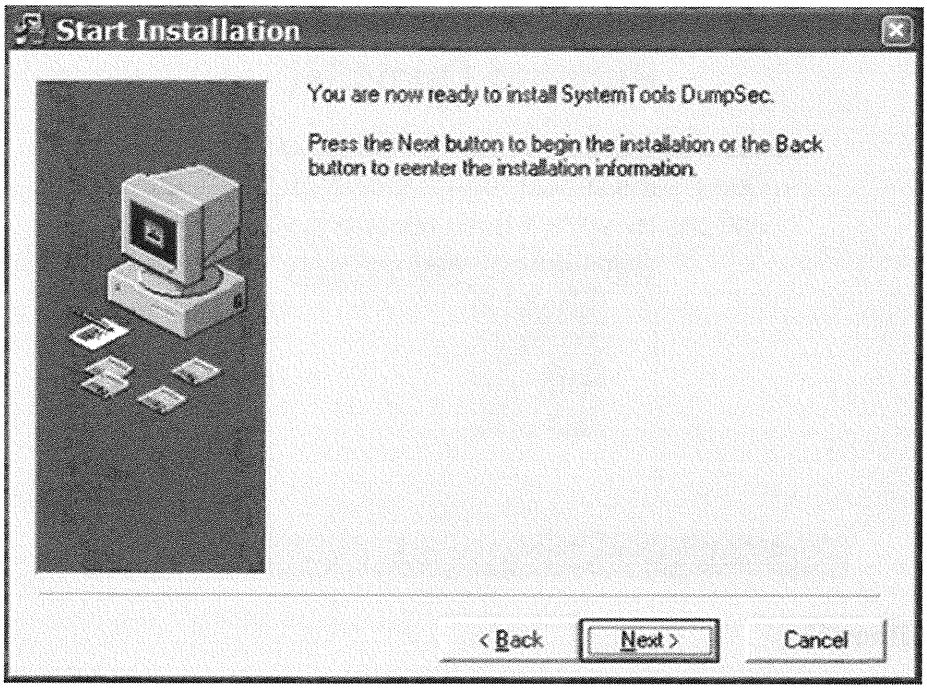
4. After the Select Components window displays, click *Next* to install both DumpSec and Hyena.



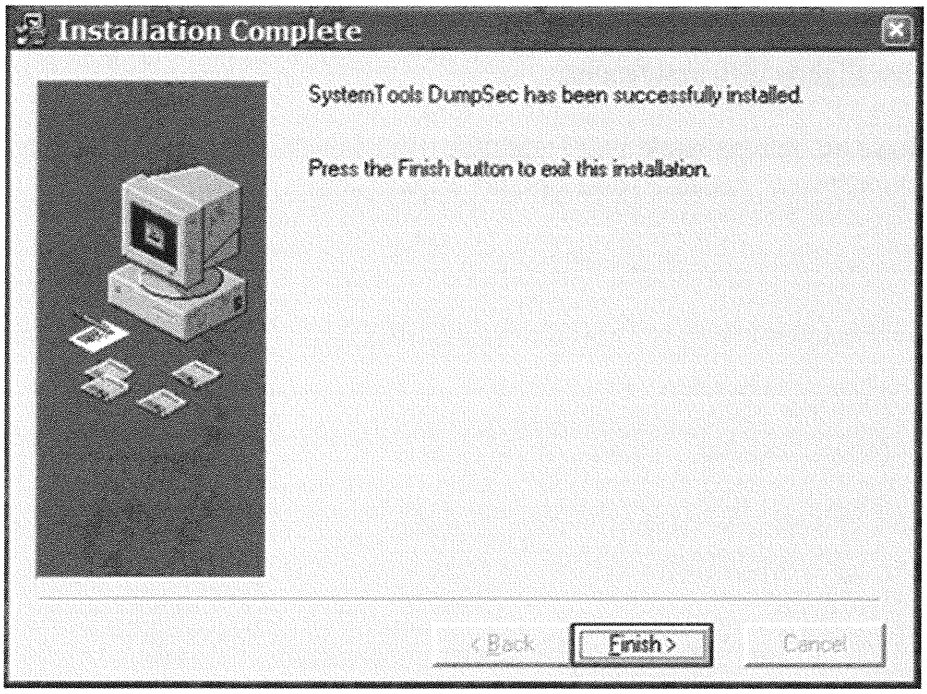
5. After the Select Program Manager Group window displays, click *Next*.

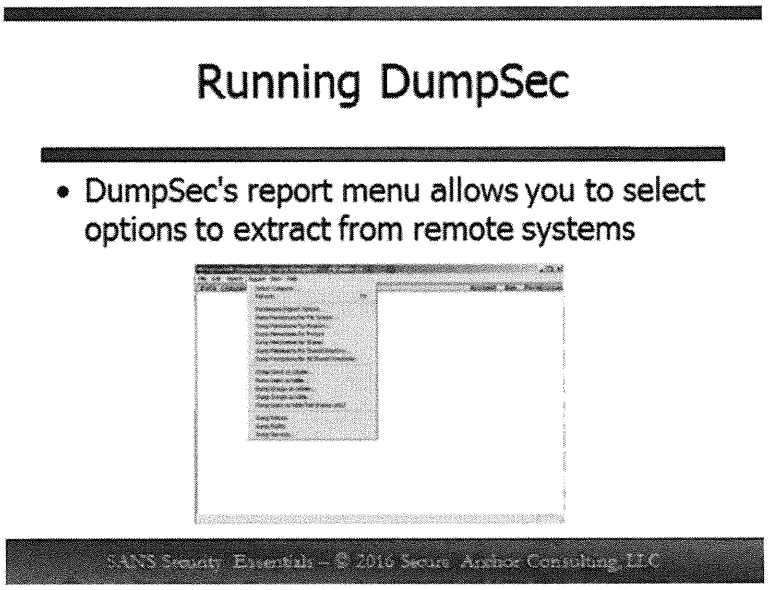


6. After the Start Installation window displays, click *Next* to start the installation process.



7. After the Installation Complete window displays, click *Finish* and you are done!





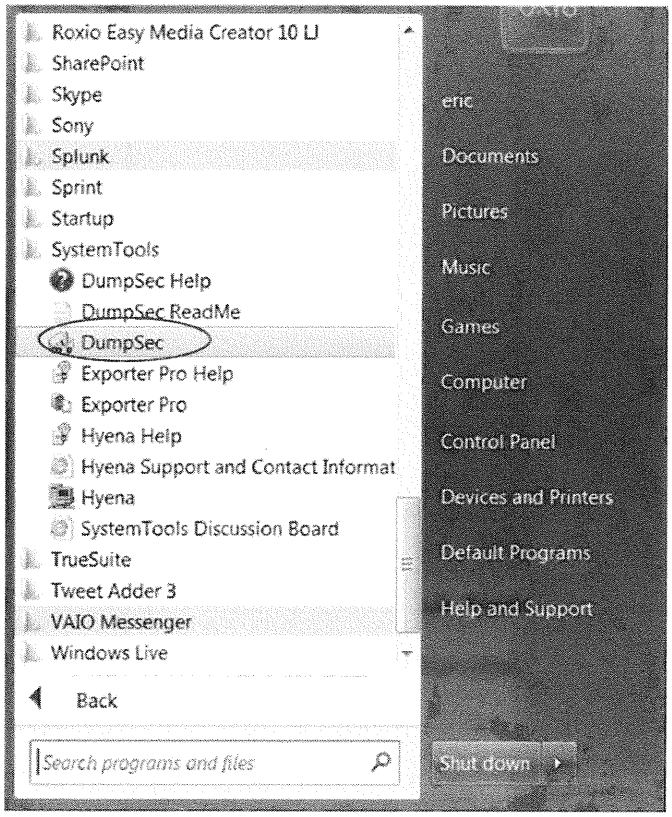
Running DumpSec

Getting results from DumpSec is as easy as installing DumpSec. This section examines the reporting options and some of the ways to make DumpSec more efficient.

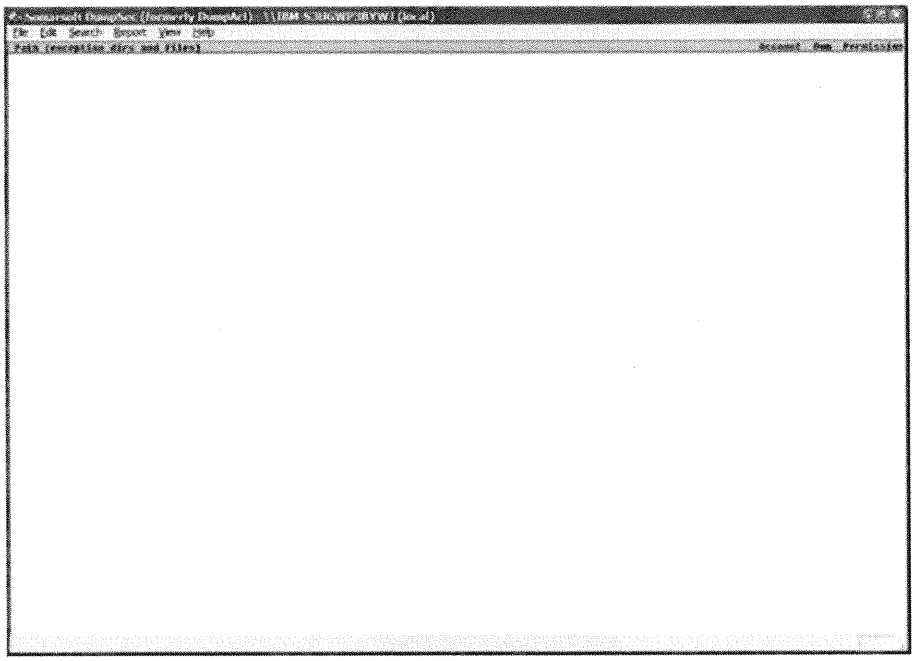
Note: Depending on how your Windows 8 system is configured, some options might not work or give slightly different results.

Note: To have all options work, exit DumpSec. Right-click the *DumpSec* icon and click *Run as Administrator*, because some options require admin access.

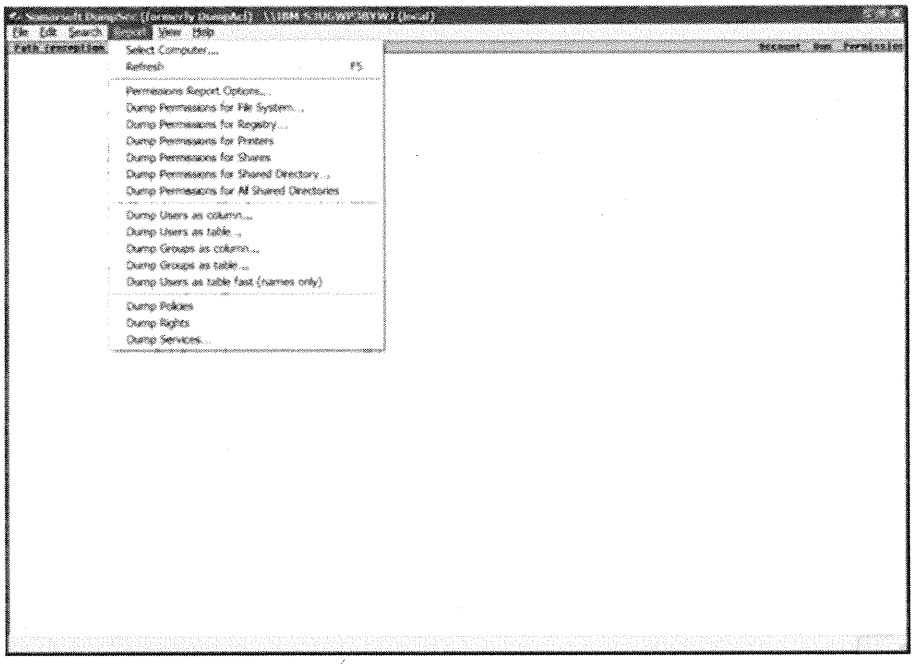
1. To start DumpSec, Click *Start*, *All Programs*, *System Tools*, and *DumpSec*.



2. After clicking DumpSec from the menu, you are presented with the DumpSec interface, as shown in the following screen.

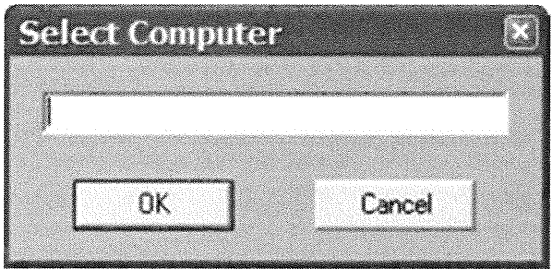


3. The Report drop-down menu contains most of the options for DumpSec. From here, you can select a remote computer to access, as well as dump various types of configuration information from the selected Windows host.



Connecting to Remote Computers

Normally in a network environment, you would type the IP address into the Select Computer dialog box from the Report menu, as shown in the following screen. For this exercise, when you open DumpSec, it automatically connects to the local computer. This can be seen by looking at the top of the screen. Because you are not connected to a network, you will not be able to enter an IP address. Therefore, at this point, you should click *Cancel* from the select computer screen because you are already connected to the local computer automatically.

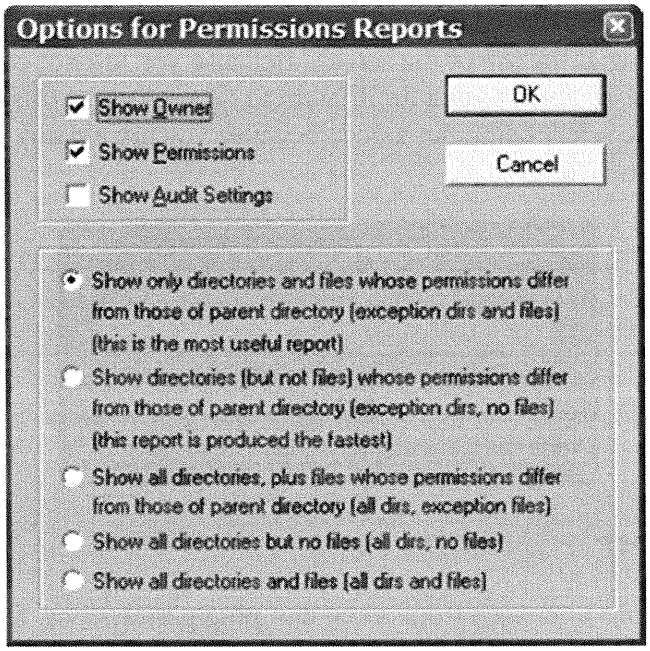


Dumping Permissions

There are a few options to choose from when dumping permissions. Some of the options depend on the following:

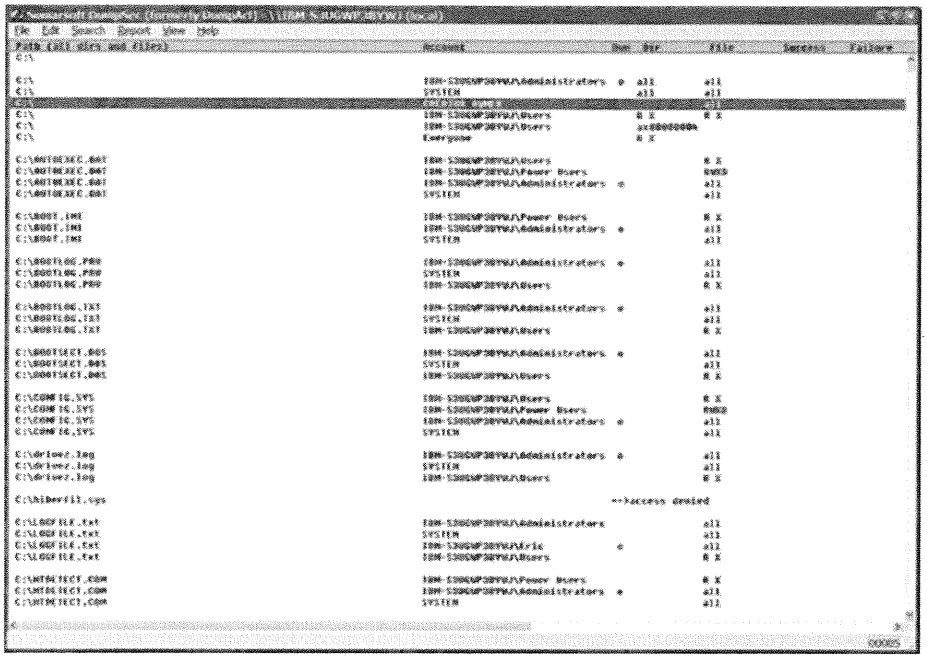
- Whether or not you want to see files and directories
- Whether or not you want to see directories, but no files
- Whether or not you want to see directories and files whose permissions differ from those of the parent directory.

To choose options based on the level of detail you desire, from the Report menu, click *Permissions Report Options*. The Options for Permissions Reports dialog box shows the various options you can choose.



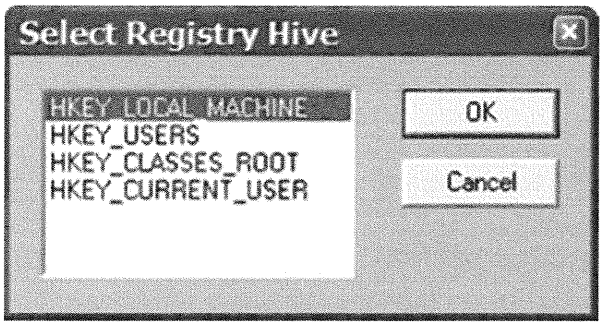
Click *OK*. Now that you have set the options, you are ready to extract information from your local system. From the Report menu, click *Dump Permissions for File System* to display a pop-up window asking you to select what directories you want to dump the information for.

For this exercise, click *c:* and click *OK*. This might take a minute or two for the system to process the directories but when it is done, the following screen is displayed.

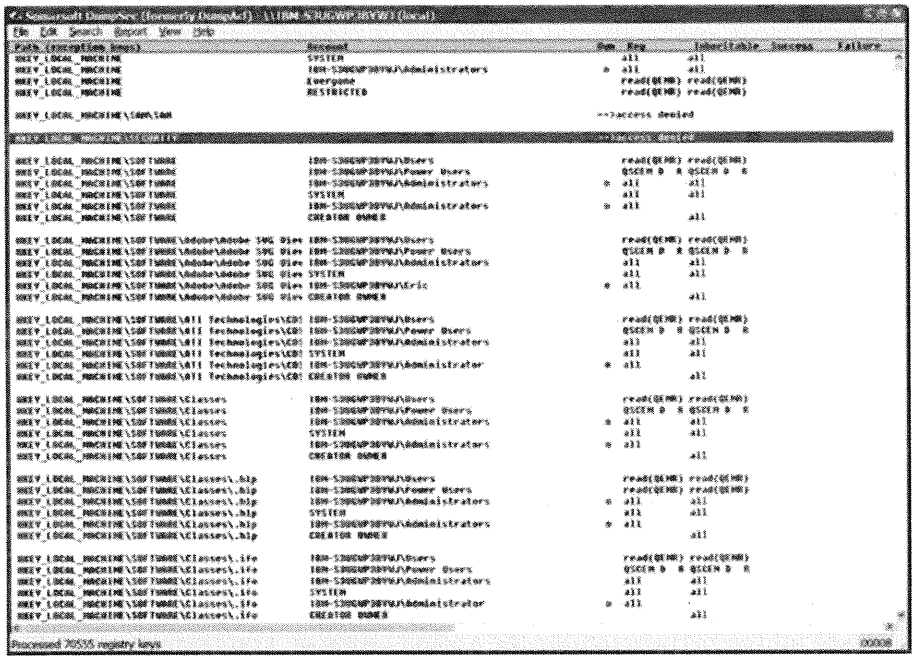


Now you can see all the permission for the various files on your system.

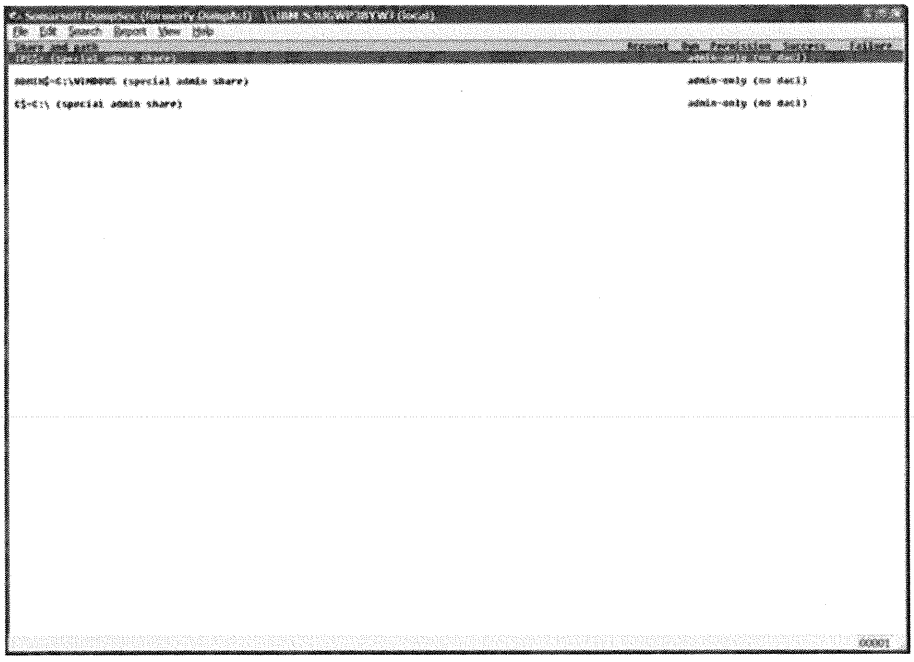
When dumping information from the registry, you can select the hive that you want to dump. First, from the Report menu, click *Dump Permission for Registry*. By selecting this option, you are presented with the screen where you can choose which hive you want to dump.



