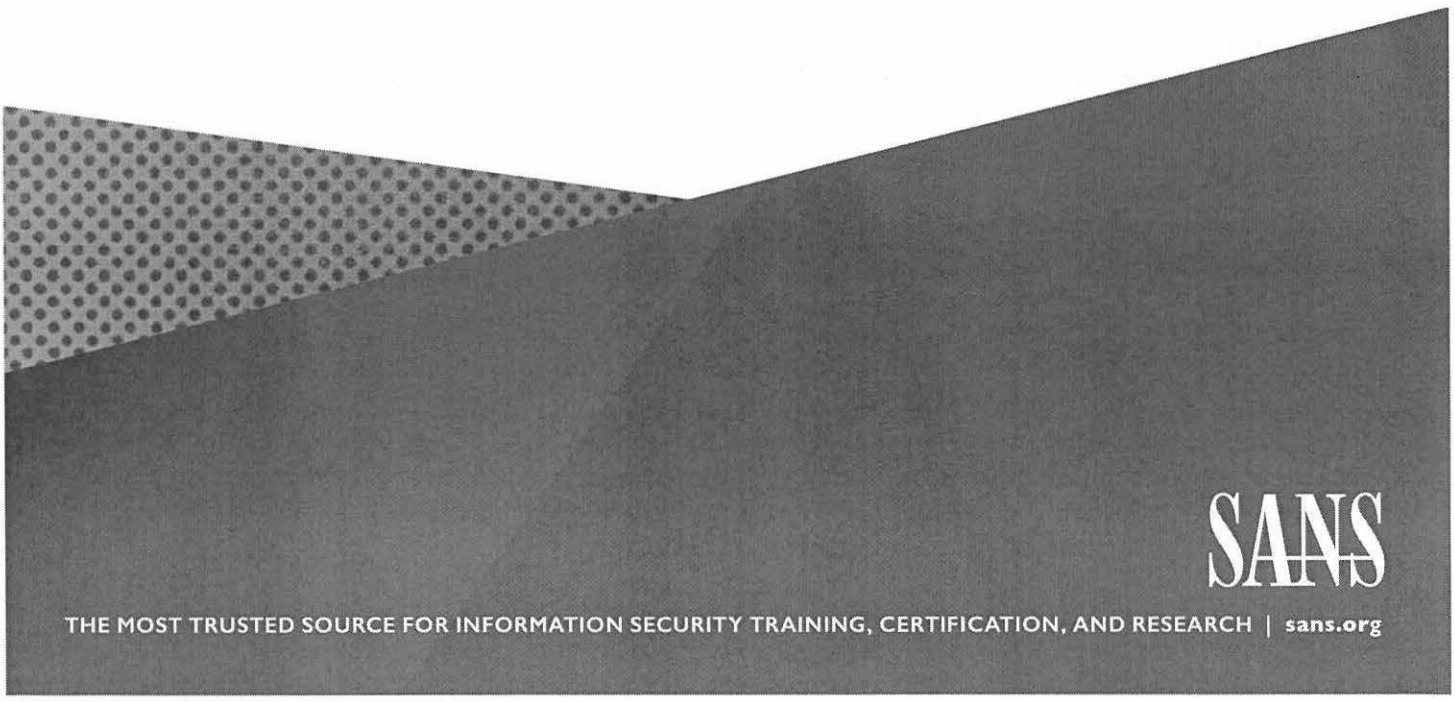
SEC560 | NETWORK PENETRATION TESTING AND ETHICAL HACKING

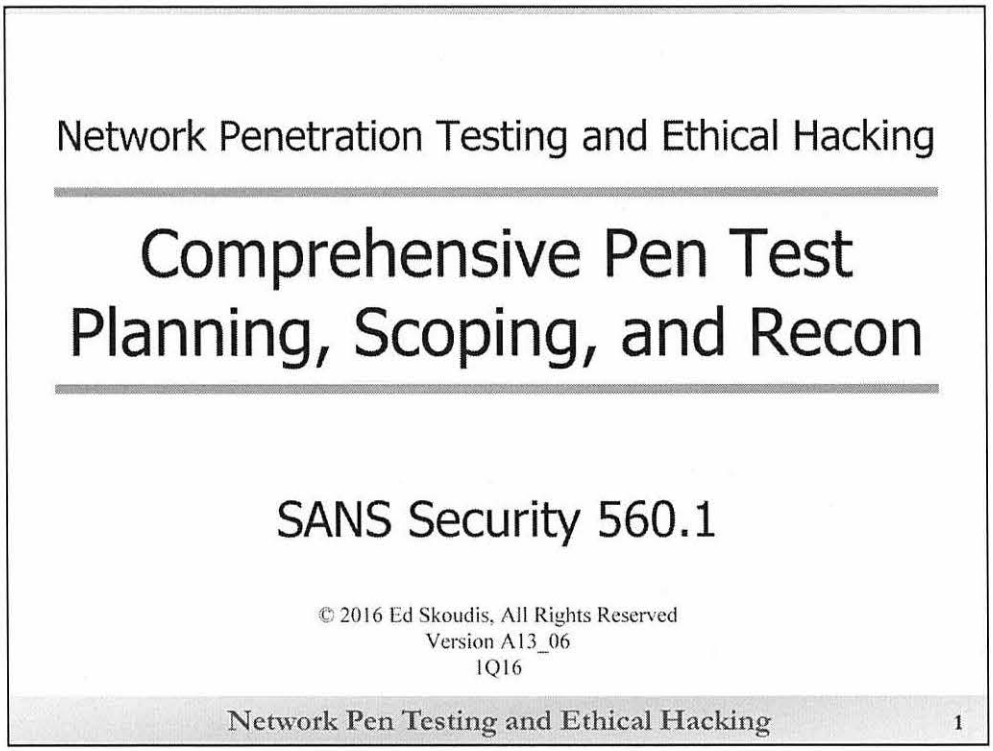
560.1

Comprehensive Pen Test Planning, Scoping, and Recon



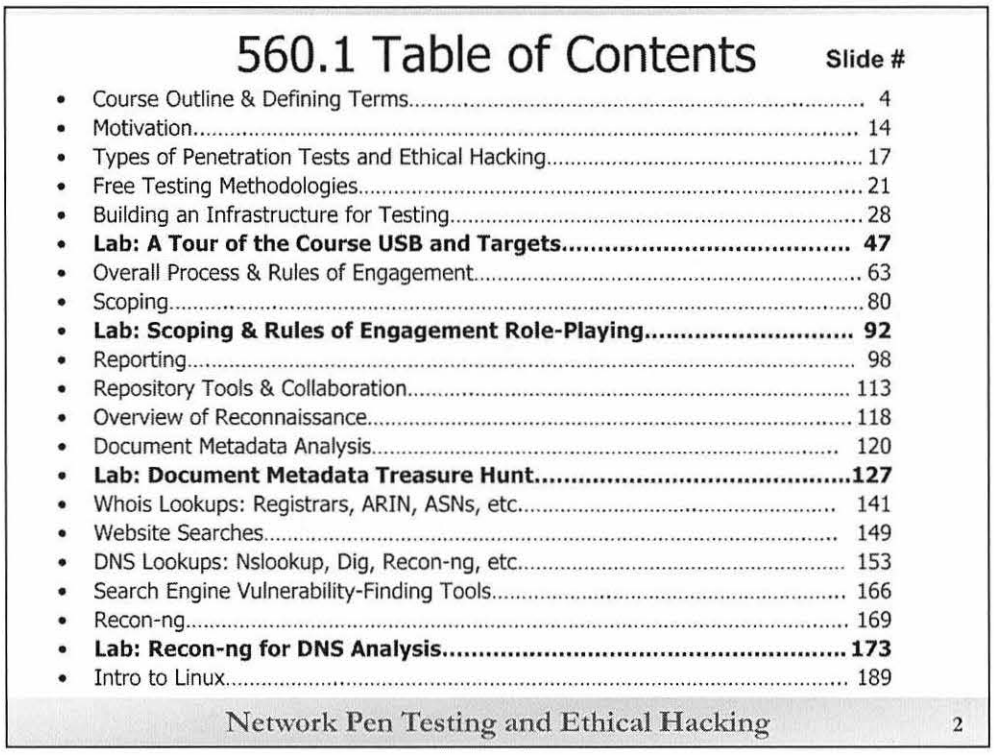
SANS

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

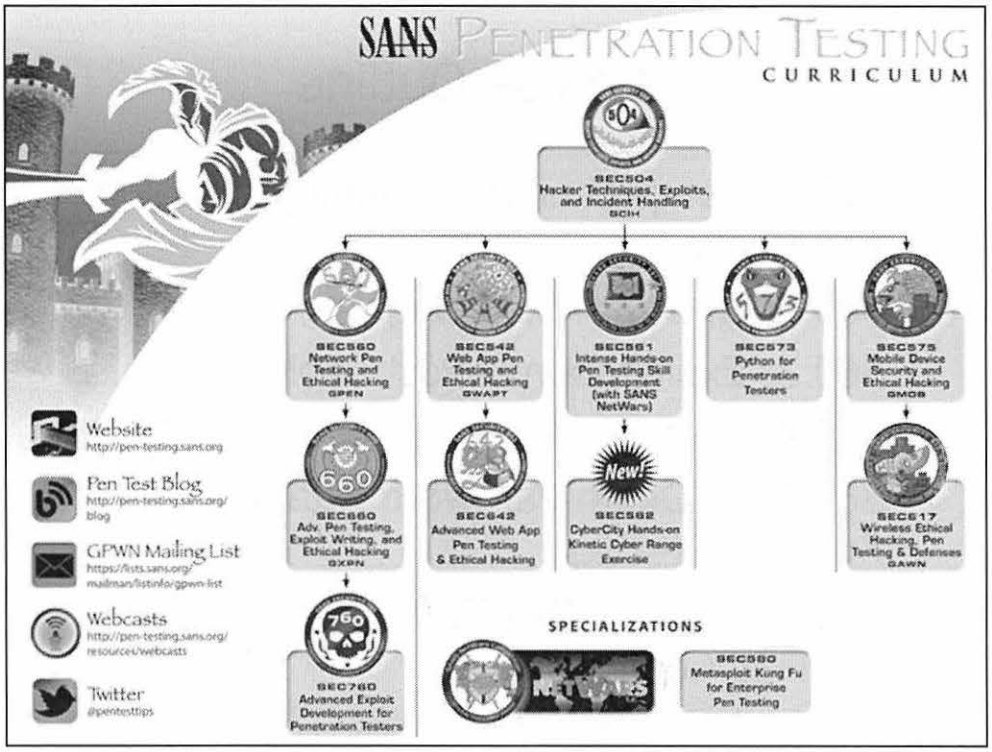


Hello and welcome! The purpose of this course is to prepare you to perform ethical hacking and penetration testing projects in a professional, safe, and repeatable manner for your organization. Also, by covering some powerful attack techniques, this course is designed to help all security professionals (not just penetration testers) improve the security stances of their organizations. Throughout this course, we cover hundreds of tools and techniques, detailing how you can use them to find vulnerabilities in your organization to help improve your organization's security. We include hands-on labs throughout, culminating in a full-day, capture-the-flag penetration test lab for the entirety of 560.6.

Let's keep this session interactive. If you have a question, please let the instructor know. Discussions about relevant topics are incredibly important in a class like this because we have numerous attendees with various levels of skill attending the class. Share your insights and ask questions. The instructor does reserve the right, however, to take a conversation offline during a break or outside of class in the interest of time and applicability of the topic.



This slide is a table of contents and also acts as an overview of what we will discuss throughout 560.1.