Clicking the default *HKEY_LOCAL_MACHINE* hive provides a similar dump as the previously shown file system option; however, in this case, registry paths are used. Again, this can take several minutes, but if you look at the bottom of the DumpSec screen, you can see that registry keys that are being processed.



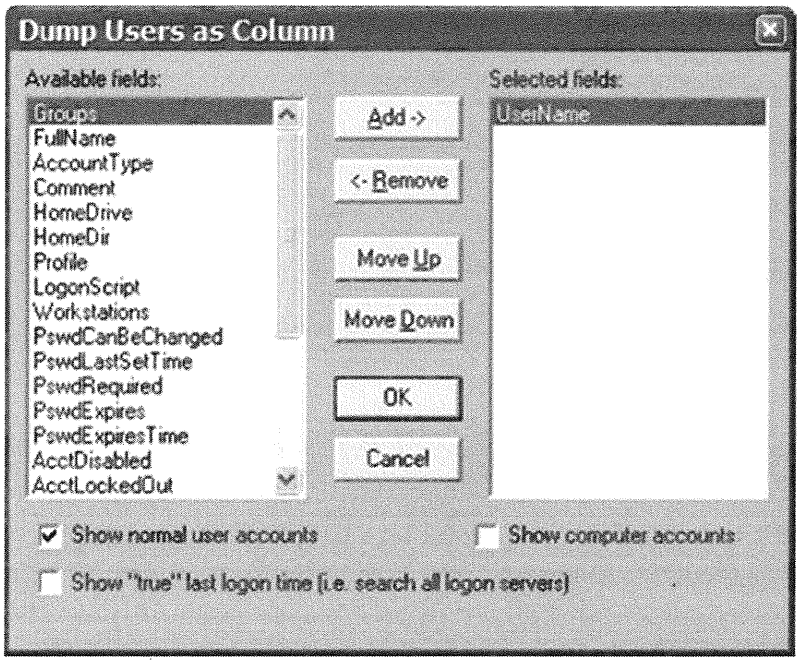
From the Report menu, the Dump Permissions for Shares option provides a report of all the shares and who has permission to access them.



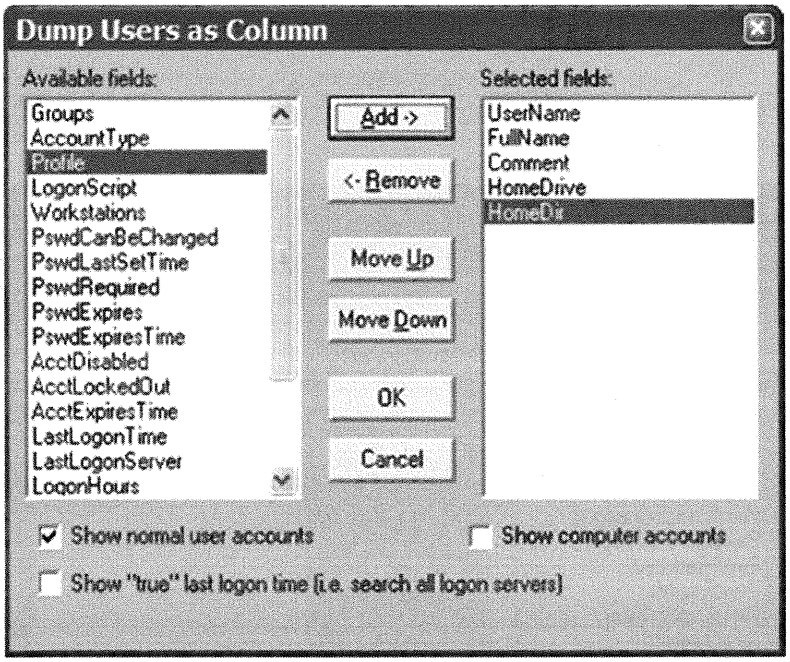
Dumping User Information

Managing user accounts is not easy when you are working with a large number of users. DumpSec provides two methods for dumping user information—the Dump Users as column report and Dump Users as table. These options can be selected from the Report menu.

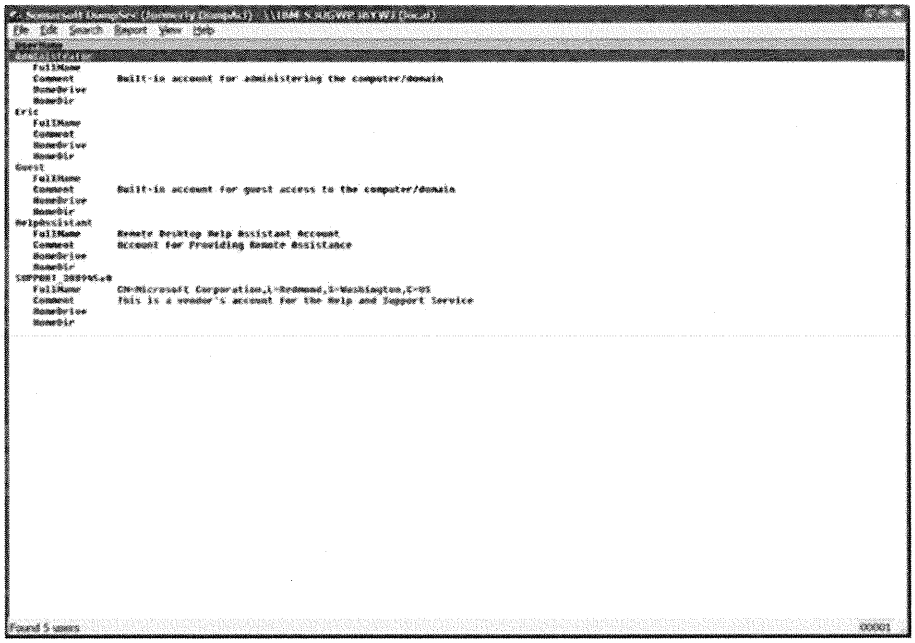
The Dump Users as column report provides a list of fields that can be incorporated into the report.



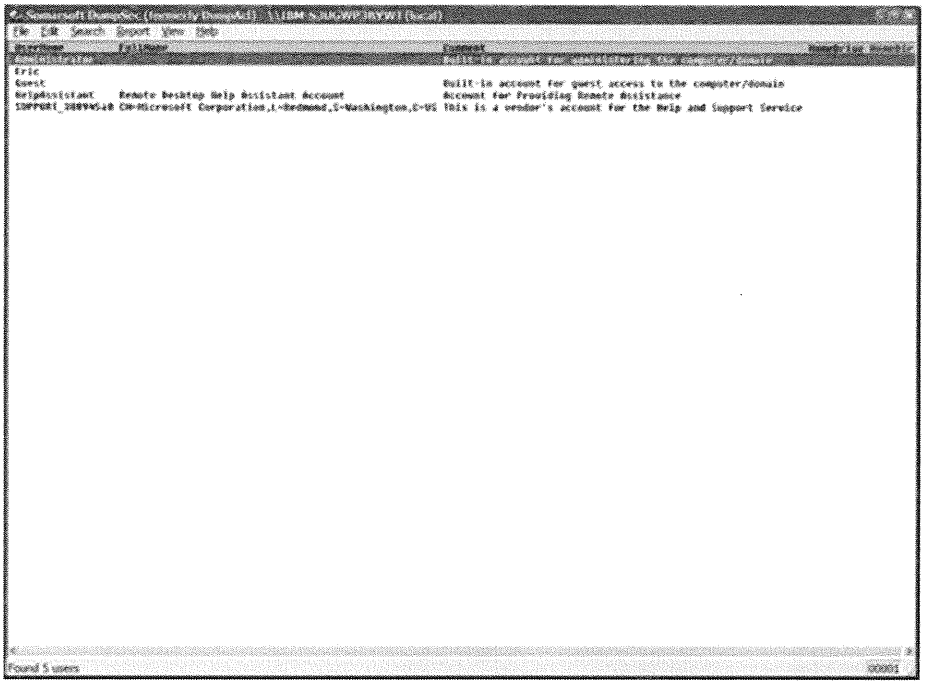
You can add fields that are displayed by selecting them on the left-hand side and clicking the *Add* button. From the left-hand column, click *FullName*, *Comment*, *HomeDrive*, and *HomeDir,* and then add them to the right-hand column as shown in the following screen. When you are done, click *OK*.



The Dump Users as column report is useful for extrapolating the groups to which a specified user is a member. A quick review of user dumps provides useful information. For example, test accounts with high privileges and poor passwords are often exploited as a way to get into a system.

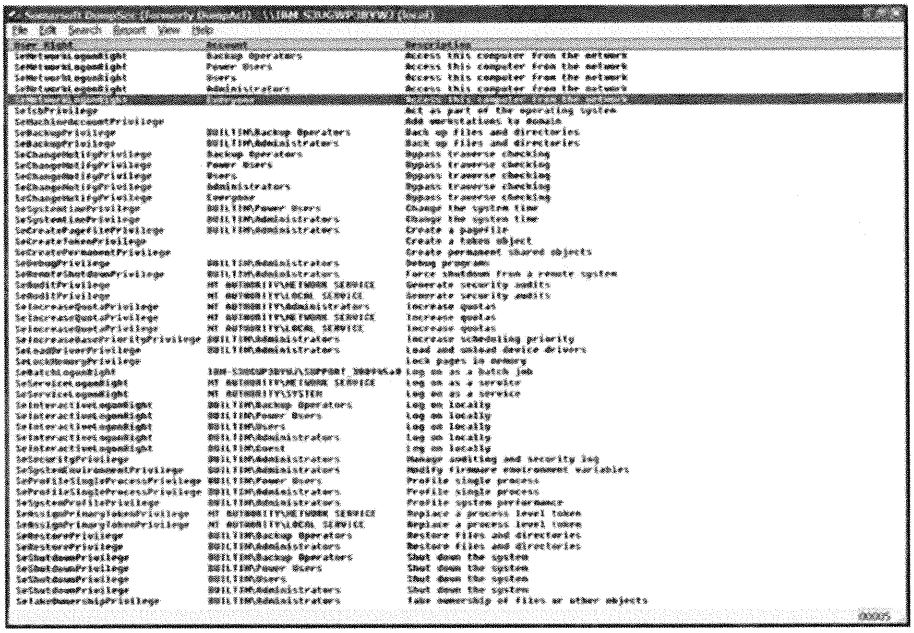


If you don't like the format of the Dump Users as column report, you can use the Dump Users as table report. This report displays the same information in a different format or view. After the Options screen displays, select the same options that you previously used.



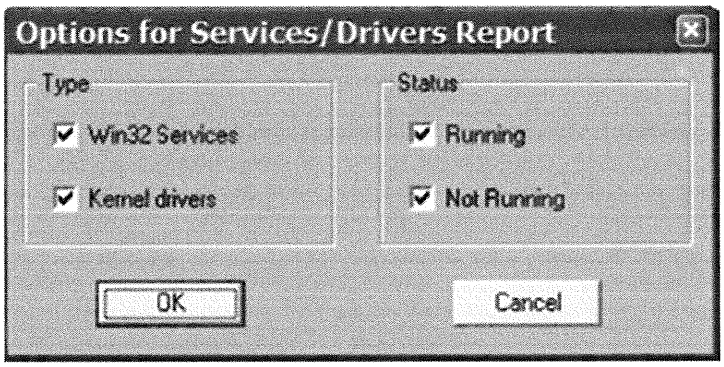
Dumping Rights

When auditing a system, it is important to know who has certain rights. One of the goals of security is to give someone the least amount of access he or she needs to do the job, which is often referred to as a principle of least privilege. To make sure someone has the minimal amounts of rights, you need to be able to audit him or her on a regular basis.

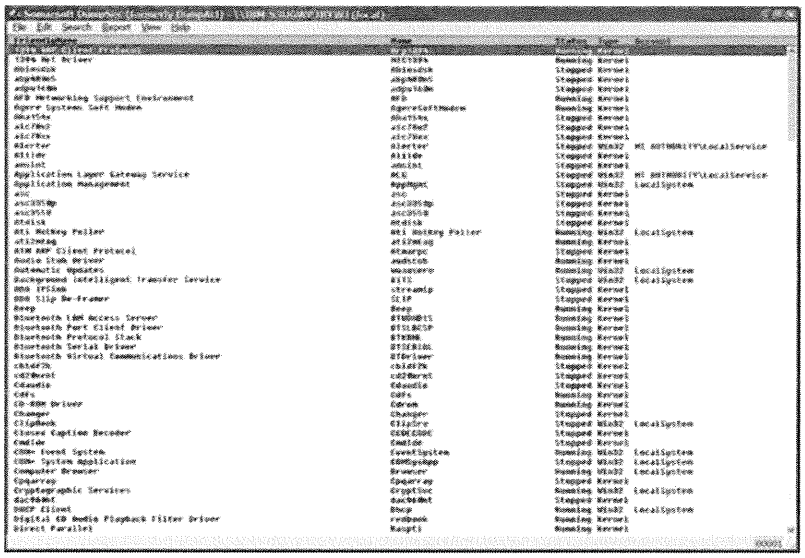


From the Report menu, click the *Dump Rights* option, which gives you a full listing of the rights across the system.

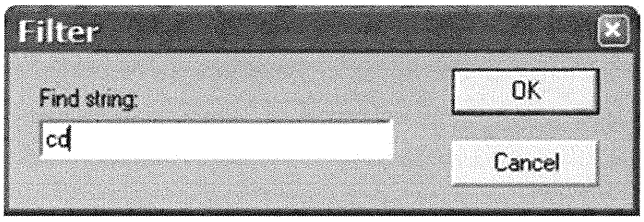
To review the Dump Services report, from the Report menu, click *Dump Services*. After you click this item, the following options display. Accept the default settings and click *OK*.



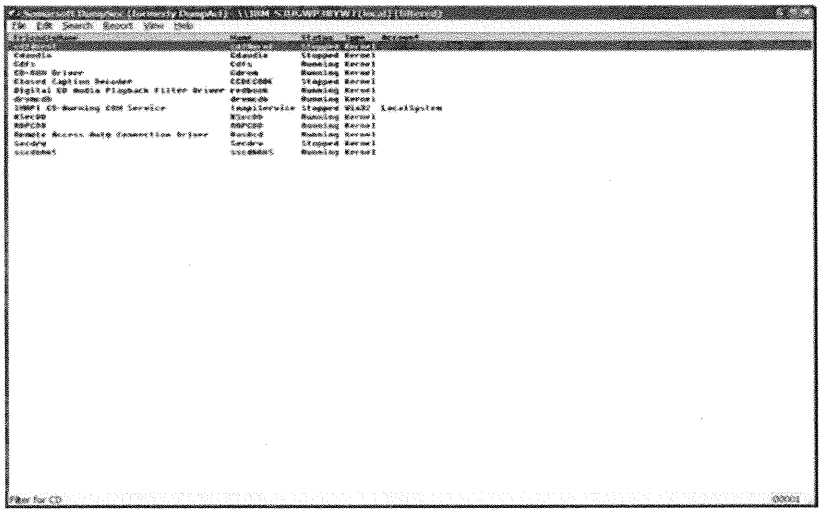
After you click *OK*, a listing of the services displays.



Another great feature included in DumpSec is the capability to filter results based on a keyword. For example, the result of the Dump Services report contains a lot of information. To show the services with the string cd, you can perform the dump, and then filter the results. To do this, from the Search menu, click *Filter*. In the Filter String dialog box, type **cd** and click *OK*.



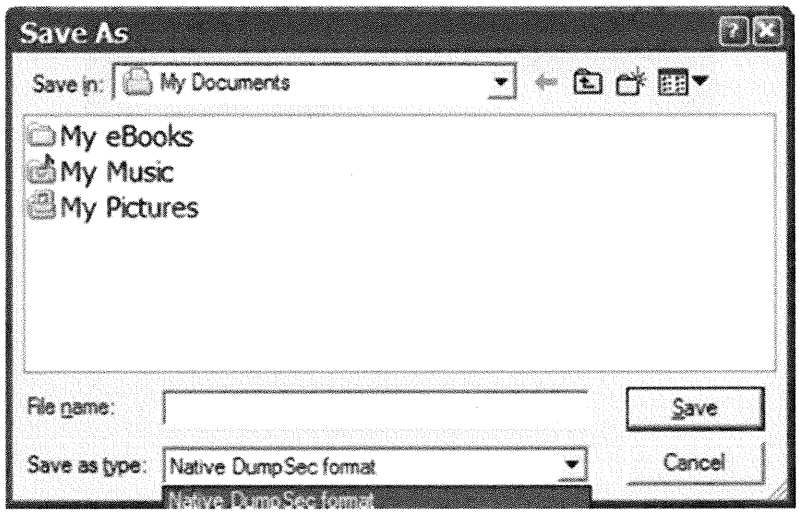
The results are shown in the following screen.



Filtering results provides you with what you want to see with little filler. It is an excellent way to get the data that you need quickly.

Saving Reports

The File drop-down menu provides an option to save reports in various formats. Click *File*, *Save Report As*, and then choose the type of format for your report, as shown in the following screen.

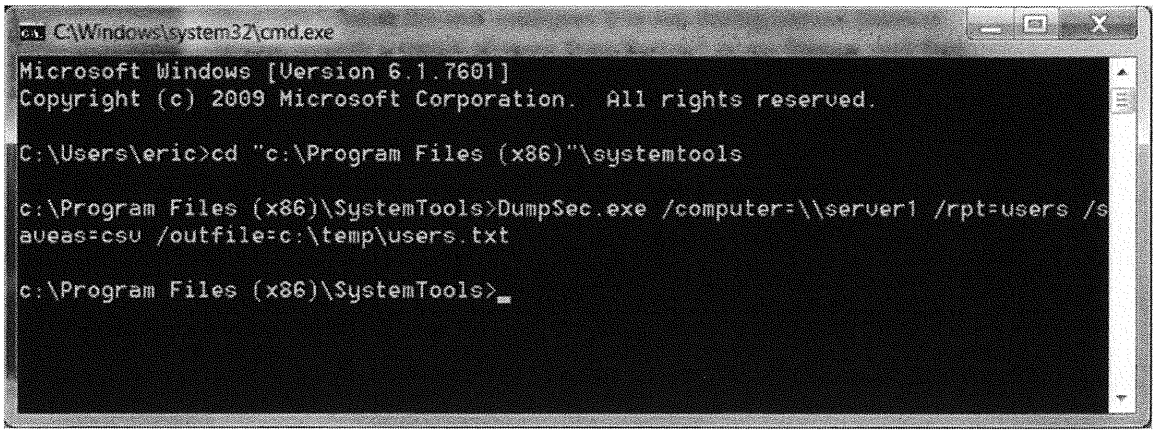


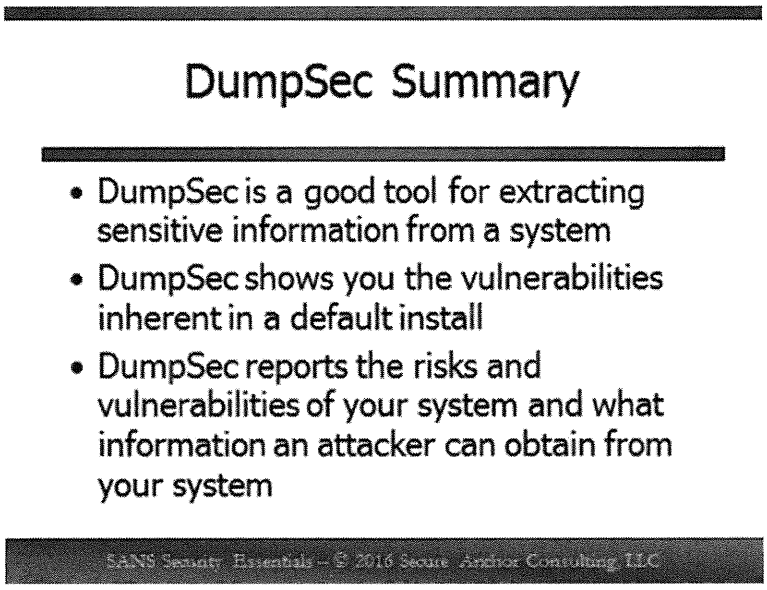
If gathering information one system at a time is not for you, DumpSec accepts command-line options that can help automate information gathering. A full listing of command-line options is provided in DumpSec's help file. For example, the following option is available:

DumpSec.exe /computer=\\server1 /rpt=users /saveas=csv /outfile=c:\temp\users.txt

You can enter this string into a batch file and configure it to run during certain times or days. This particular string outputs a report of users from Server1 in csv format, and then saves it to a file called "users.txt," which is located in c:\temp.

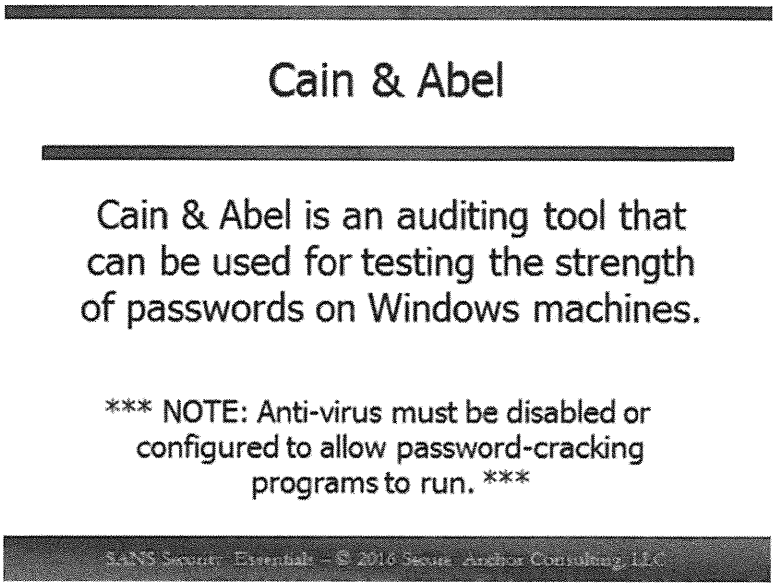
Note: Remember, if you did not add the DumpSec directory to your path, you would need to cd to that directory to run the command.