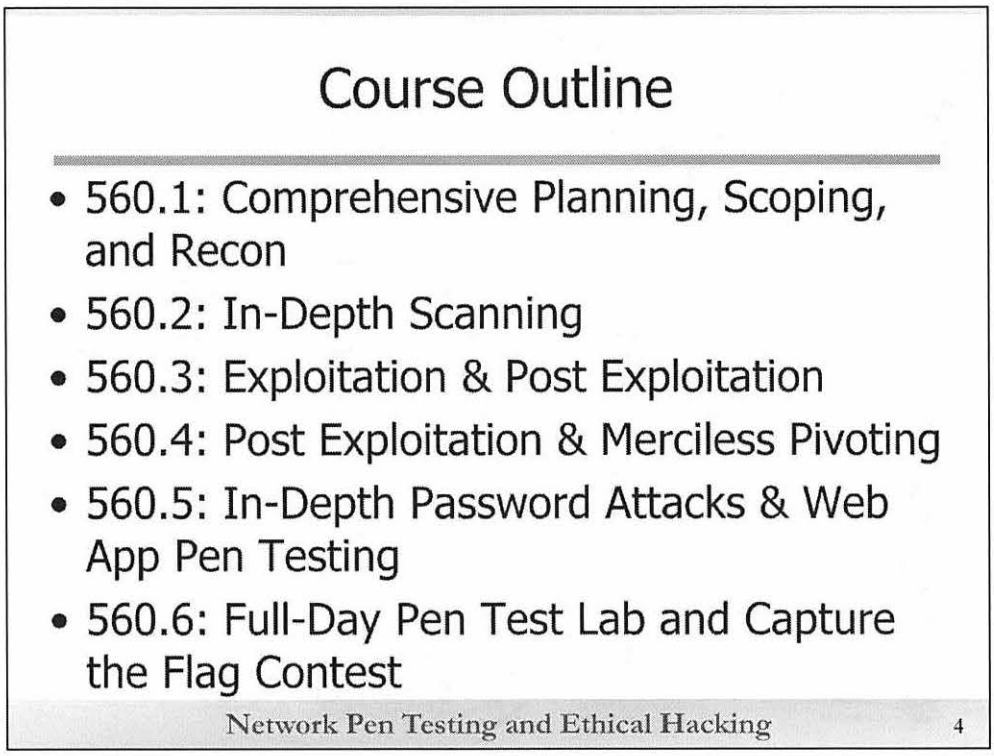


Once you complete 560, you might want to further develop your skills with other in-depth courses in the SANS Penetration Testing Curriculum. Each of these courses was created to give you the skills you can apply directly in doing your job as an information security professional. Each of these 6-day courses is available at live conferences, private courses, across the Internet via SANS vLive, and in the SANS OnDemand system. Following is a list of the courses:

- **SANS Security 504: Hacker Techniques, Exploits, and Incident Handling**: This session, one of SANS' most popular courses, focuses on how to respond to computer attacks using a detailed incident response methodology.
- **SANS Security 542: Web App Pen Testing and Ethical Hacking:** If you are interested in focusing on web application penetration testing, this course delivers the skills you need to thoroughly analyze web apps.
- **SANS Security 561: Intense Hands-On Skill Development for Penetration Testers:** This course is 80%+ hands-on, helping you build serious pen test skills quickly.
- **SANS Security 562: CyberCity Hands-On Kinetic Cyber Range:** This course is also 80%+ hands-on, with missions in the SANS CyberCity kinetic range, which features a miniature city with a real power grid and other components.
- **SANS Security 573: Python for Pen Testers:** This offering helps penetration testers master the Python programming language, and it shows attendees how to build custom tools and tweak existing tools to add more functionality.
- **SANS Security 575: Mobile Device Security and Ethical Hacking:** This course provides the in-depth knowledge that organizations need to design, deploy, operate, and assess their mobile environments, including smartphones and tablets.
- **SANS Security 617: Wireless Ethical Hacking, Pen Testing, and Defenses:** This fantastic course provides in-depth information about attacking and defending wireless LANs, Bluetooth devices, Zigbee, and more.
- **SANS Security 642: Advanced Web App Pen Testing and Ethical Hacking:** This course builds on SANS Security 542, providing advanced, hands-on skills in web application analysis and penetration testing.
- **SANS Security 660: Advanced Penetration Testing, Exploits, and Ethical Hacking:** This exciting and advanced course helps penetration testers take their skills to the next level, and it covers topics such as NAC bypass, route injection, domain compromise, and exploit development to dodge modern OS defenses such as DEP and ASLR.
- **SANS Security 760: Advanced Exploit Development for Penetration Testers:** This is our deepest technical offering, with Windows kernel manipulation, patch diffing, and many other deep attacks and exploits.