DumpSec Summary

As you can see, DumpSec provides you several opportunities to gather information and save that information into useful reports. Whether you dump from a local or remote host, or whether you save or filter dumps, DumpSec allows you to work more efficiently. As stated previously, it's important to know your system and to know the vulnerabilities of that system. DumpSec's reporting options allow you to know your system.



Cain & Abel

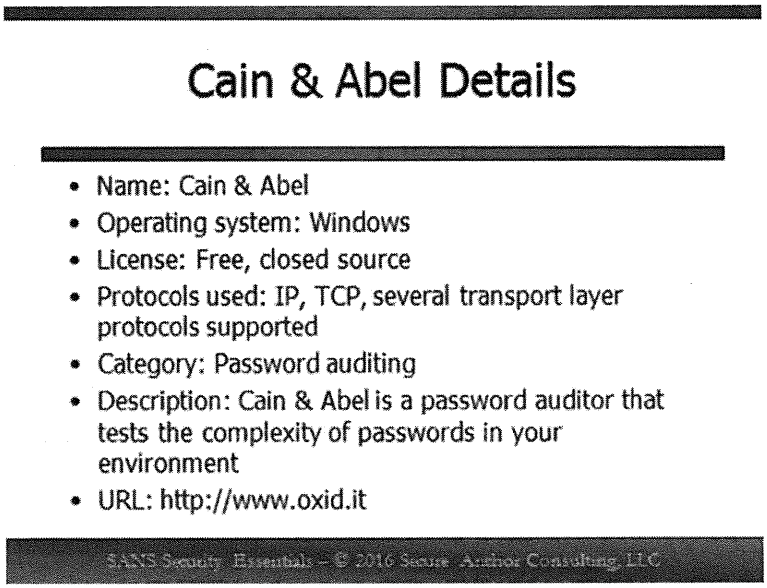
Important: Make sure that anti-virus software is turned off or temporarily disabled or it will stop the installation of password crackers.

Cain & Abel

Most organizations don't really know how easily or quickly an attacker can break your passwords. One of the easiest things you can do to test your password policy and guidelines is to audit the password strength with a tool like Cain & Abel. This is critical because in most organizations, passwords are the first and only line of defense. If an attacker can acquire a password, he or she can access sensitive information and cause damage to your organization.

Cain & Abel is a Windows-based, GUI password-auditing tool. It allows a network administrator to test the strength of the password policy through many means. You can sniff passwords from the network, crack the SAM on an emergency repair disk (ERD), crack the local SAM, or crack the SAM of a remote device (as long as you have the appropriate permissions).

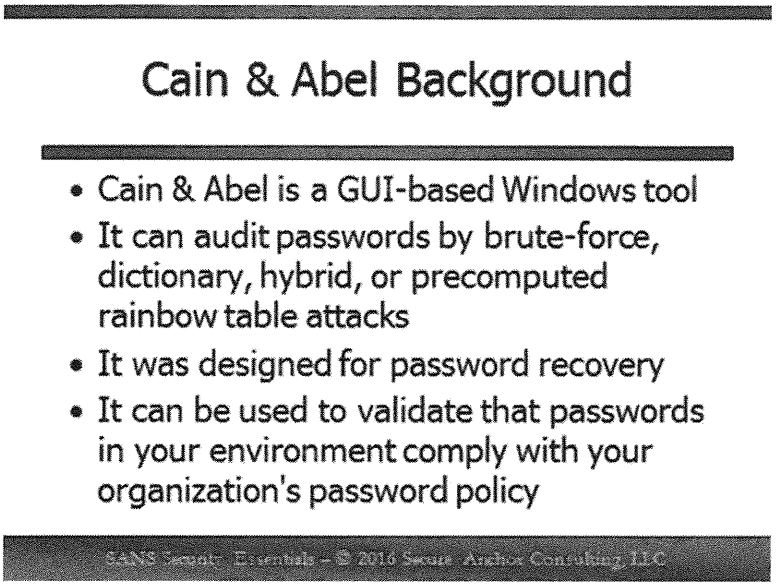
Warning: Make sure you get the appropriate permissions to do this work prior to performing any audits.



Cain & Abel Details

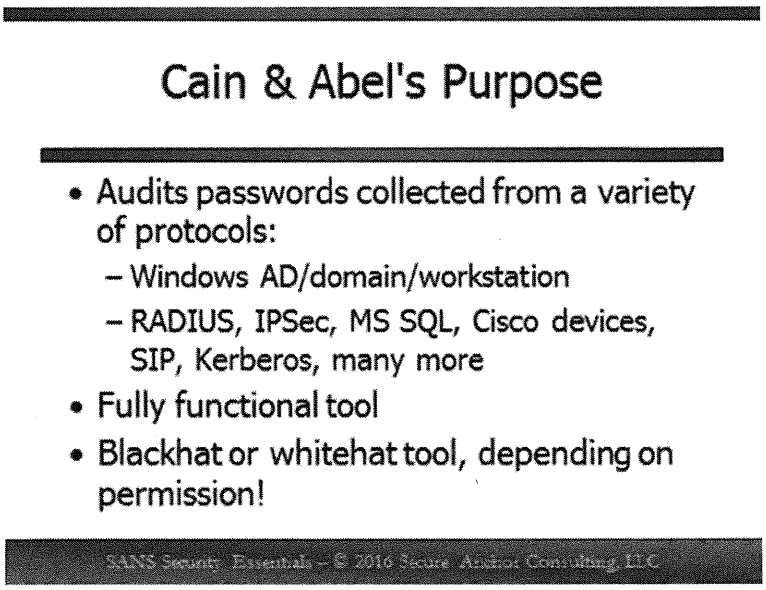
The following topics and action items are covered in this chapter:

- Learning about Cain & Abel and how it can be used to assess passwords for a Windows domain
- Identifying the common options in Cain & Abel
- Working through examples of running Cain & Abel
- Practice running Cain & Abel against local machines
- Practice cracking a sample SAM file from a Windows Active Directory server



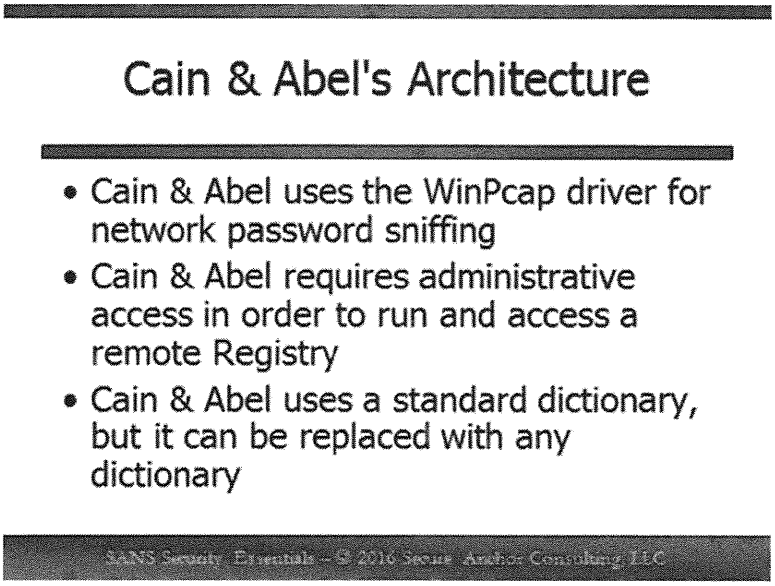
Cain & Abel Background

This section intentionally left blank.



Cain & Abel's Purpose

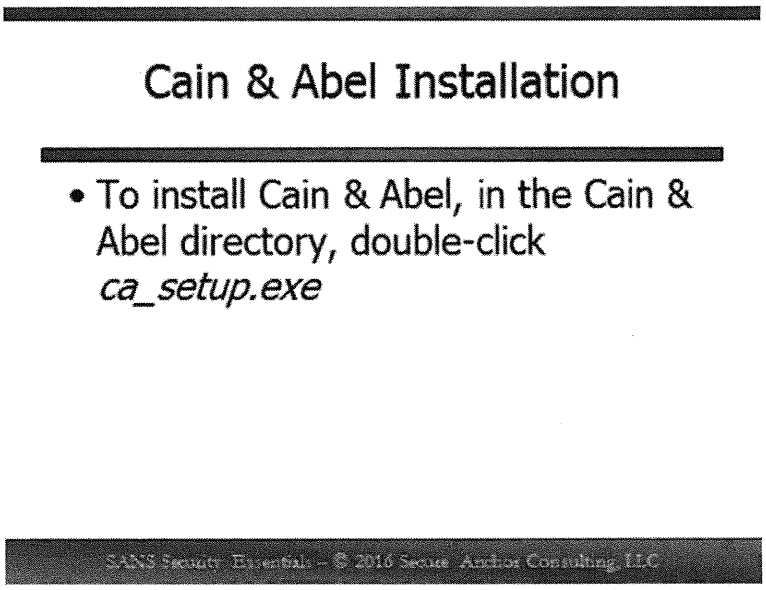
This section intentionally left blank.



Cain & Abel's Architecture

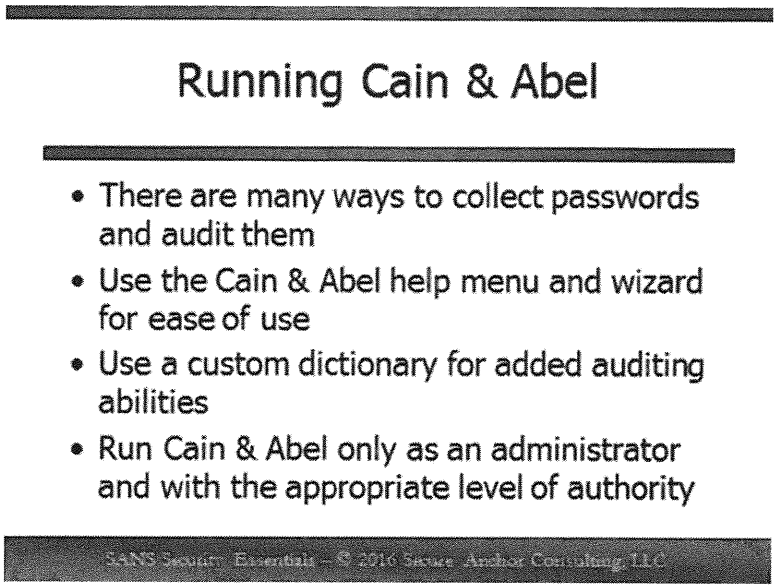
You might be asking, "How can I possibly get the SAM file off a running server?" If an administrator is following standard procedures, then he most likely has created an ERD and updates it on a regular basis. Most administrators want to make a backup of the registry, so they frequently use the -s option. When you use this option, you create a file called "SAM" in the c:\winnt\repair directory. It is a perfect duplicate of your SAM and can actually be removed. Of course, if an attacker gets access of your ERD, the game is over.

You use Cain & Abel to break Windows-based passwords. Cain & Abel can also be used to crack Unix passwords; however, it is not optimized for this purpose. A tool such as John the Ripper is more appropriate for cracking Unix passwords (see www.openwall.com/john for more information).



Cain & Abel Installation

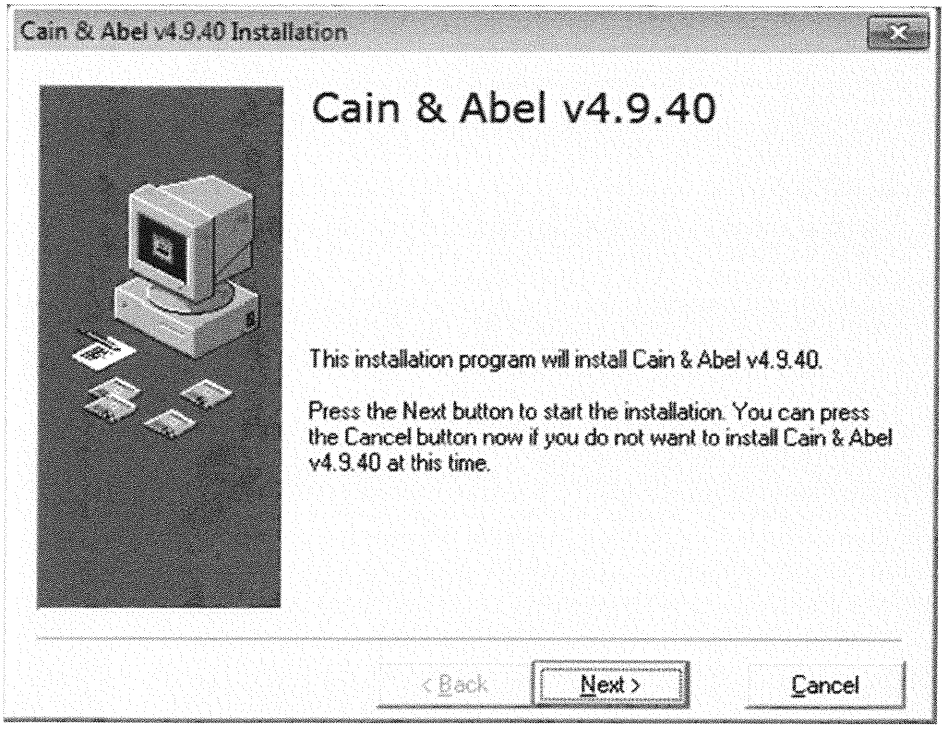
This section intentionally left blank.



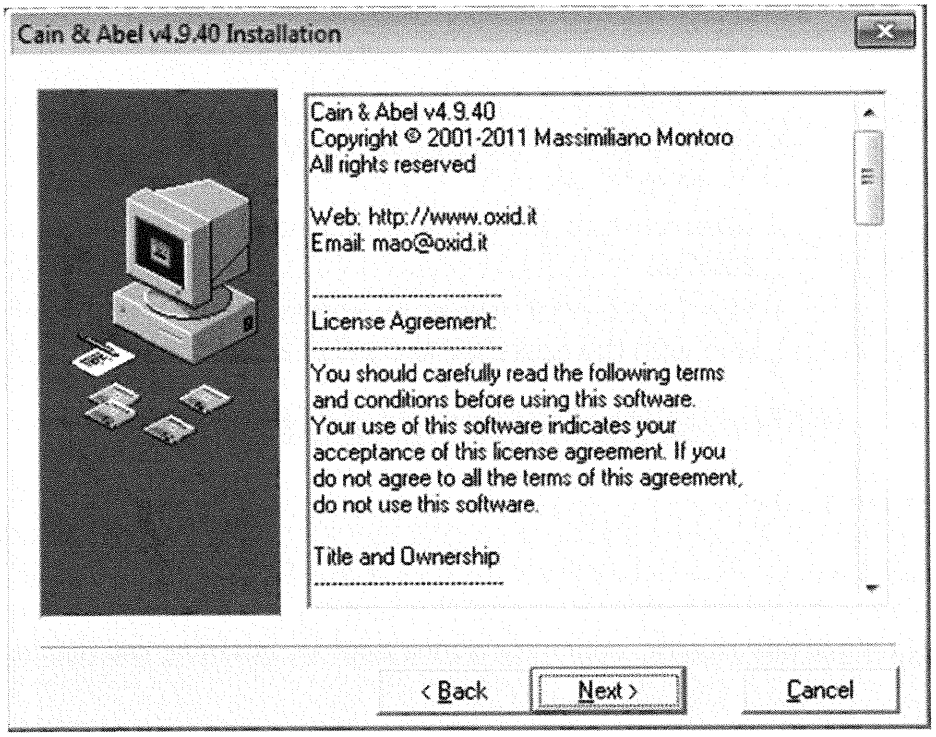
Running Cain & Abel

This section shows you how to install and run Cain & Abel. When installing Cain & Abel, you must also install both components: Cain and Abel, even though we will be using only Cain in this exercise. Following are the steps for installing and running Cain & Abel:

1. From the Cain & Abel directory on the CD, double-click *ca_setup.exe* to begin the installation. After the Cain & Abel installation window displays, click *Next*.

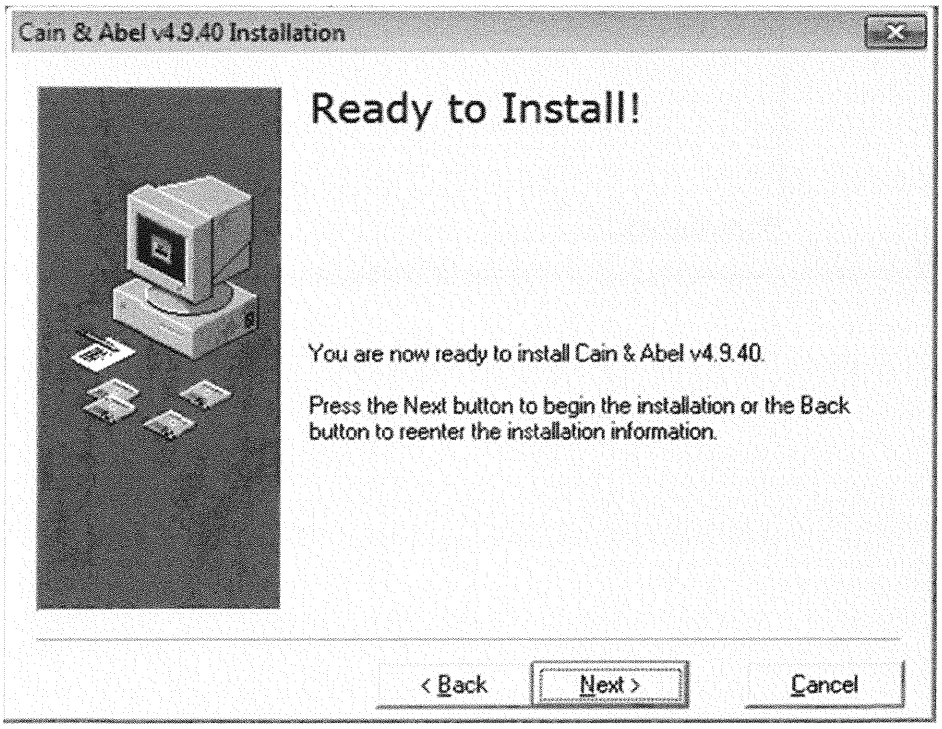


2. After the license agreement and Cain & Abel author contact information display, carefully read the license agreement to ensure you agree with the terms, and then click *Next*.

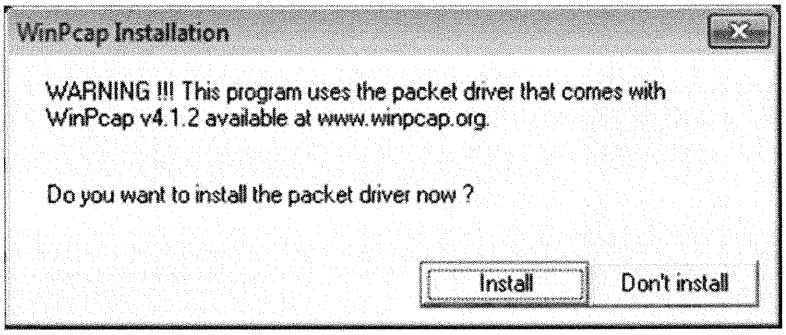


3. After the Select Destination Directory window displays, accept the default location and click *Next*.

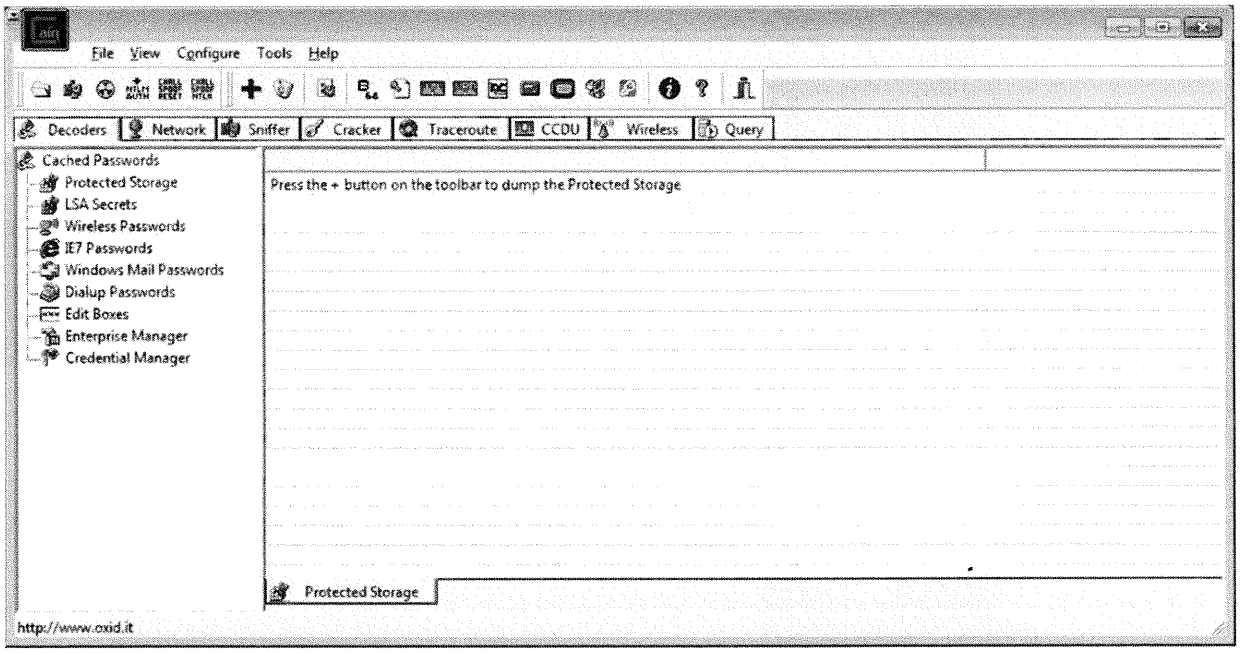
4. After the Ready to Install! window displays, click *Next*.