In addition, SANS offers a 2-day course related to penetration testing skills in demand today—SANS Security 580—which focuses on the amazing Metasploit tool.

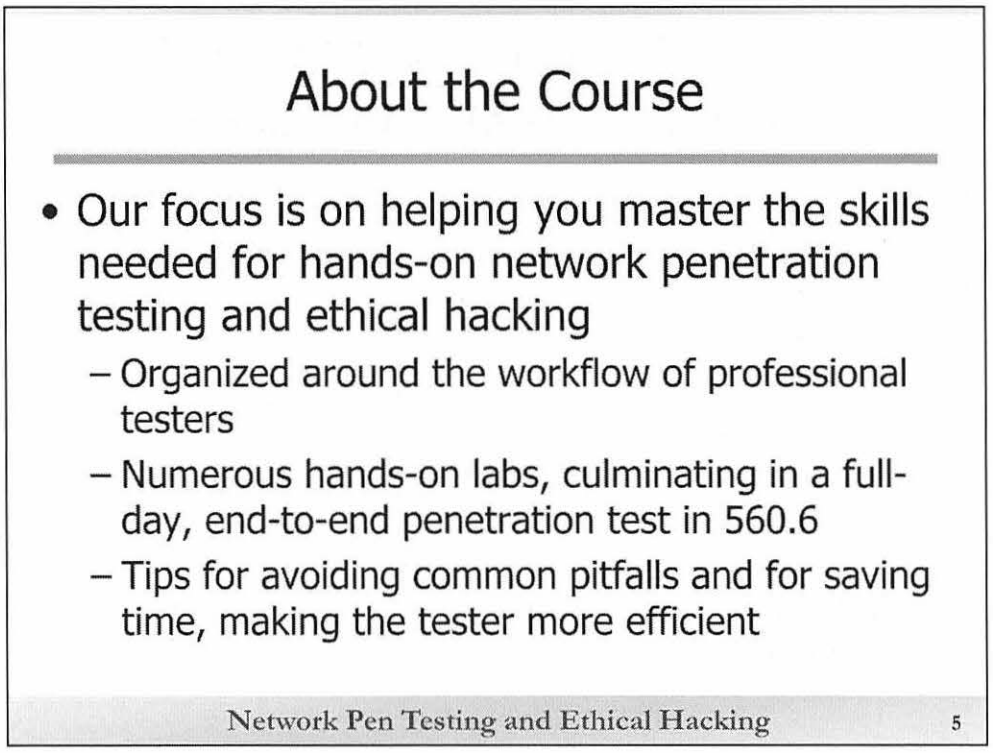
Finally, NetWars is an information security challenge environment, in which participants develop and measure the effectiveness of their offensive and defensive skills.



This course is divided into several sections, each designed to prepare you in a vital aspect of network penetration testing and ethical hacking.

- 560.1 sets the stage, defining terms and providing a detailed discussion of the planning process, including building a penetration testing and ethical hacking infrastructure, establishing ground rules for testing, and scoping projects. This section also covers the Reconnaissance phase of a test in detail.
- 560.2 zooms into scanning, covering the tools and techniques that professional penetration testers and ethical hackers need to master to find target machines, openings on those targets, and vulnerabilities.
- 560.3 deals with exploits, talking about the different categories of exploits, the manner in which they are packaged, and how to use exploitation tools. This section of the class also covers some of the common pitfalls that come up after exploitation and how to avoid these problems, including tips for antivirus evasion.
- 560.4 focuses on post-exploitation, covering a variety of techniques pen testers can apply after they've compromised a target environment to pillage target organizations, so we can better understand the business risk. We also cover various ways to pivot through compromised machines in a target environment.
- 560.5 deals with in-depth password attacks (including pass-the-hash attacks) and web application penetration tests. Passwords are often one of the weakest areas in target organizations, and attacking them skillfully is an important part of most penetration tests. Furthermore, although this is not a web app pen test course, network penetration testers and ethical hackers are often called upon to analyze web apps.

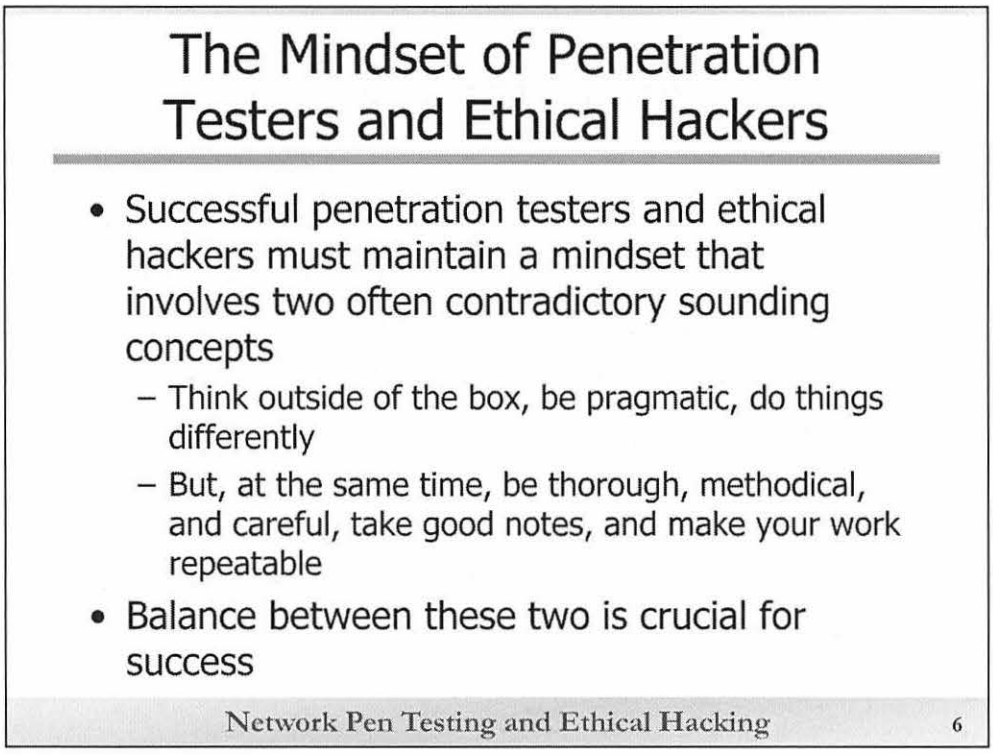
This all leads to 560.6, in which we apply the concepts from throughout the course in a full-day, end-to-end penetration test lab, which includes a Capture the Flag (CtF) contest. The skills you master from labs throughout the class will be applied in this lively contest on the last day.



The overall objective for this course is to help prepare you with the skills need to perform network penetration testing and ethical hacking. Some people who take this class are professional penetration testers looking for some extra tips and tools for their arsenal. Others have never hacked a box before and want to get started. Others are cyber defenders who want to learn more about the offensive skills attackers use. Some course attendees are forensics experts looking to better understand the attacks they will analyze. We welcome attendees from across the spectrum of information security professionals. We have strived to develop the materials to help you master the skills of a network penetration tester and ethical hacker regardless of where you sit on that spectrum.

This course is organized around the workflow of a professional penetration tester and ethical hacker, describing the various steps and options a tester takes at each step. Note that the general flow of work, however, isn't set in stone. Good testers are pragmatic, often improvising based on the particulars of a given project when the opportunity arises. The class includes numerous hands-on labs, each of which is designed to impart an important skill that network penetration testers and ethical hackers require.

The course is also chock-full of tips for avoiding common pitfalls that network penetration testers and ethical hackers face. Based on input from numerous professional penetration testers who have learned these lessons the hard way, these tips throughout the course are designed to help you maximize the effectiveness of your own penetration practices. Also, many of these tips are designed to save you a lot of time, making you more efficient. Often, when testing, you need to achieve some goal. One way of going about that goal may take 3 hours and work only 10% of the time, whereas another method might take 3 minutes and have a 90% success rate. Following the tips of this class can help you focus your valuable time on the latter.