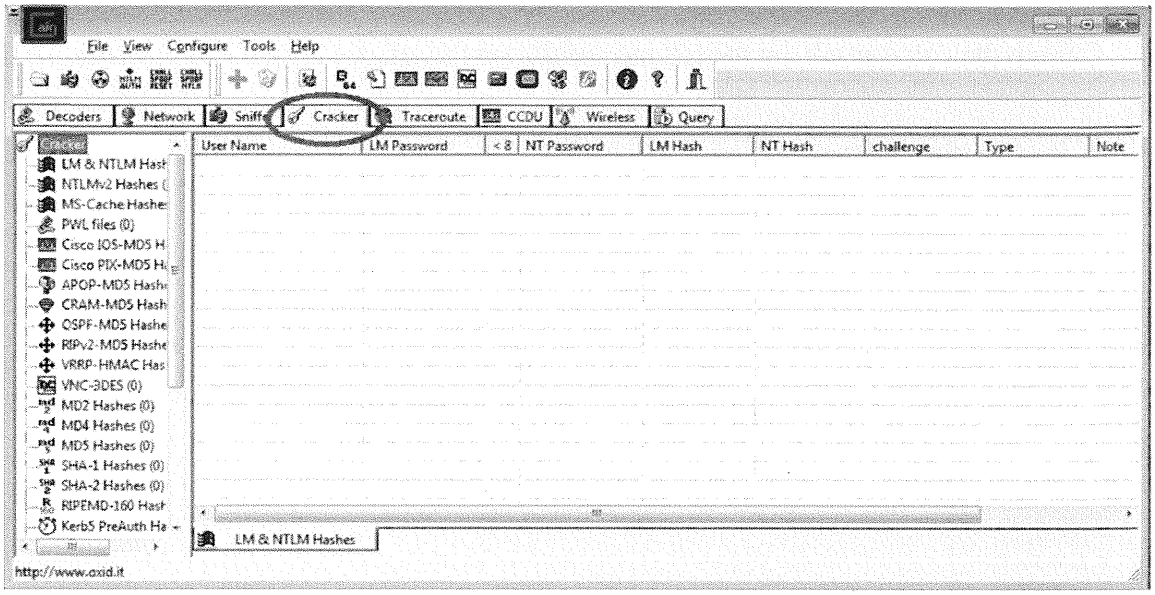
5. After completing the installation of Cain & Abel, the installation wizard prompts you to install the WinPcap drivers that are needed for packet capture. Click *Install*. Follow the prompts to install WinPcap on the system.



6. Start Cain by clicking *Start -> All Programs->Cain->Cain*.



7. Cain uses a tabbed interface to separate several of the different program functions. Click the *Cracker* tab to access the password-cracking functions.



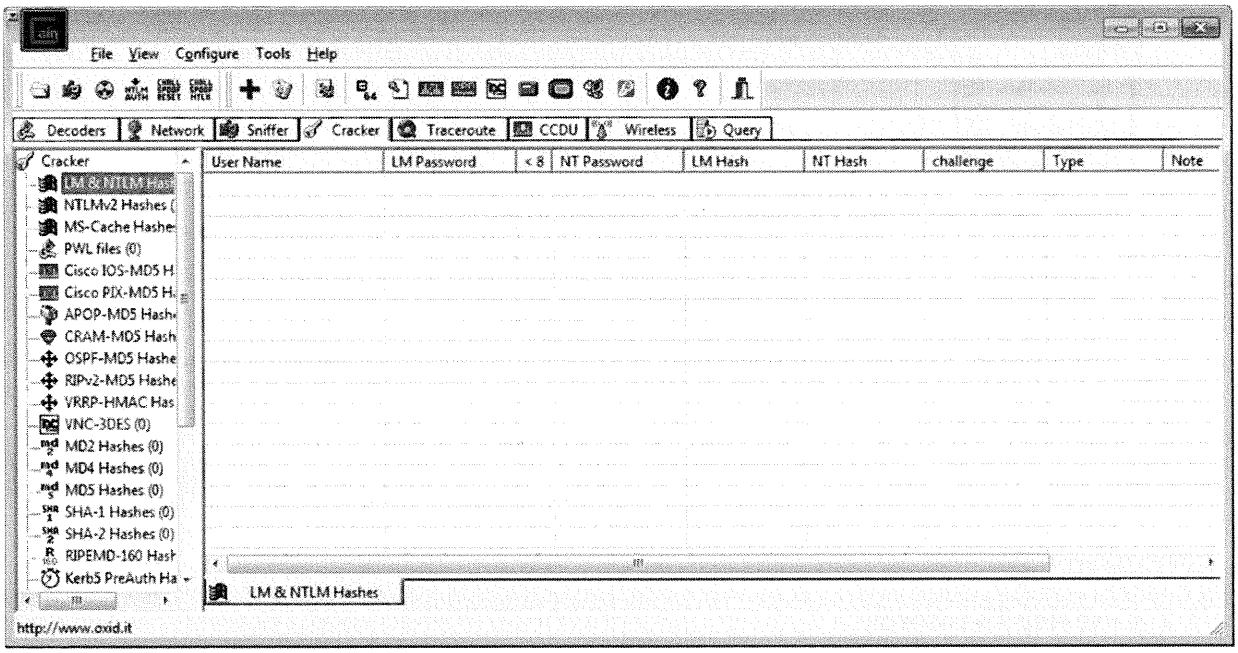
Cracking Passwords

Cain identified all the different passwords algorithms and protocols it supports for auditing in a tree hierarchy on the left side of the screen. For the purposes of this exercise, we focus on auditing LM & NTLM Hashes, used by Microsoft Windows networks.

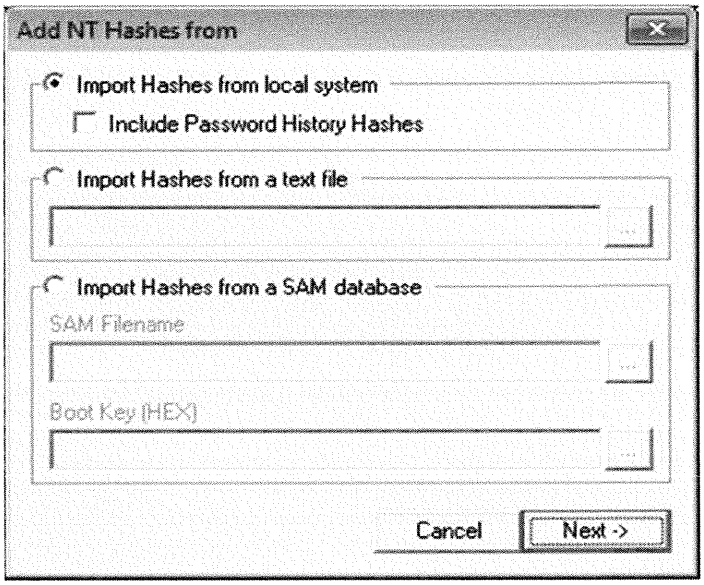
Auditing Windows Passwords

To start the Windows password hash collection wizard, perform the following steps:

1. From the Cracker tab, on the left side of the screen, click *LM & NTLM Hashes*.



2. Click the blue plus sign on the toolbar to start the password hash collection wizard.



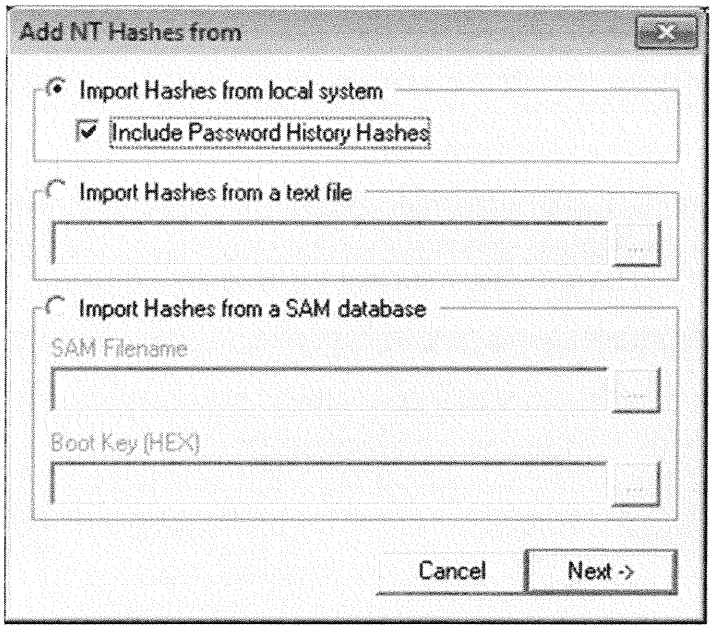
When you start the wizard, three options indicate where you can collect the encrypted passwords:

- The local system
- A text file
- The SAM database

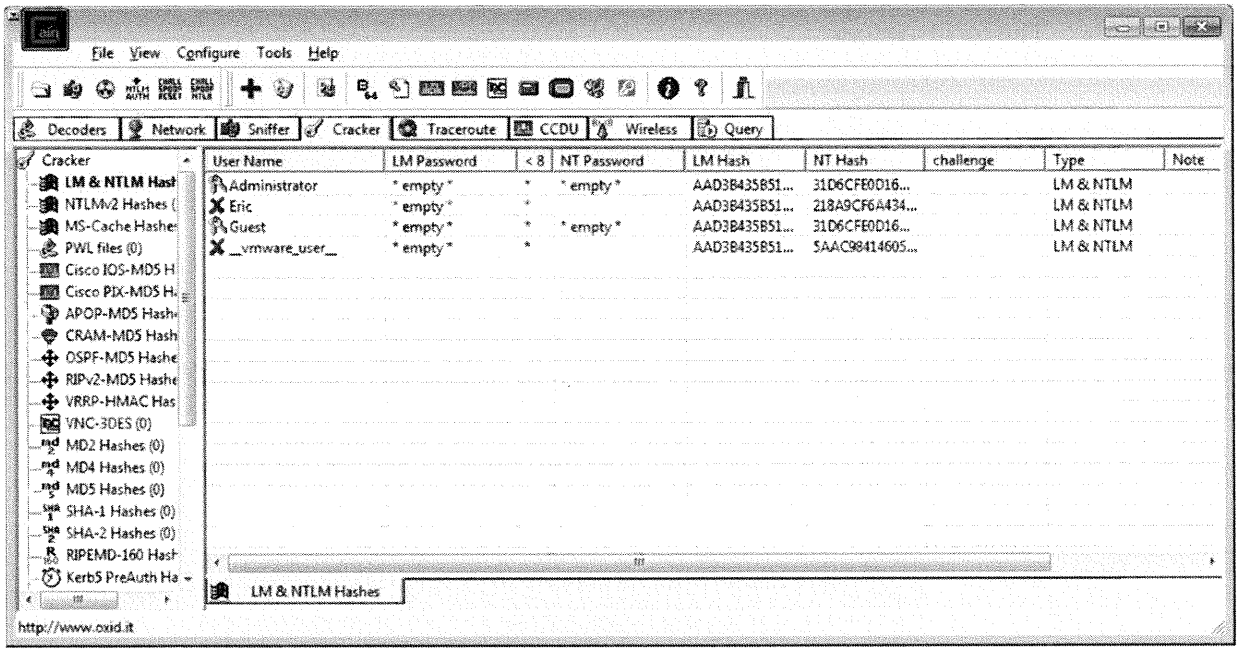
Cracking Passwords on the Local System

To crack passwords on the local system where Cain is installed, perform the following steps:

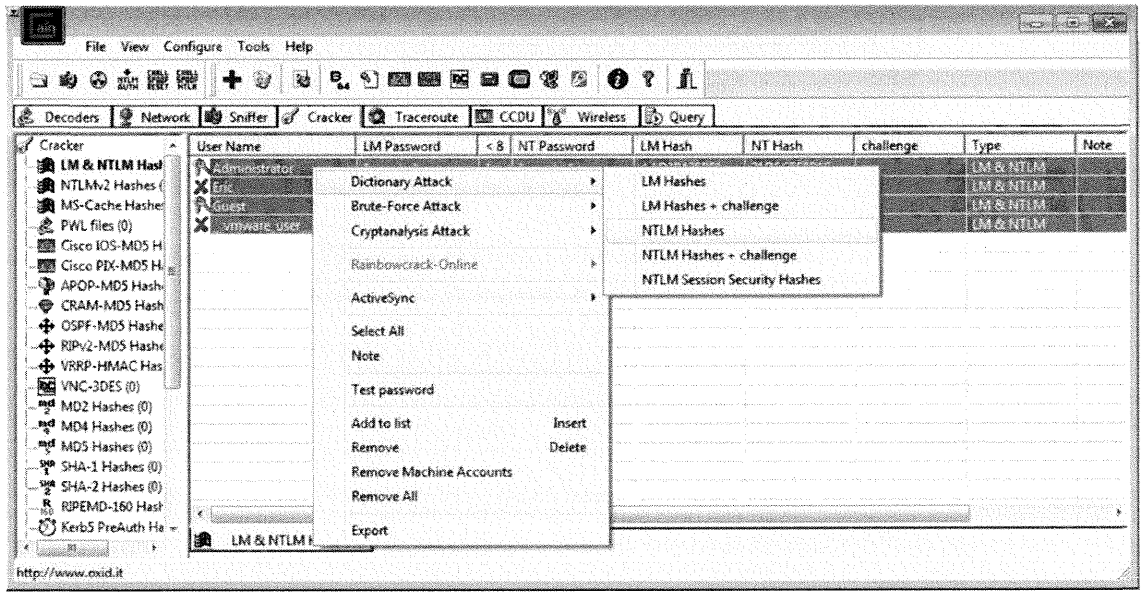
1. Click *Import Hashes from local system*. Optionally, click *Include Password History Hashes* to also audit previous passwords stored by the operating system. Click *Next*.



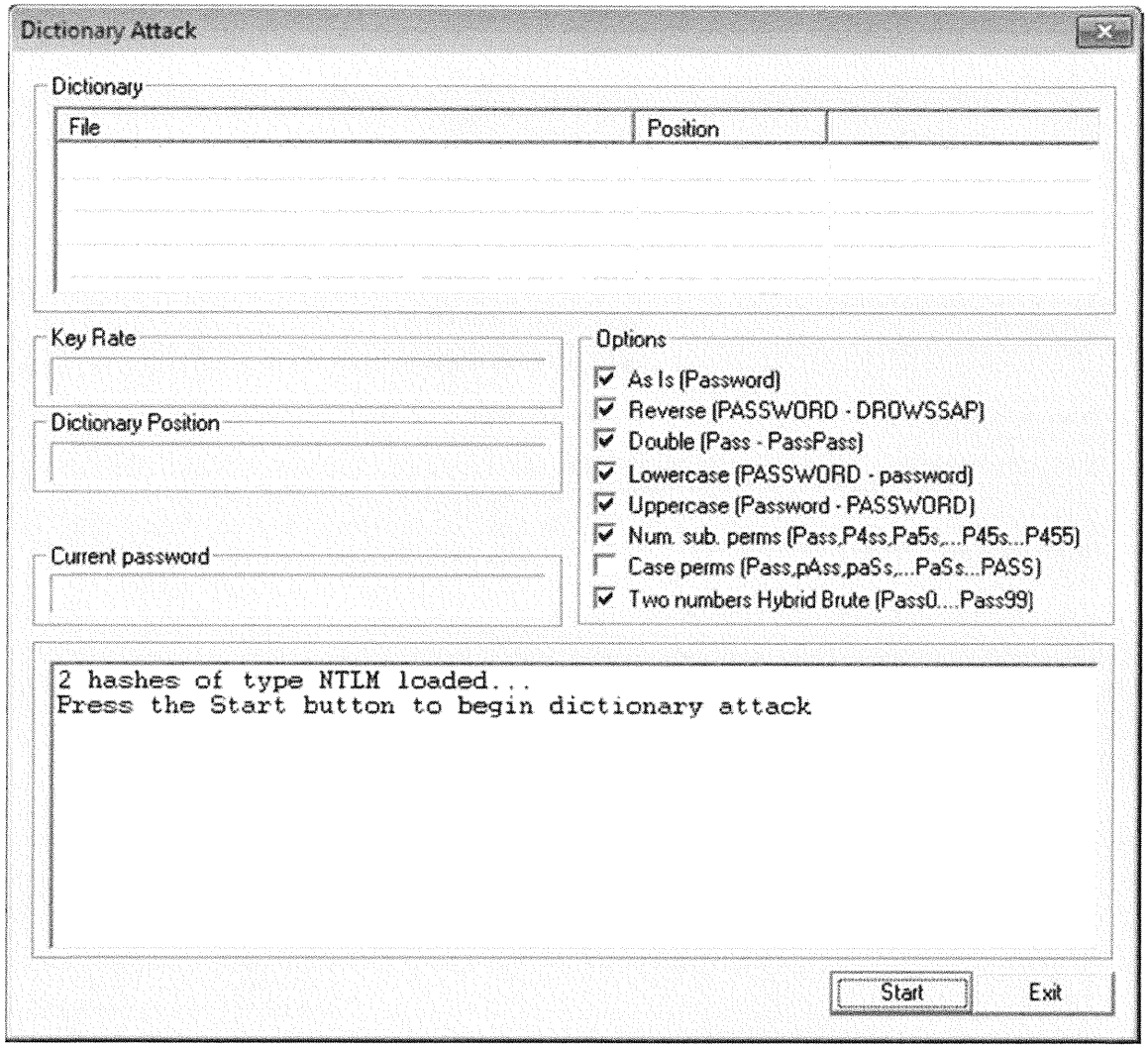
2. Cain dumps a list of local user accounts and password hashes into the Cracker tab view, as shown in the following screen.



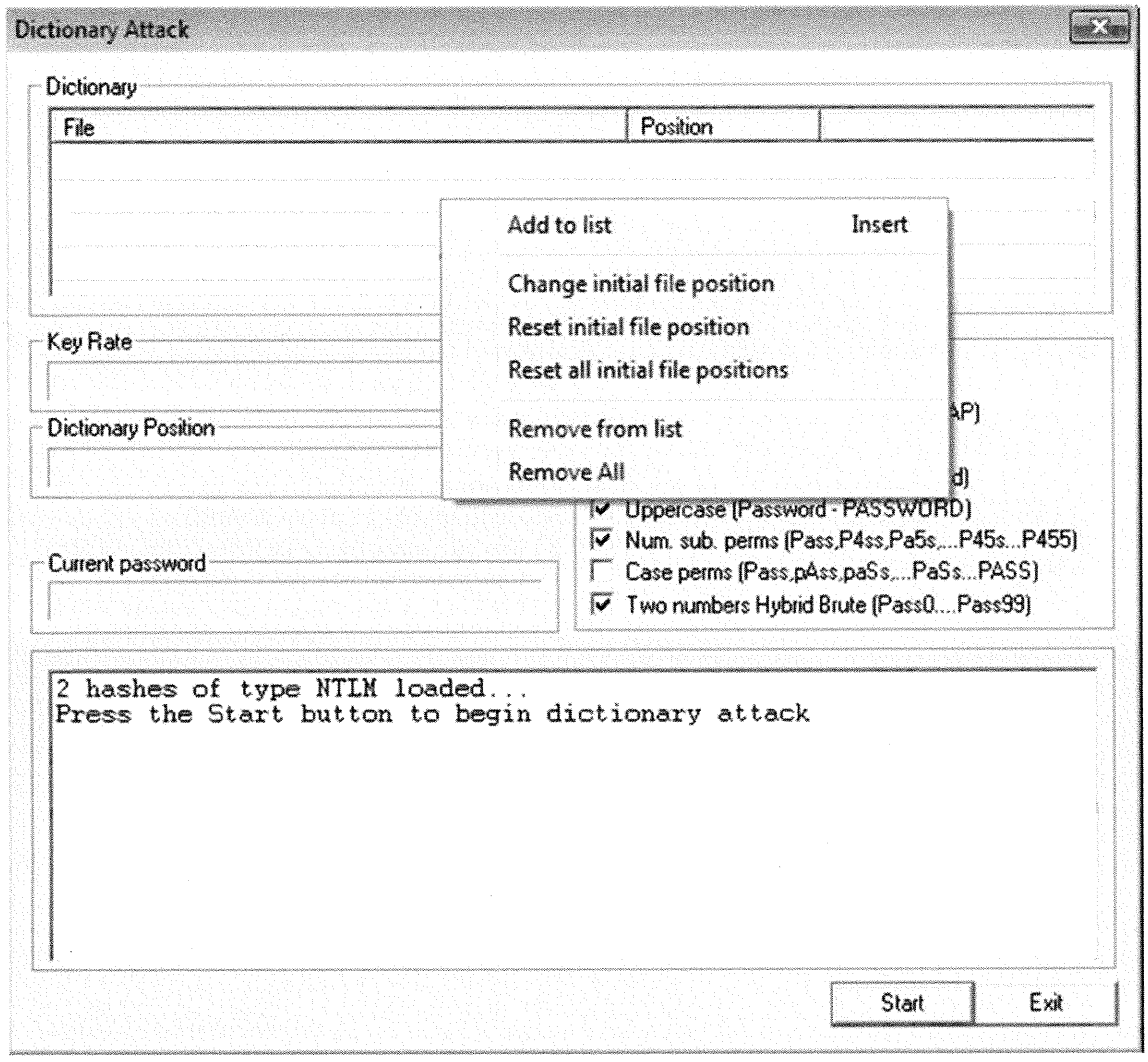
3. To initiate the password audit, right-click on the account you want to audit to open a list of options, as shown in the following screen. Click *Select All* to audit all the local system accounts. Cain highlights all the accounts.



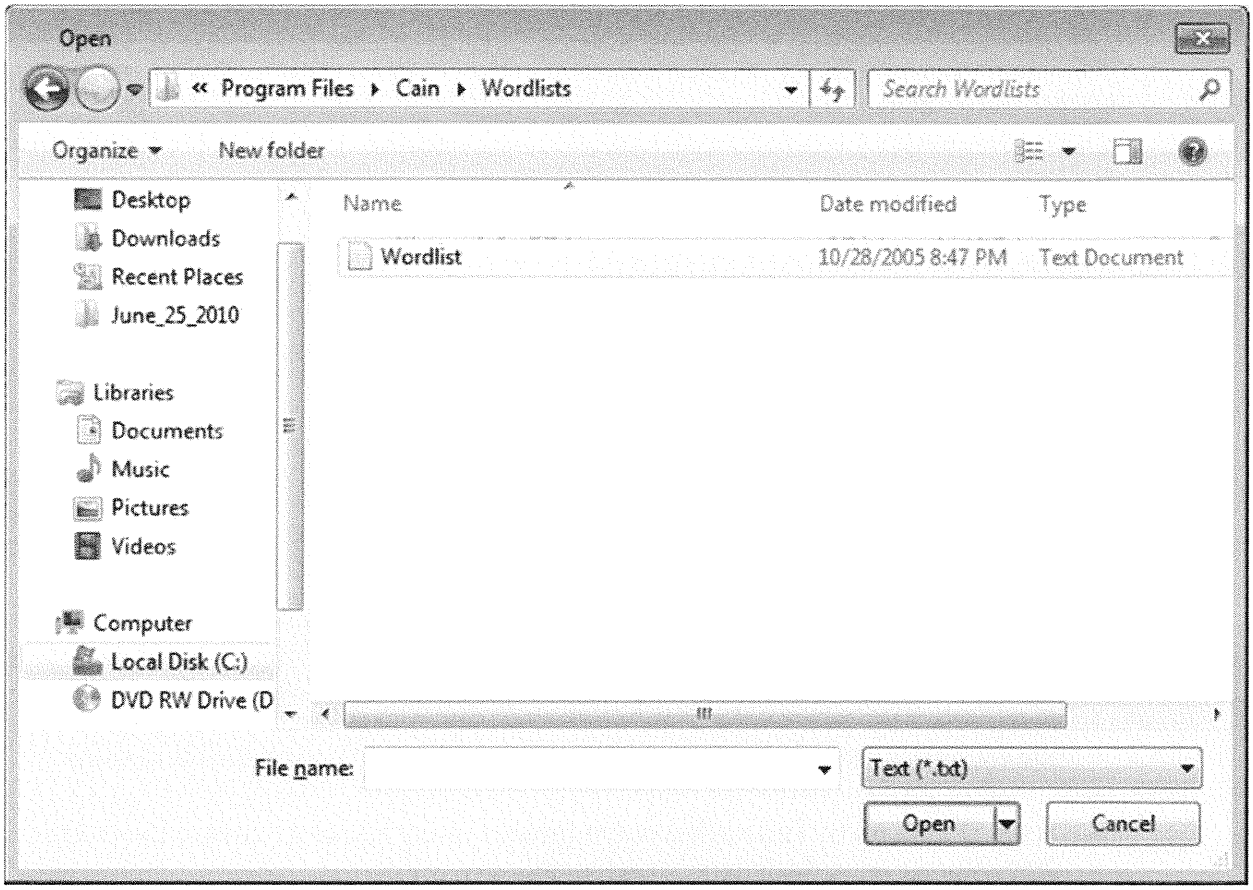
4. Next, right-click on any account, and then click the *Dictionary Attack* (NTM) option to open the Dictionary Attack window. Because Windows 8 has LM turned off by default, click *NTLM*.



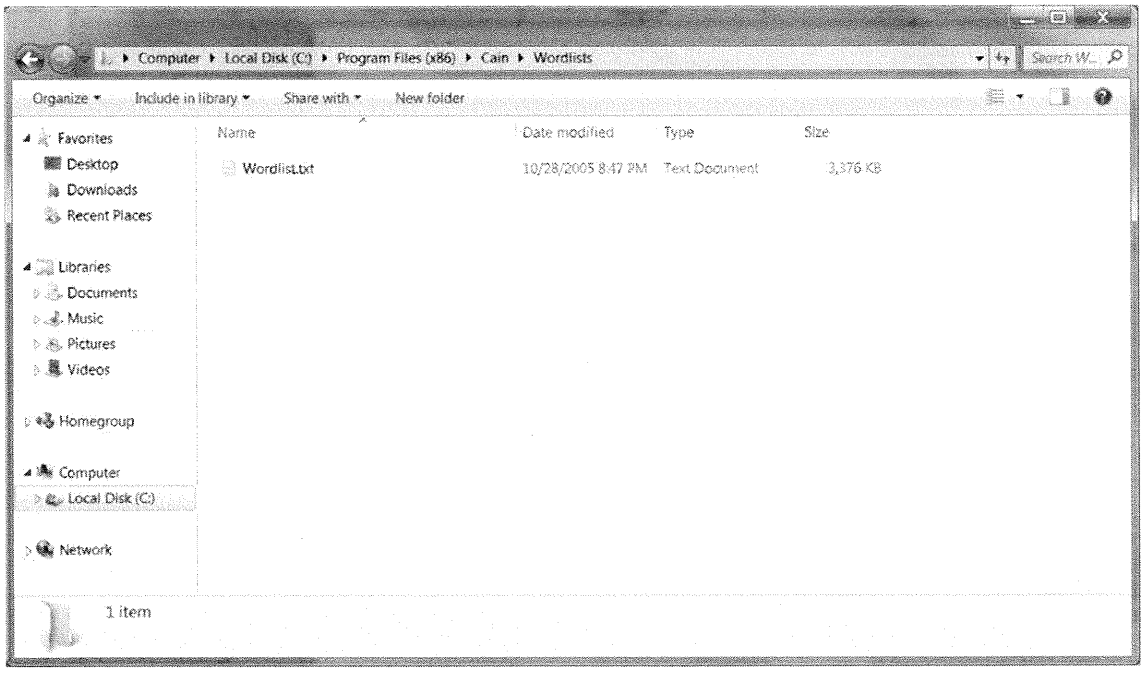
5. Cain supplies a minimal dictionary file that can be used to audit for simple passwords. Right-click under the Dictionary section, and a pop-up menu displays. Click *Add to list*.



6. Navigate to the *C:\Program Files\Cain\Wordlists* directory, click the *wordlist.txt* file, and then click *Open*. Note that it is impossible to add multiple files; Cain uses all the words in each file for the audit.



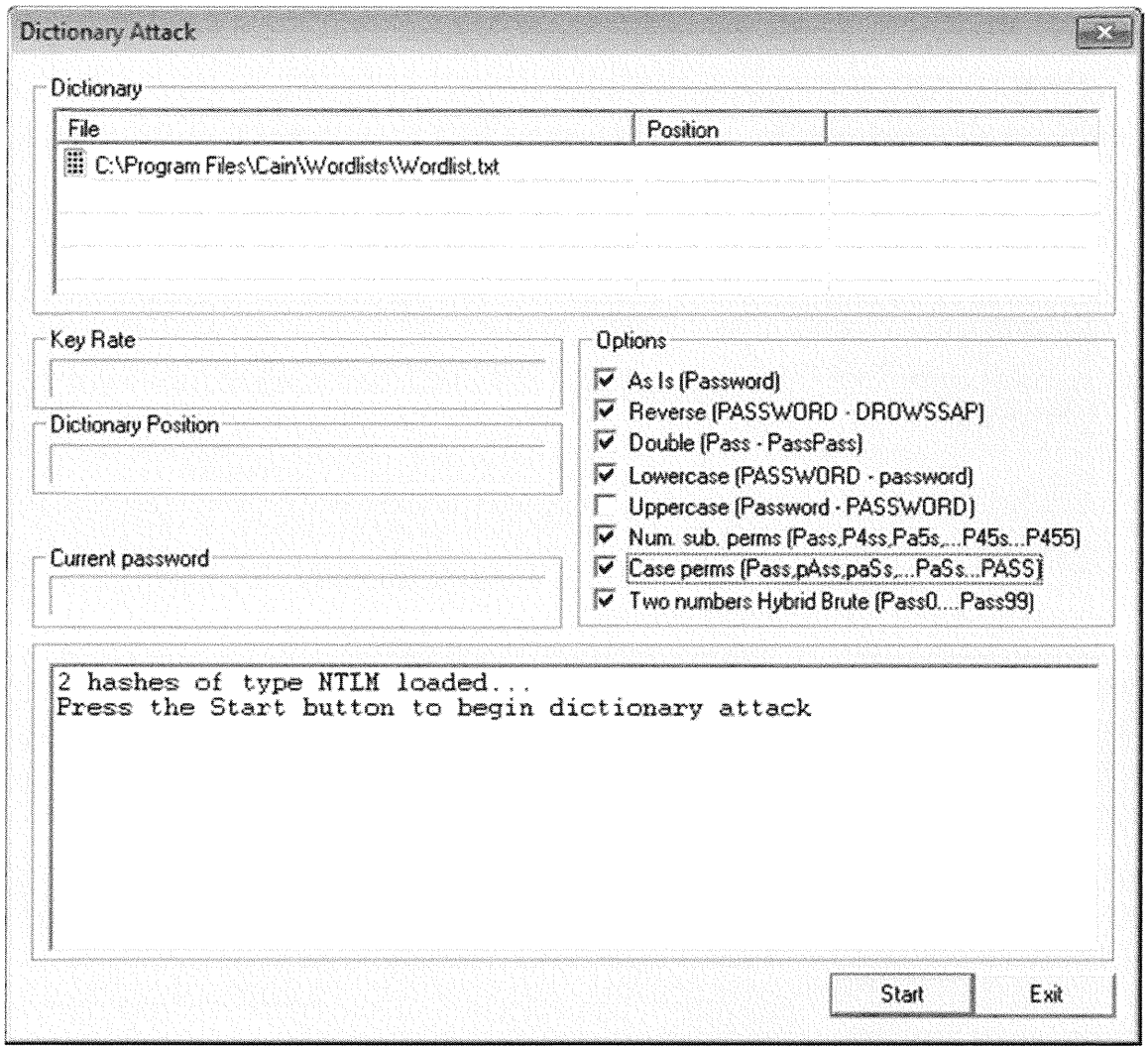
Note: On some Windows 8 systems, you might have to go to c:\Program Files (x86)\Cain\Wordlists.



7. After selecting the wordlist.txt file, Cain returns to the Dictionary Attack window. Next, select the password permutations you want to use for the audit by selecting the following settings in the Options section of the Dictionary Attack window:

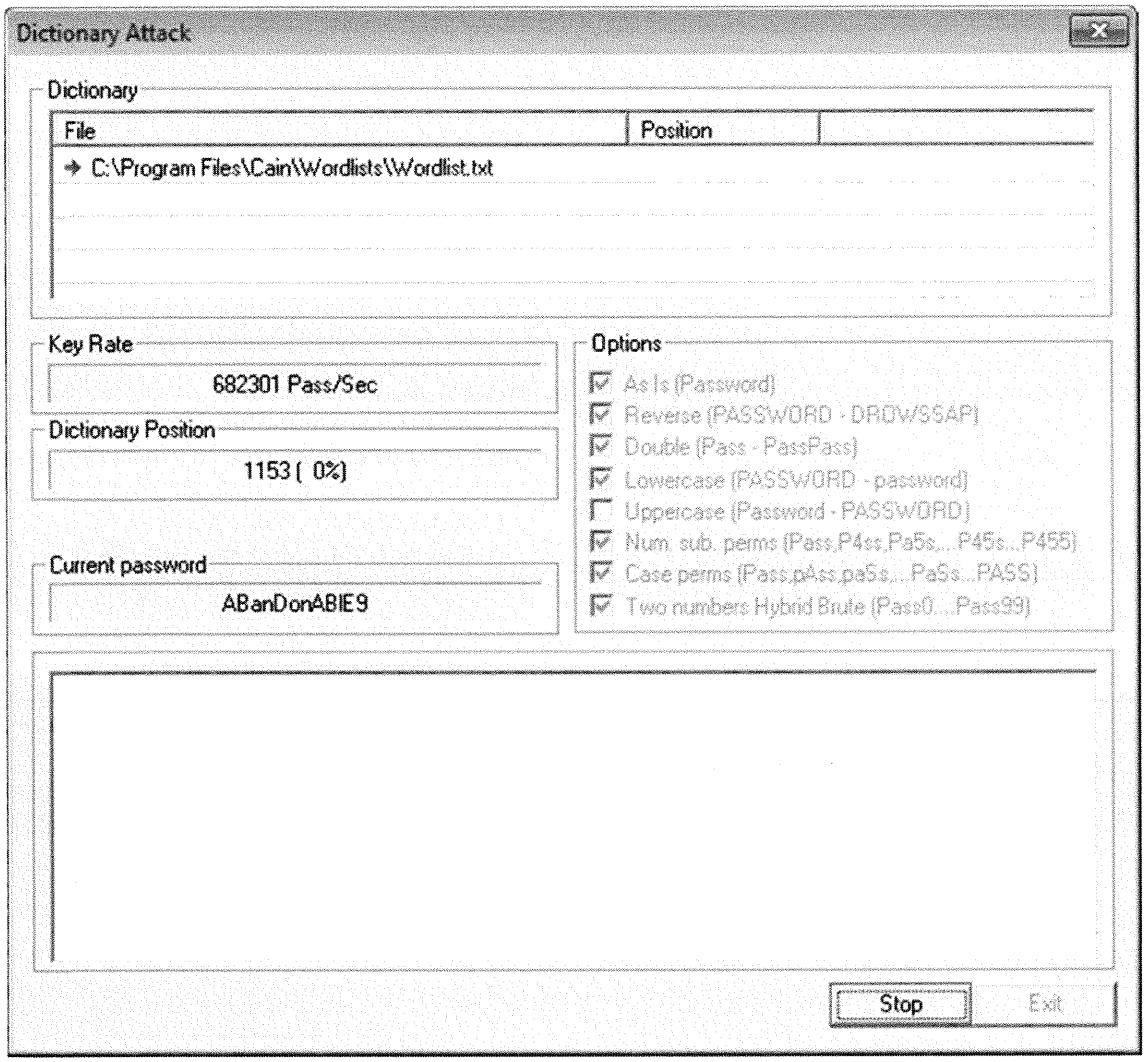
- **As Is (Password):** Test each word in the dictionary file as a potential password.
- **Reverse:** Reverse the letters in each dictionary word.
- **Lowercase:** Try the dictionary word in all lowercase letters.
- **Case perms:** Try the dictionary word with mixed case.
- **Two numbers Hybrid Brute:** Try each dictionary word after appending 00–99 to the end of the word.

Note: The Uppercase option is mutually exclusive with the mixed-case option; it is not necessary to select both options.



8. Start the audit by clicking the *Start* button. Cain uses each word in the dictionary according to the selected options until there are no remaining words, or all passwords have been revealed. Revealed passwords are displayed in the lower text box, as shown in the following screen.

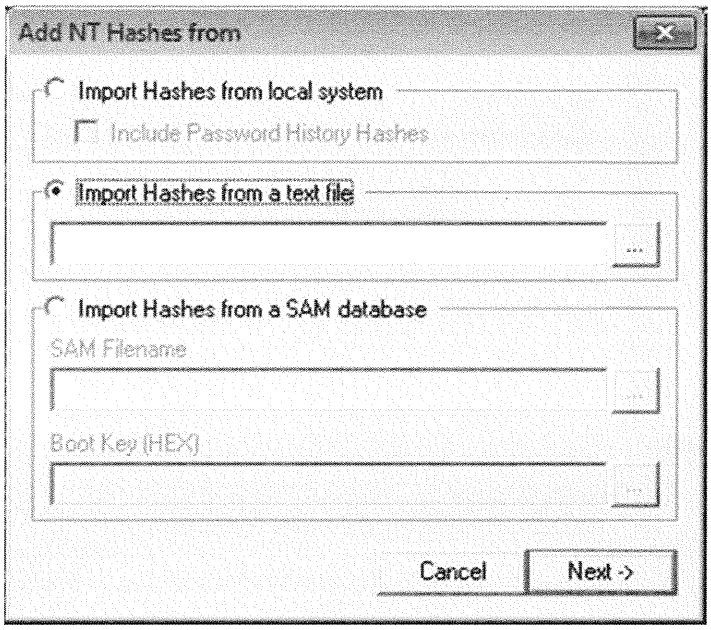
After you are finished with the audit, click *Exit*. The Exit button will not be highlighted until the program has finished running.



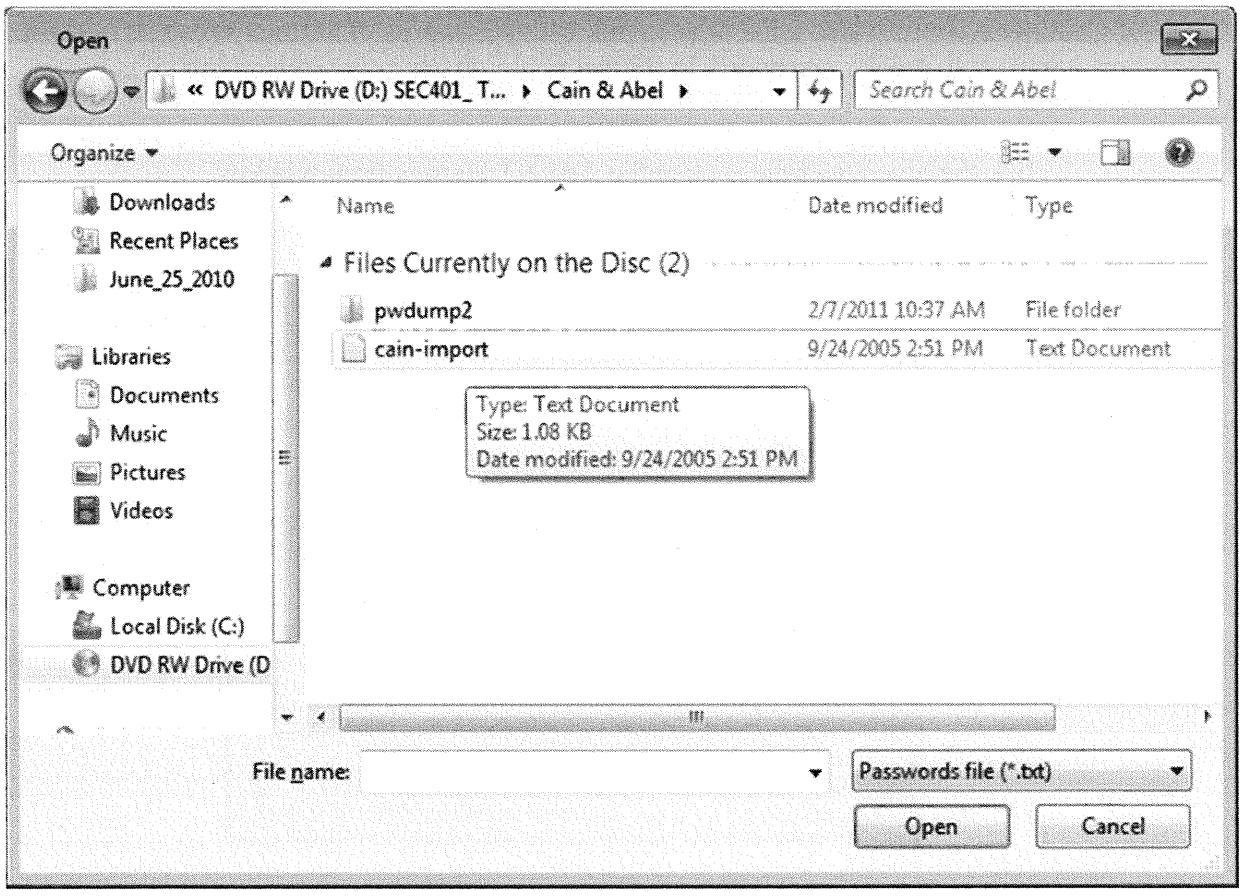
Cracking Passwords from a File

Depending on what accounts are on your local system and the strength of the passwords, you might have received limited results from the previous scan. To help show the power of Cain, we have provided a list of accounts with passwords of various strengths to show how easy it is to crack a password. In this section, you are going to load this file.

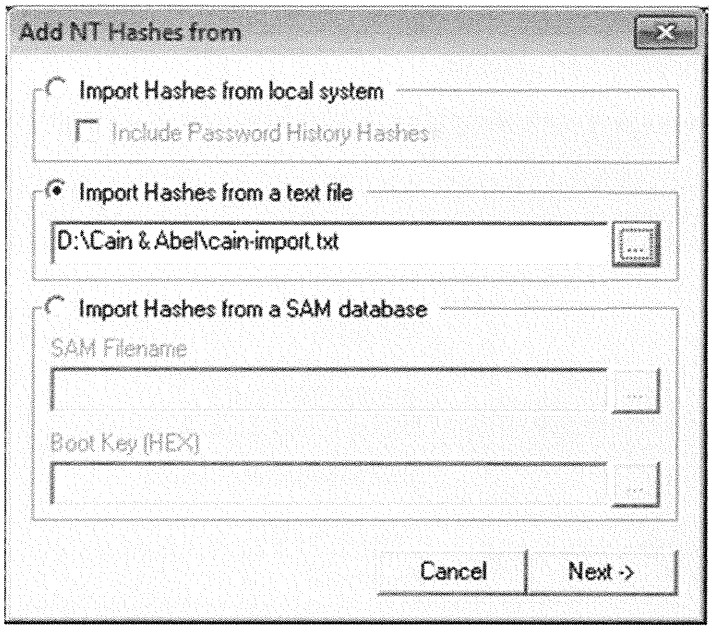
1. From Cain's Cracker tab, click *LM & NTLM Hashes*, and then on the toolbar, click the blue plus sign to open the Add NT Hashes from window.