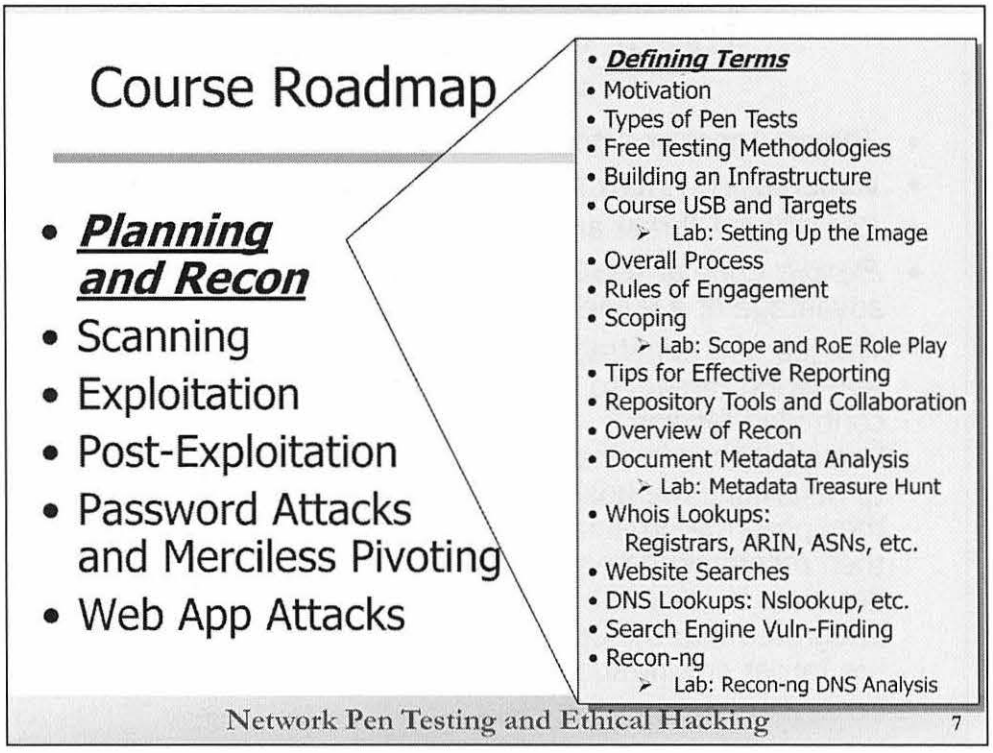
At the outset of this class, let's briefly explore the mindset of penetration testers and ethical hackers. A noted penetration tester, someone whose name you would likely recognize but who has requested anonymity, said: *"We break computers, making them do stuff that their designers, implementers, and system administrators didn't plan on them doing."*

That's what our job is: Find flaws that enable attackers to do evil on target machines so that an organization can better understand its business risks and resolve vulnerabilities before mayhem ensues. However, to successfully achieve that goal, penetration testers and ethical hackers must maintain a mindset that involves two often-contradictory-sounding concepts.

First, a penetration tester or ethical hacker must be flexible and pragmatic, thinking outside of the box. To be successful, you need to think differently than most traditional system administrators or network architects, trying to solve problems in often untraditional ways.

But, at the same time as you wield your pragmatic style, you have to be thorough, methodical, and careful. Your work, to be valuable, must be understandable and reproducible so that the target organization can understand its vulnerabilities and risks and take action to mitigate the flaws. You need to take good notes and produce a high-quality report that presents your findings in a digestible form for people who don't perform penetration testing or ethical hacking professionally—people who may not share your pragmatic, think-differently mindset.

Some people struggle with this mindset, erring by allowing one side to dominate over the other. However, many people can resolve this conflict between these two mindsets, balancing them. To be a successful penetration tester, you need to strive for this balance.