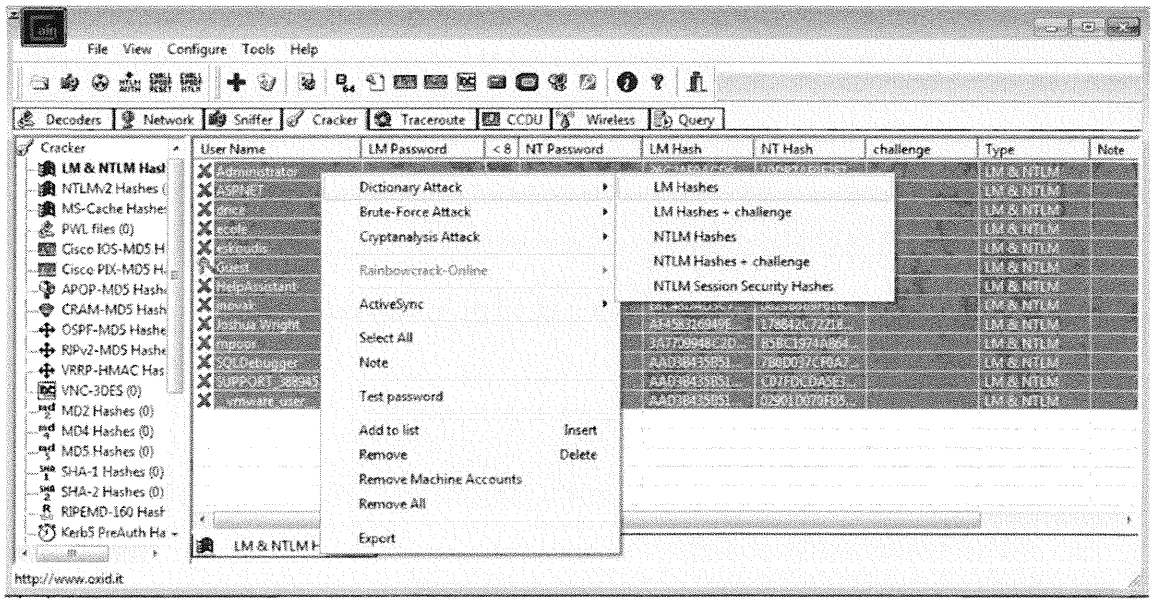
2. Click the *Import Hashes from a text file* radio button, and then click the square next to the input box to open a file selection dialog box. Navigate to the *Cain & Abel* directory on the CD, click the *Cain-import.txt* file, and then click *Open*.



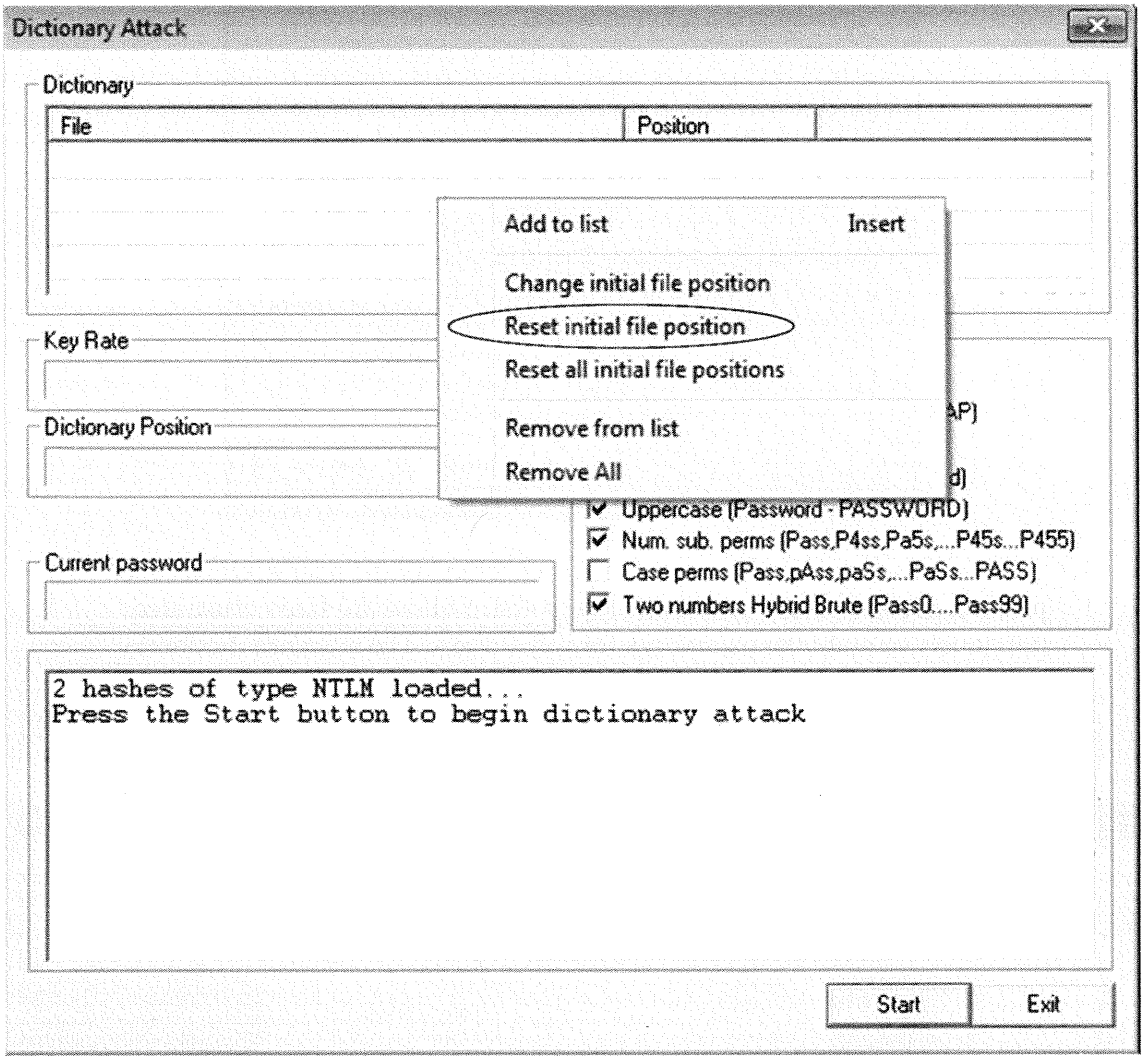
3. In the Add NT Hashes from dialog box, click *Next*.



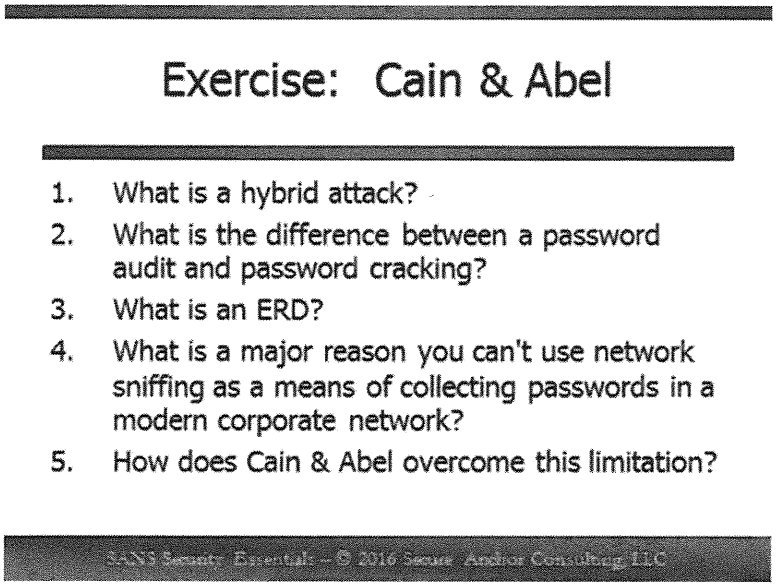
4. Cain returns to the Cracker view after importing all the user accounts and password hashes that were in the import file. Follow the steps illustrated previously to audit the passwords for these users. The standard dictionary supplied with Cain reveals only some or partial passwords for the supplied account information. Because these passwords are from an XP machine, you can use the LM cracking so it will go quicker.



After the Dictionary Attack screen appears, right-click in the Dictionary section and click *Reset initial file position*. Otherwise if the dictionary is at the end of the file, it will not be able to crack any new passwords. The dictionary should be reset to the initial position after each run of the password cracker.

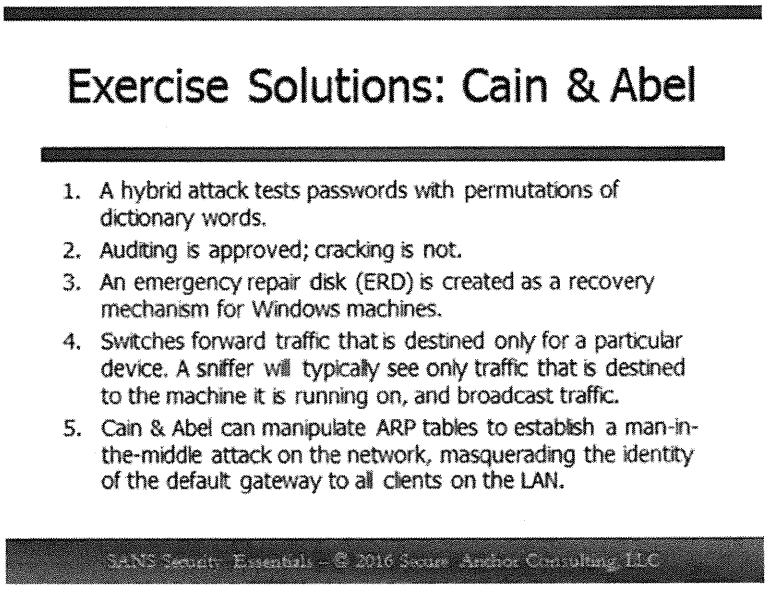


You can now start the new password-cracking attempt by clicking *Start*.



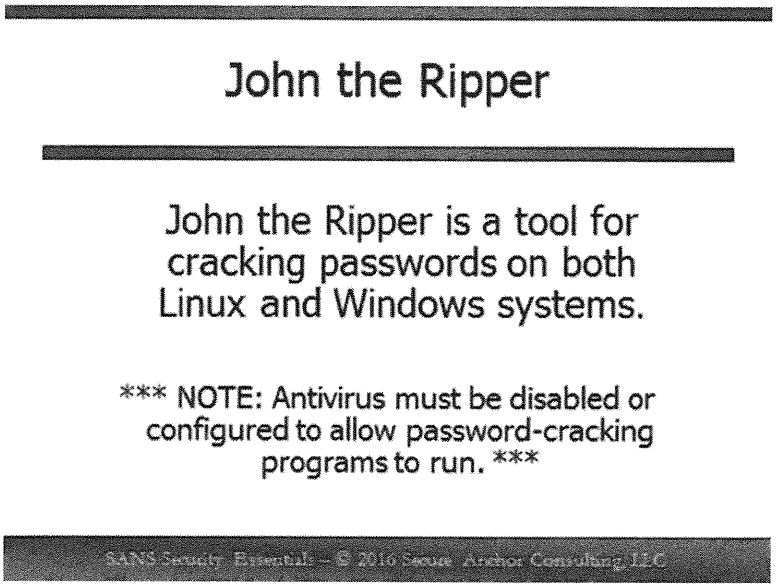
Exercise: Cain & Abel

This section intentionally left blank.



Exercise Solutions: Cain & Abel

This section intentionally left blank.



Important: Make sure that anti-virus software is turned off or temporarily disabled or it will stop the installation of password crackers.

John the Ripper

How many times have you received a complaint from a user named "Bob" saying he can't remember his password and has been locked out?

What Bob doesn't realize is that strong passwords are a foundation for securing critical data and resources within an organization. Policies dictate that strong passwords must be a certain length, must contain certain types of characters (alphanumeric, multi-case, special characters, and so on), and must change every month or so. These policies are designed to guide users such as Bob on how to maintain and manage their passwords. Beyond these policies, organizations often ensure users properly maintain password strength by auditing them. As organizations struggle with implementing single-sign-on technologies, users juggle multiple passwords for multiple applications. These ingredients lead to disaster from a password-management standpoint. Good security practice recommends that administrators periodically audit accounts as well as review and audit password strength using a password-cracker utility such as John the Ripper.

As a password cracker, John the Ripper is available for many different operating systems, including Unix, DOS, and Windows 95/98/NT/2000/2003/XP/Windows7/2008. It is a command-line interface that places results in an output file. For GUI-trained users, this might seem cumbersome, but in fact, John the Ripper is a strong, capable tool that all administrators should know how to use.

This exercise introduces you to John the Ripper and shows you how this password-cracking tool can assist with auditing the strength of passwords within network systems. You learn how to crack both Unix and Windows-style passwords as well as gain an

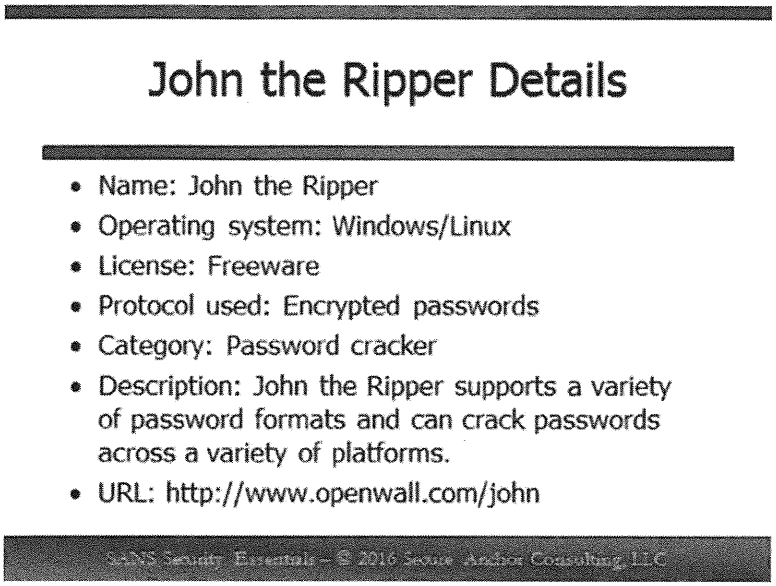
understanding of the tool's processes. John the Ripper can crack a password in four different modes: single, dictionary, hybrid, and brute force. This chapter explains the differences between these modes and demonstrates the single-mode crack.

Warning: You should use only a password-cracking utility with the written permission of the data owner. Get that permission from a senior member of the organization before performing any cracking. This exercise is not designed to train any individual to illegally gain access to system data and take advantage of network resources, but to illustrate how you properly audit password strength.

The current version of John the Ripper, version 1.6, is available from http://www.openwall.com/john.

The program is compressed using Pkzip, which is available in a GUI interface from http://www.winzip.com.

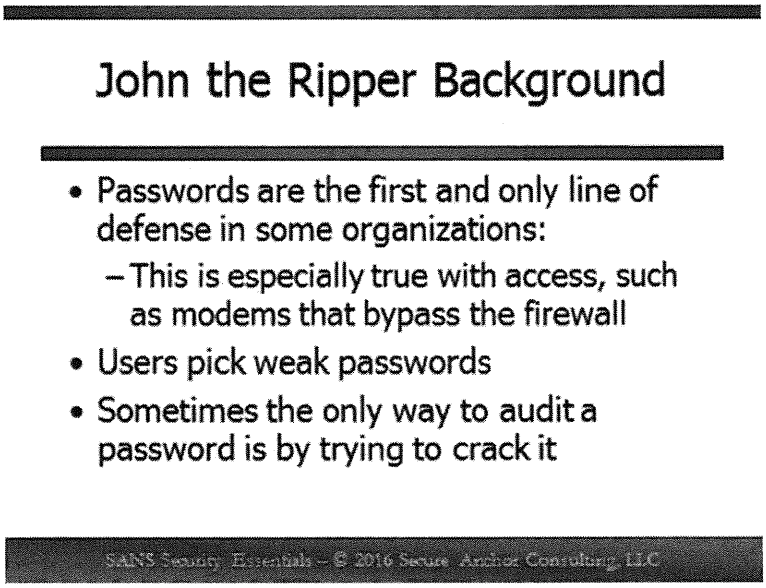
An uncompressed version of John the Ripper is available on the class DVD.



John the Ripper Details

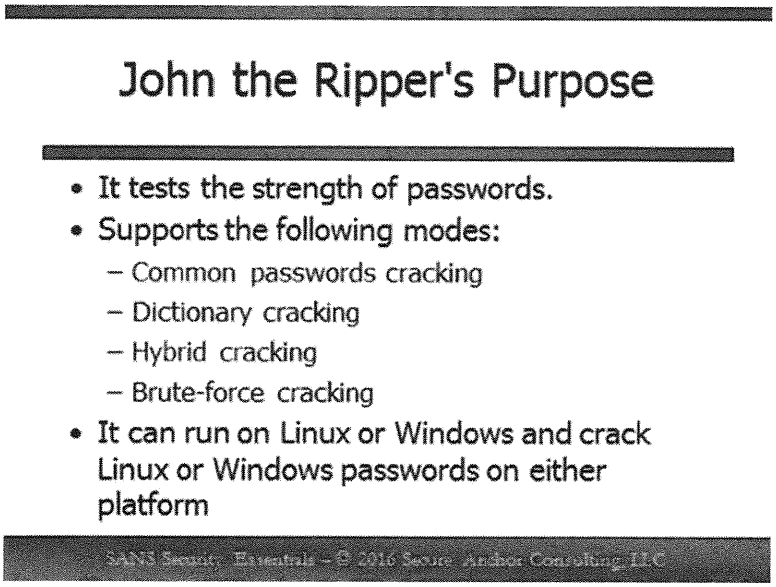
The following topics and action items are covered in this chapter:

- Installing John the Ripper on Windows 8
- Cracking and identifying passwords from a Unix password file on a Windows 8 system
- Cracking and displaying the passwords from the Windows 8 registry



John the Ripper Background

This section intentionally left blank.



John the Ripper's Purpose

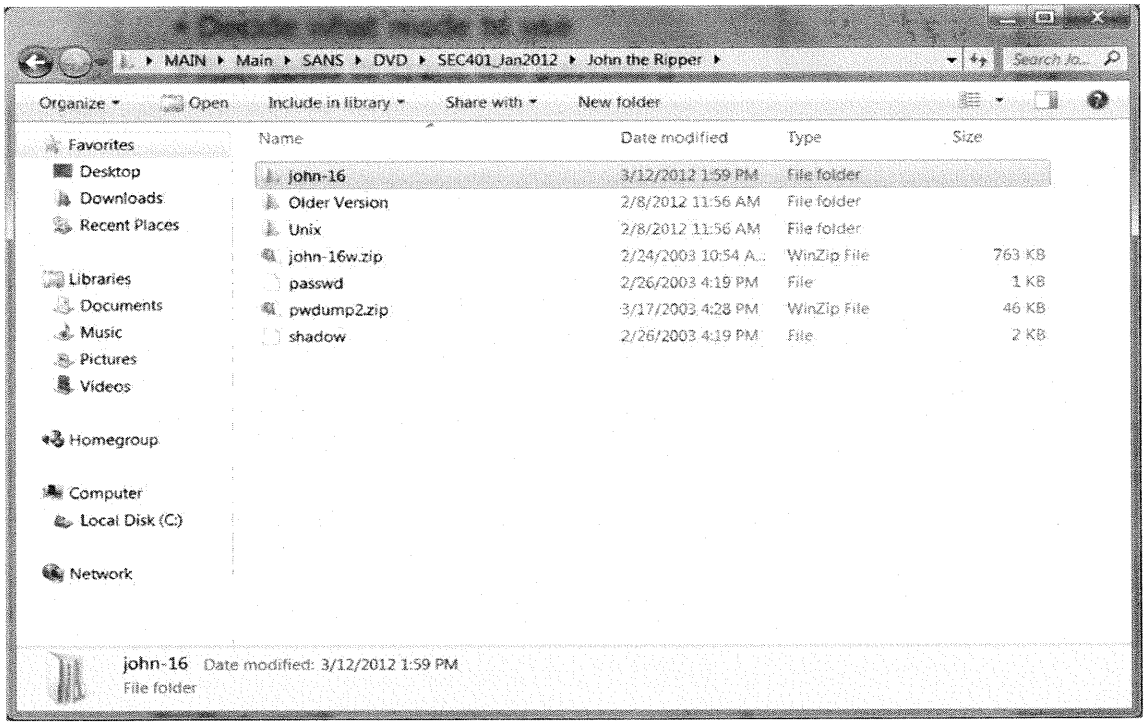
John the Ripper's purpose is to test the strength of passwords. John the Ripper supports four modes: common password cracking, dictionary cracking, hybrid cracking, and brute-force cracking.

John the Ripper Installation

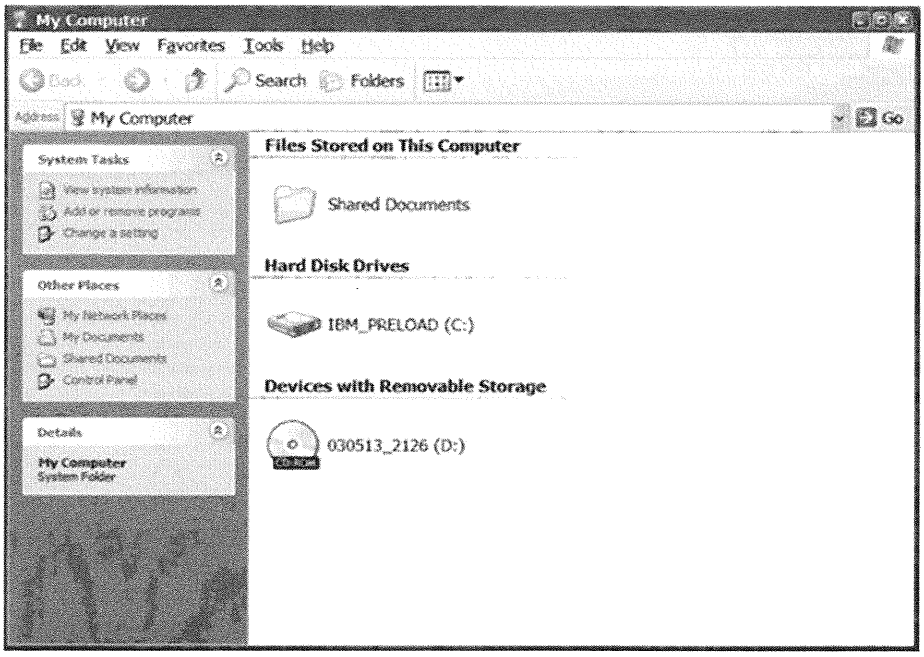
- Unzip the files and extract them to a directory
- Prepare the password files
- Decide what mode to use
- Run **john** to crack the password files

John the Ripper Installation

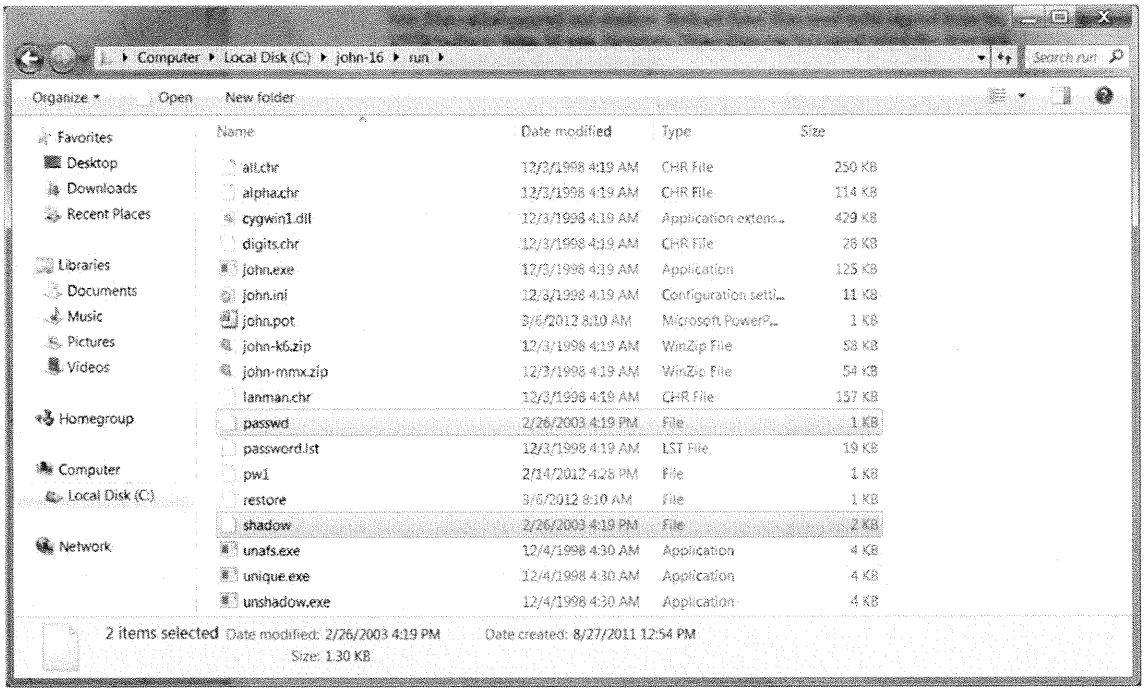
1. Prior to starting the install process, you would typically need to uncompress the john zip file. To make it easier for you, an uncompressed version of john is included on the class DVD. This folder is on the DVD under the John the Ripper directory called "john-16."



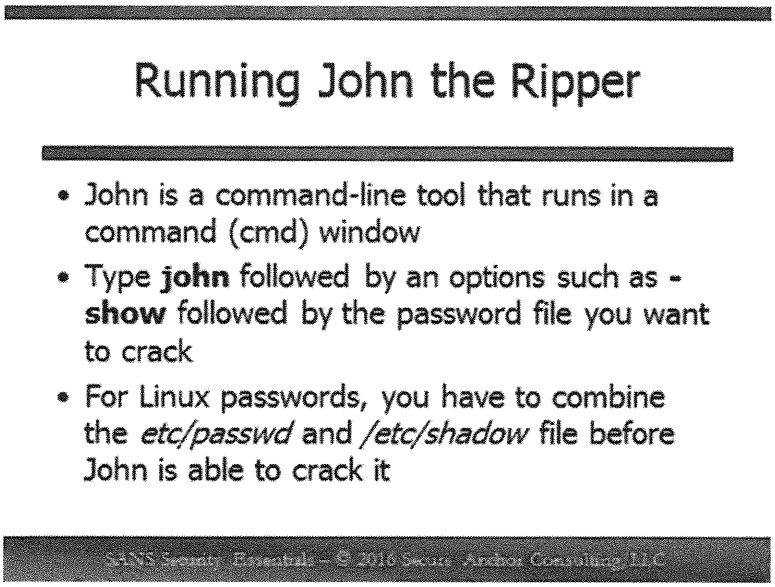
2. You need to copy the entire directory john-16 to the C: hard disk drive. The easiest way to do this is to open *My Computer* and then drag and drop this folder to the hard disk drive labeled *C:*.



3. You now have a directory called "c:\john-16" with a sub-directory called "run." Now you must copy over the Linux password files because you are going to crack Linux passwords on a Windows platform. From the DVD under the John the Ripper directory, copy the *passwd* and *shadow* files to the *c:\john-16\run* directory. These files can be copied using the drag-and-drop method from the previous step.



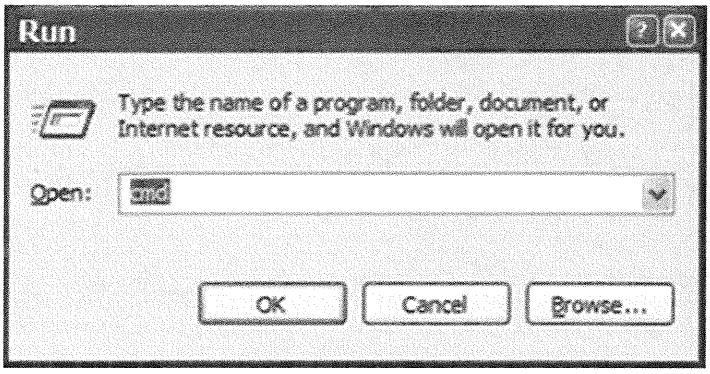
Make sure you verify that your c:\john-16\run directory contains both the passwd and shadow files, before you continue.



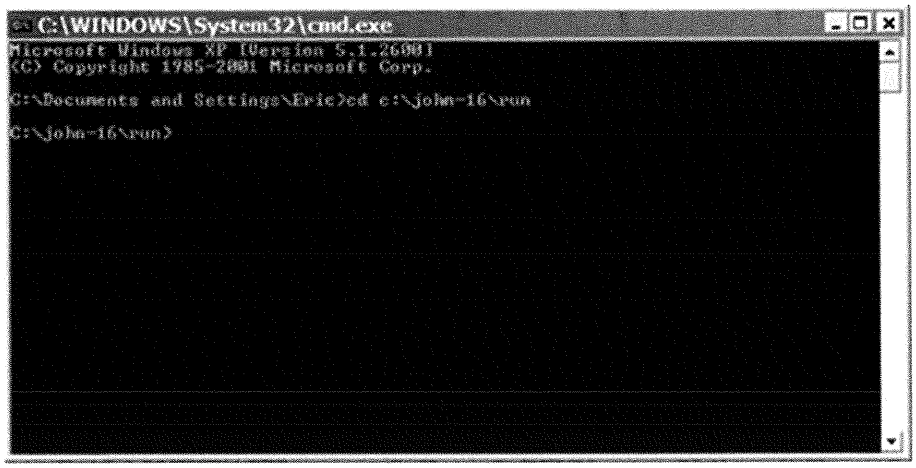
Running John the Ripper

To run John the Ripper, perform the following steps:

1. First, you need to prepare the Unix password file that John the Ripper will crack. Click *Start*, *Run*, and then type **cmd**.



2. Switch to the directory containing the John the Ripper files by typing **cd c:\john-16\run**.



Note: For this exercise, a passwd and shadow file are provided from a Red Hat system. The files are on a default installation of Red Hat in the /etc/ directory.

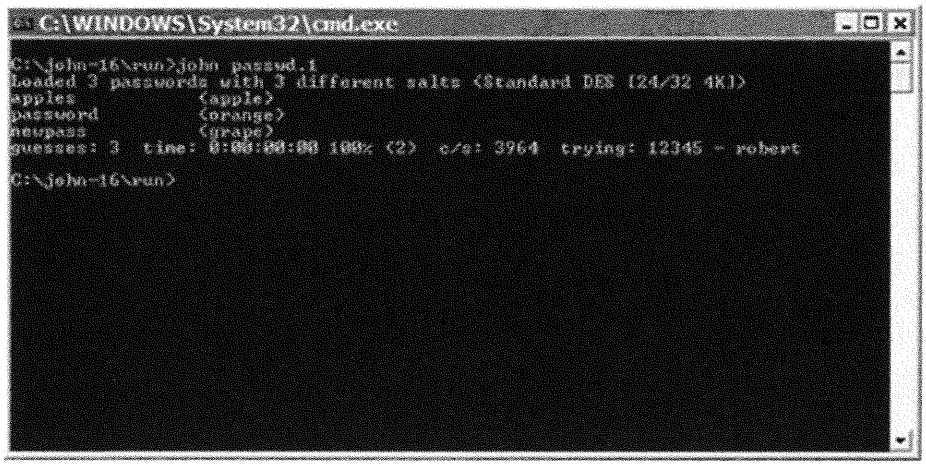
3. To combine the passwd and shadow files, type **unshadow passwd shadow > passwd.1**. To view the results of this command, type **type passwd.1**.

Unshadow is the name of the program, the password file is passwd, the shadow-file is shadow, and the output file is passwd.1. John requires that the passwd and shadow file be combined together before running the password cracking.

Cracking Passwords from Linux Machines

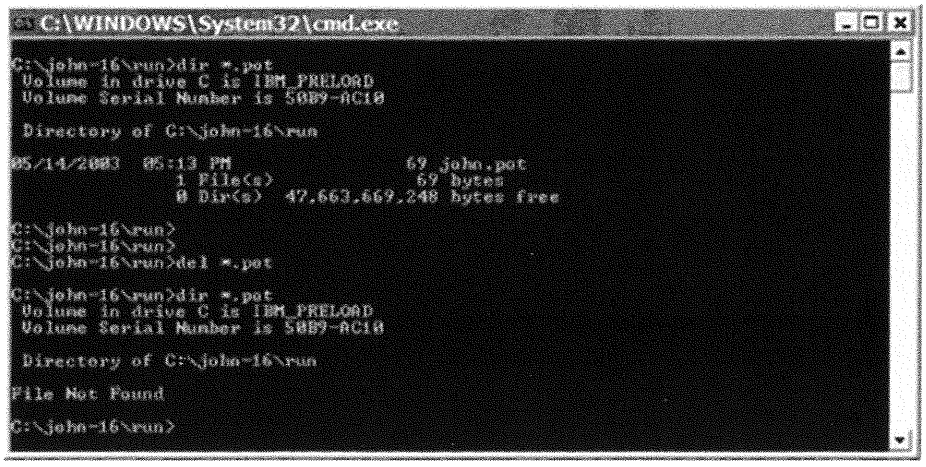
Following are steps for cracking passwords from Linux machines:

1. To decrypt the passwords in single mode, you execute John the Ripper pointing to the passwd.1 file by typing **john passwd.1**.

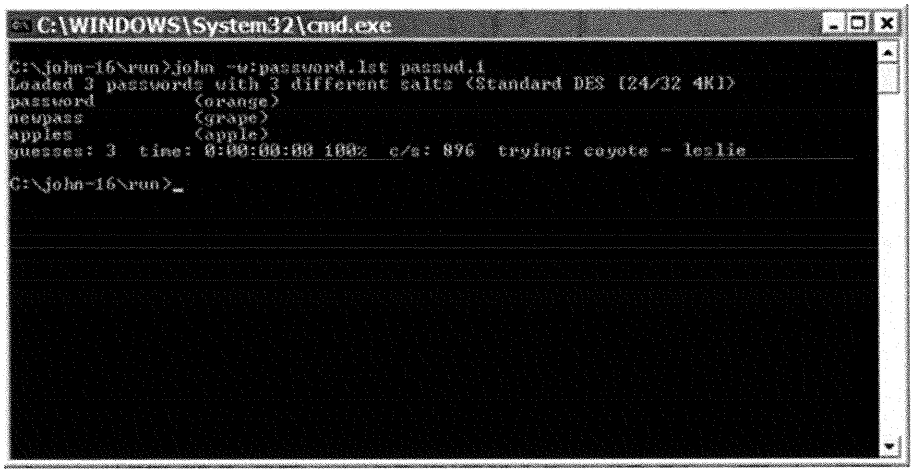


   The preceding screen shows that user ID apple has a password of apples, user ID orange has a password of password, and user ID grape has a password of newpass.

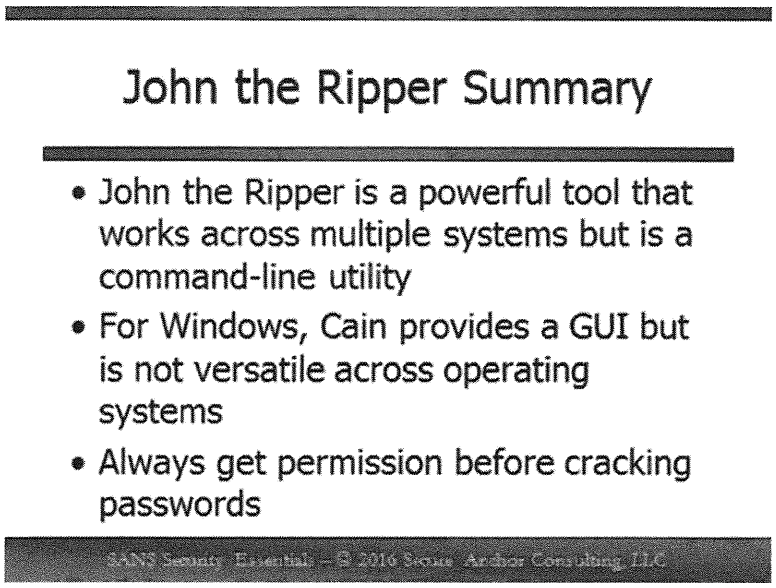
2. John the Ripper stores the passwords that have been cracked in a file called "john.pot." Therefore, if you try to crack the same passwords using different methods, it will not work because John the Ripper already knows what the passwords are. Before continuing, type **del *.pot**.



3. Now you can attempt to crack the passwords using the dictionary attack (also known as wordlist mode). This attack takes longer because it applies a list of passwords—all the words found in a dictionary. Type **john -w:password.lst passwd.1**.

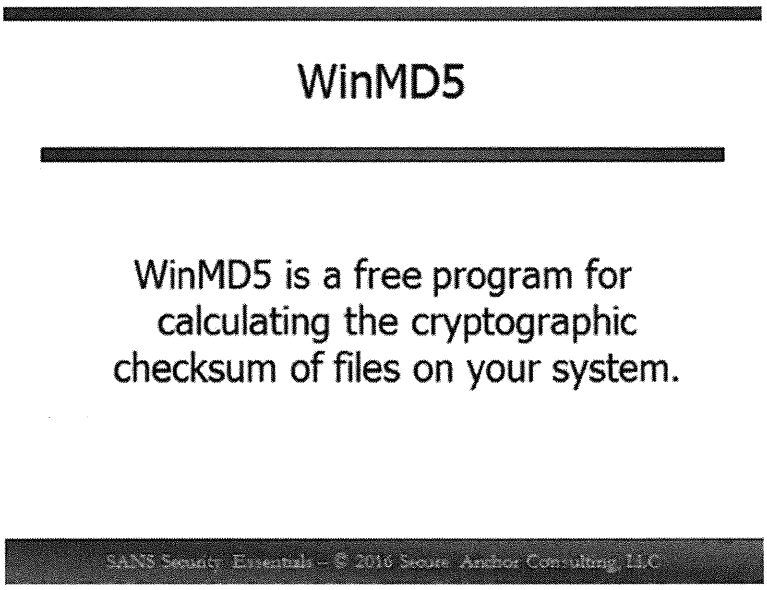


4. To perform a brute-force attack (also known as incremental), type **john -i passwd.1**. For strong passwords, this mode can take days, weeks, or months to run the password cracking.



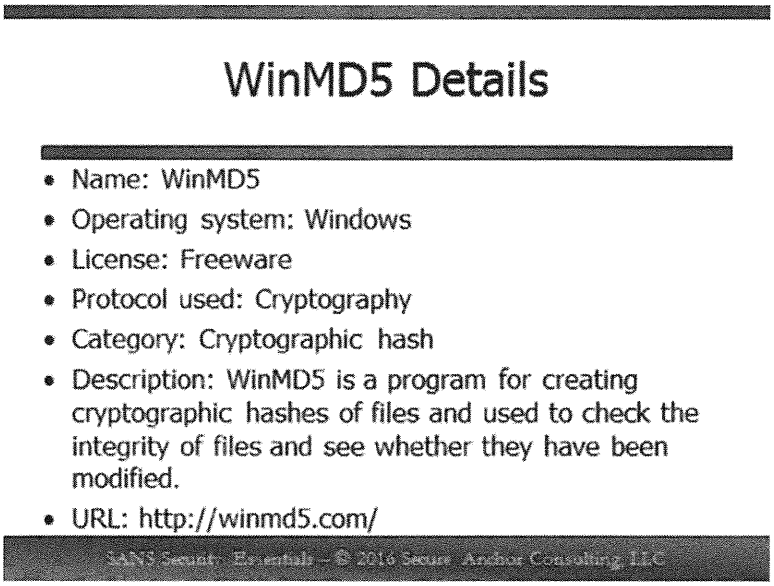
John the Ripper Summary

John the Ripper is a wonderful tool for network administrators to enforce a strong password policy. It is not designed for illegal activity, and prior to use, make sure you have written permission from senior management. John the Ripper performs different types of cracks: single mode, the one performed in this exercise; dictionary or wordlist mode, which applies a dictionary list of passwords for comparison; and brute-force or incremental mode, which is the slowest of the three modes and attempts every combination of letters and numbers. John the Ripper is portable for Unix or Windows, but it does not have a GUI interface.



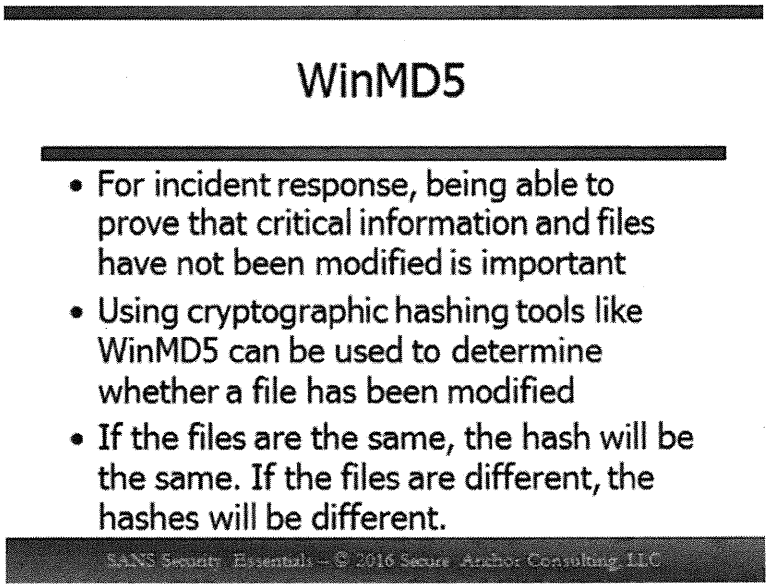
WinMD5

In performing incident response and in validating the security of a system, it is important to be able to prove that a file, or group of files, has not been modified. WinMD5 is a tool that can be used to calculate the integrity of a system. By calculating the checksums across a file, you can determine whether a file has been modified.



WinMD5 Details

This section intentionally left blank.

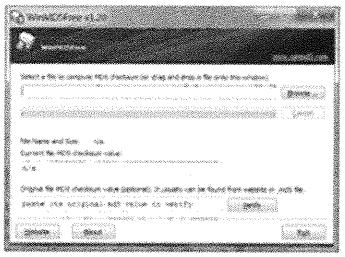


WinMD5

This section intentionally left blank.

Running WinMD5

- To run the program, just double-click *winmd5.exe* and the program will start

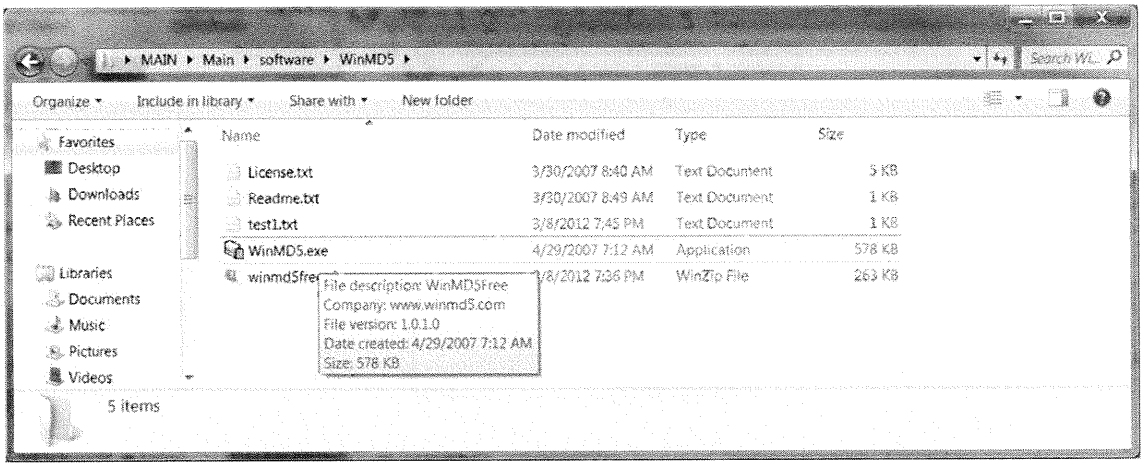


SANS Security Essentials – © 2016 Secure Anchor Consulting LLC

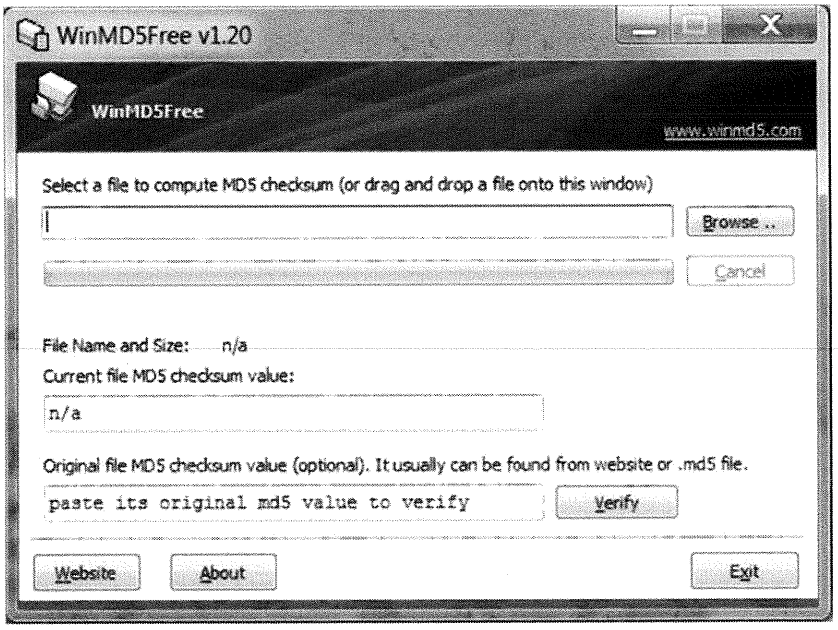
Installing and Running

WindMD5 is a simple program that does not have an installation program.

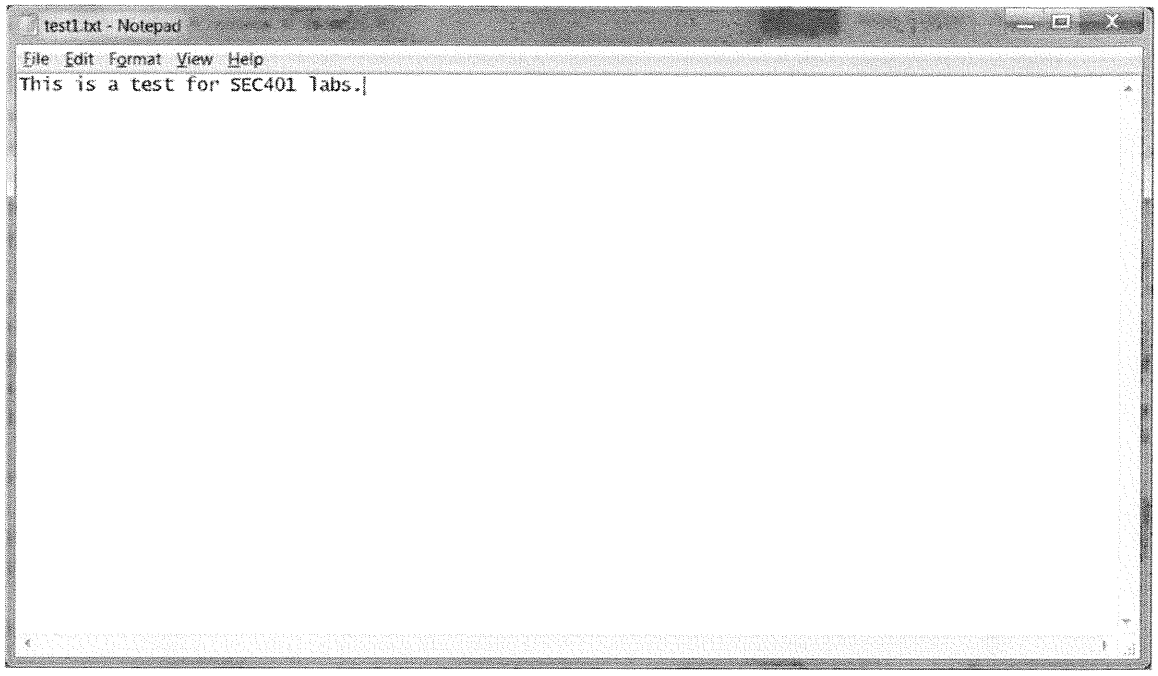
1. The WinMD5 directory can be copied from the DVD to the desktop. After the folder is copied over, open the folder.



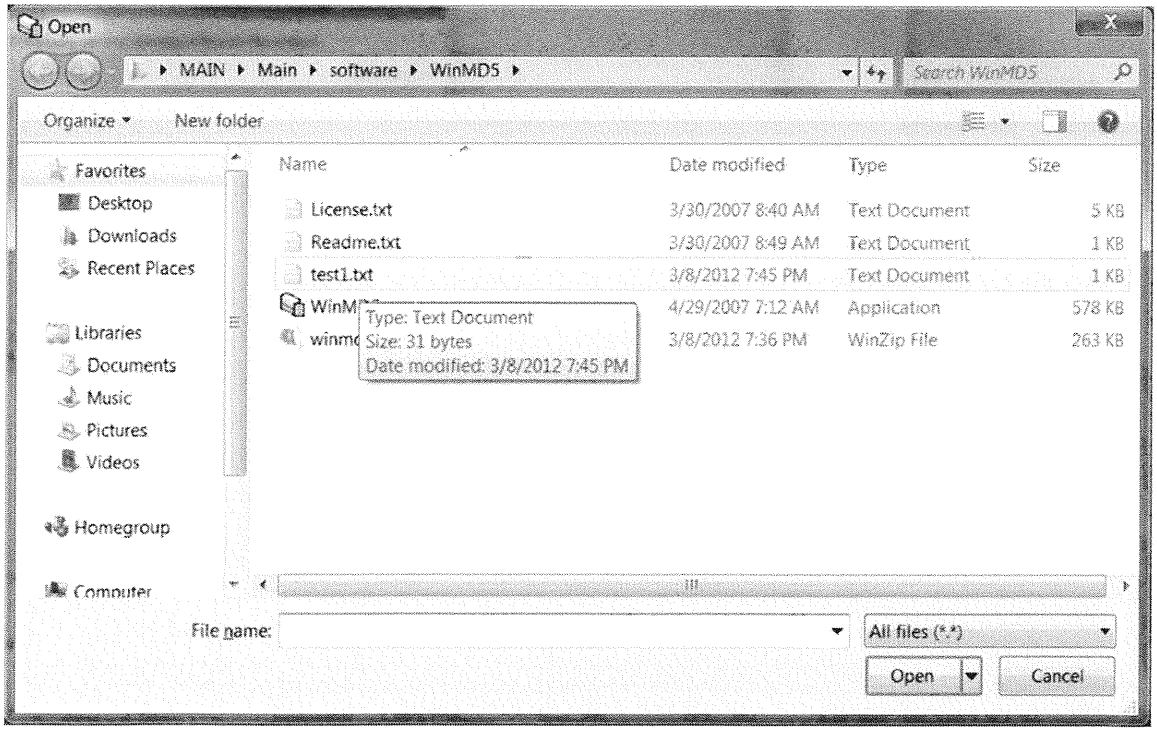
2. Double-click *WinMD5.exe* to start the program.



3. After the MD5 checker program displays, create a new document called **test1.txt** and type some information into the file. Save the file when you are done.

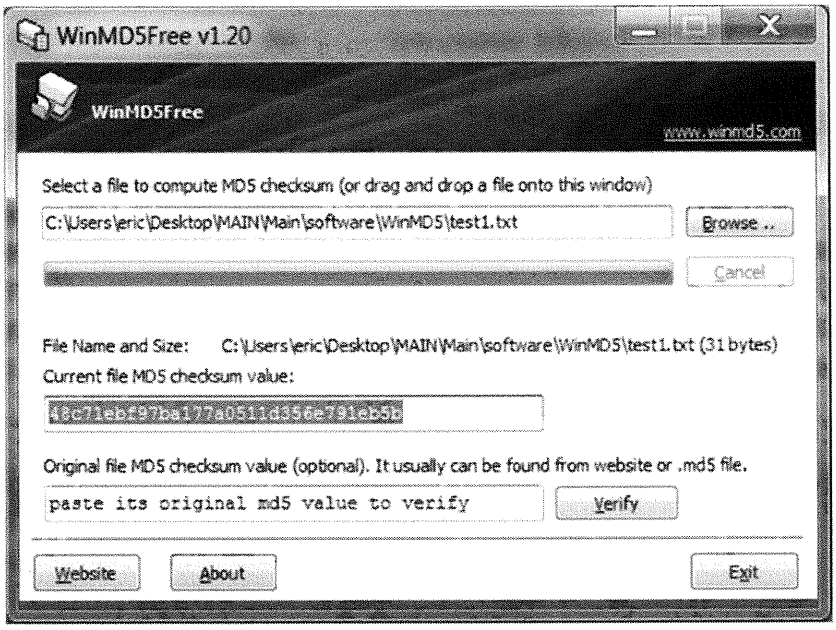


4. In the WinMDFree screen, under browser, select the *test1.txt* file you created. After selected, click *Open*.

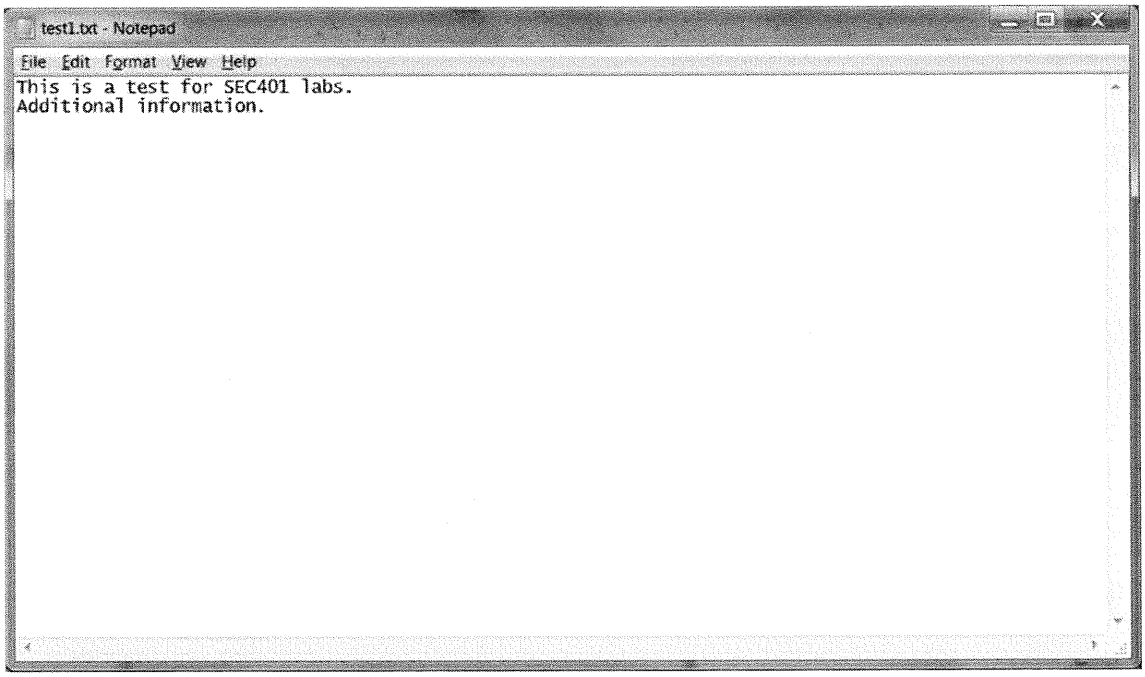


5. If you typed the same text as the previous step, your checksum should be as follows: 48c71ebf97ba177a0511d356e791eb5b

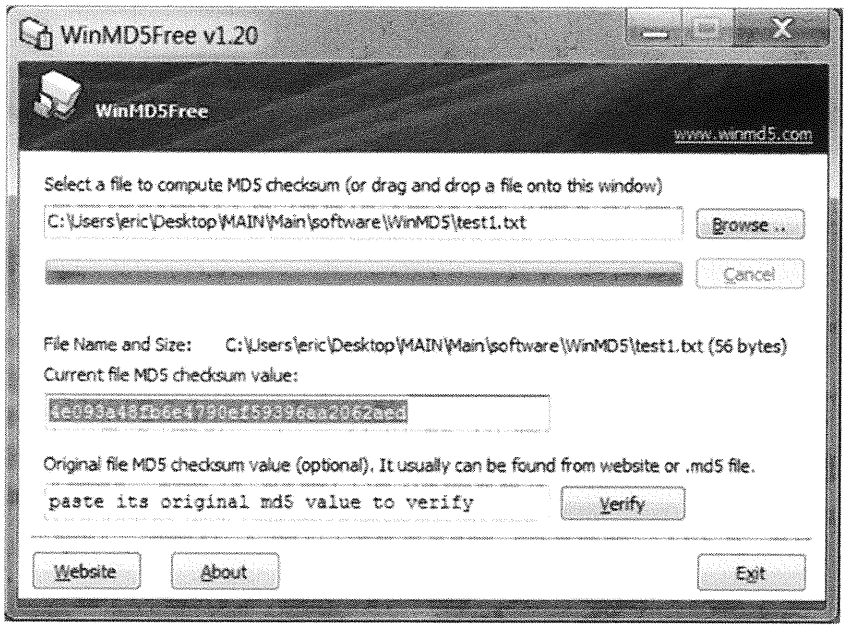
6. If it is different, do not worry because spaces or hidden characters can change the checksum value.



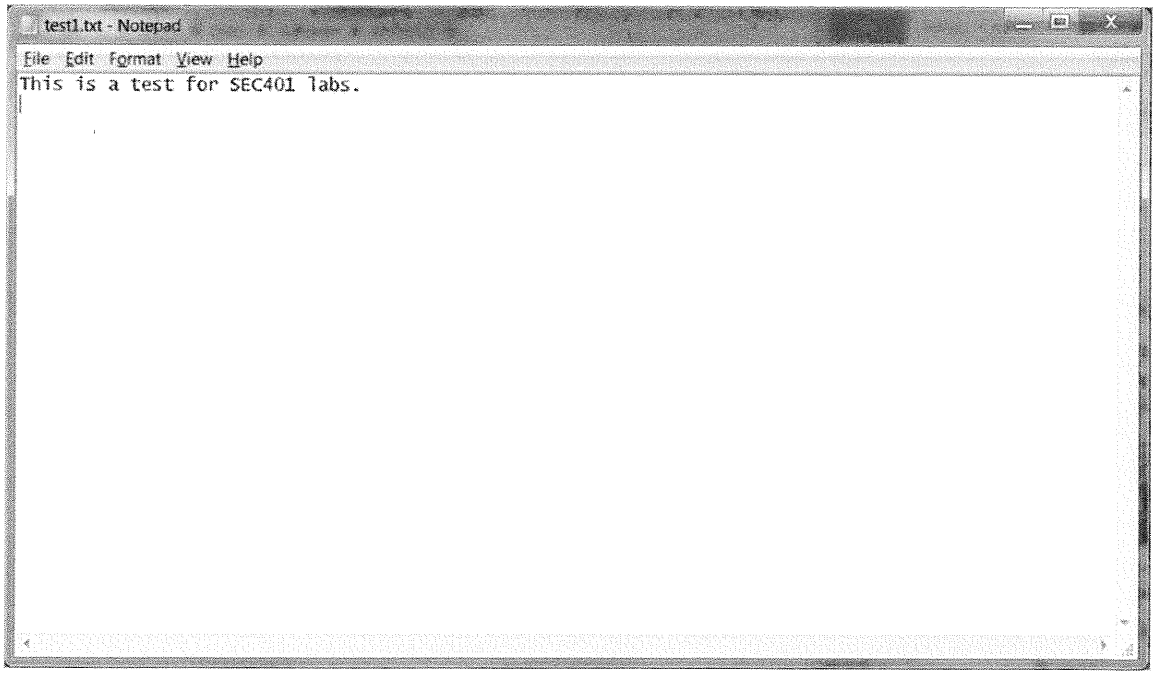
7. Now edit the test1.txt file to add additional text and save the file.



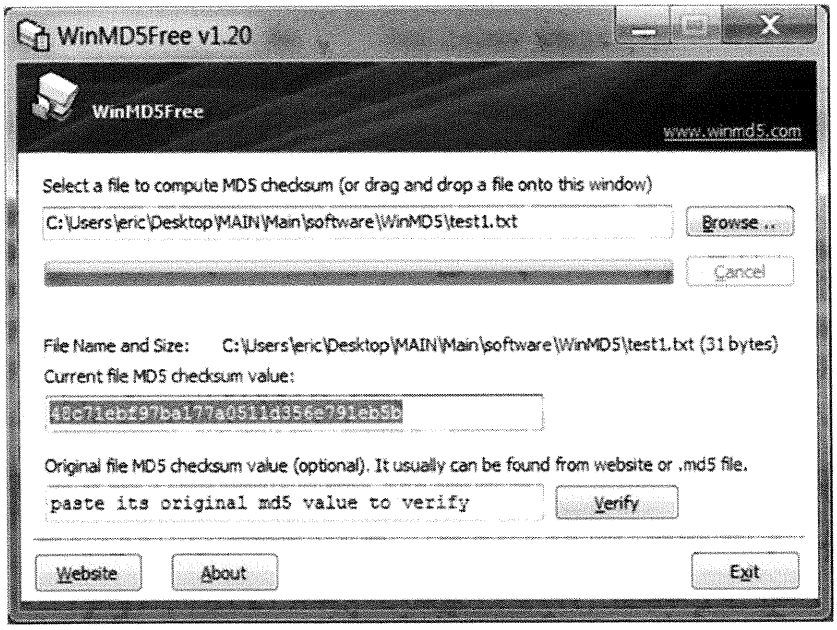
8. In WinMD5Free, click *Browse*. Select the file again and click *Open*. The checksum is now different because the file changed: 4e093a48fb6e4790ef59396aa2062aed



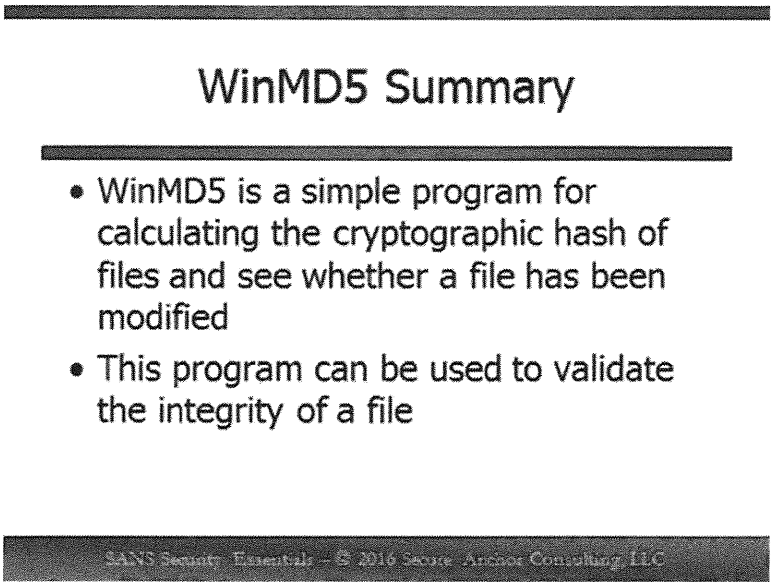
9. Edit the file to its original state, and then save the file.



10. In WinMD5free, click *Browse*, select the file again, and click *Open*. Notice that the checksum is back to the original value. If they are not the same, it could mean that there are hidden characters in the file based on the edits that were performed.



This shows how hashing can be used to verify the integrity of a value. The only time the outputs are supposed to be the same is if the inputs are the same. Now remember, collisions can occur but because they are not predictable, they are an acceptable level of risk.



WinMD5 Summary

This section intentionally left blank.

SECURITY 401 –
SANS Security Essentials

The End

This section intentionally left blank.