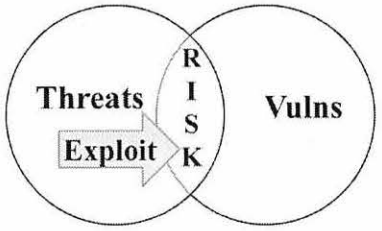


To start the session, we need to define some terms so that the terminology is consistently used throughout the rest of the class. What is ethical hacking? How is it associated with penetration testing? How do vulnerability scans and a penetration tests differ? We address each of these questions next.

It is important to note that different people use the various terms we define in different ways. We present a set of definitions that are common but not universal. In other words, we introduce this common terminology so that we can be consistent throughout this course and with the most common use of these terms. However, keep in mind that usage can vary for your organization or with some of the enterprises that you test.

Threat Versus Vulnerability Versus Risk

- *Threat:* Agent or actor that can cause harm
- *Vulnerability:* A flaw someone can exploit to cause harm
- *Risk:* Where threat and vulnerability overlap
- *Exploit:* Code or technique that a threat uses to take advantage of a vulnerability
- The job of a penetration tester is to model the actions of real-world threats to find vulnerabilities ... and then, in a controlled fashion, to exploit these vulnerabilities to determine the business risk they pose the organization... and then recommend appropriate defenses that can be integrated into the operations of the target organization



Network Pen Testing and Ethical Hacking 8

Ethical hacking and *penetration testing* are tools for dealing with threats, vulnerabilities, risks, and exploits. Many people in the information security business throw around these terms interchangeably, often confusing threats with risk, or vulnerabilities with exploits. Each has a distinct meaning, though, and the terms should be applied carefully.

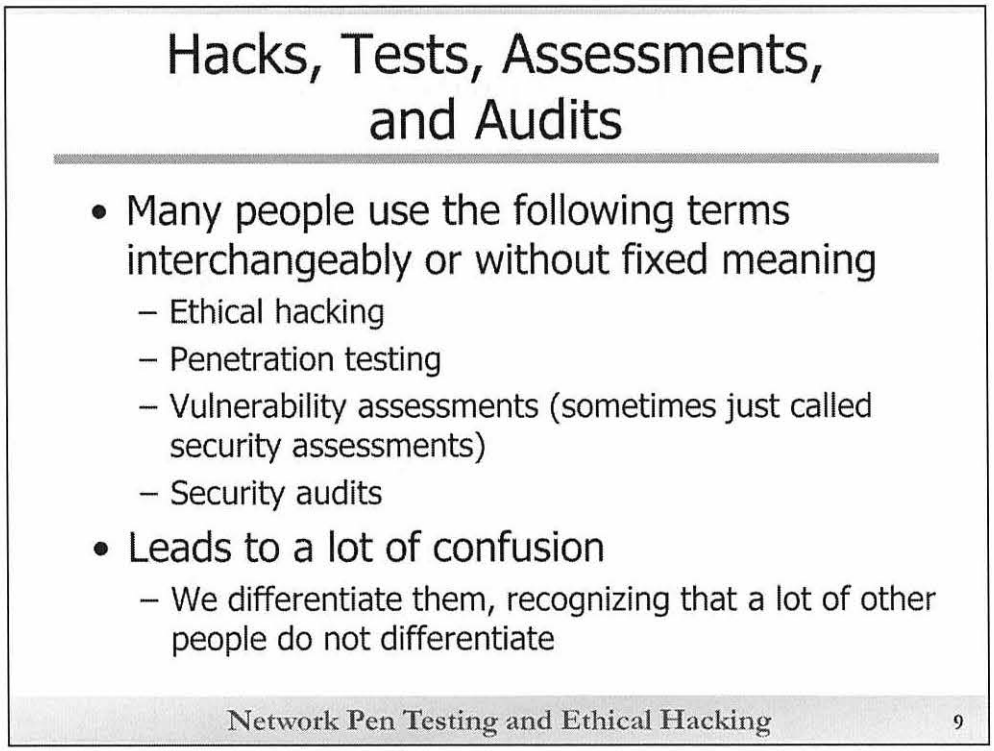
A *threat* is an actor or agent that may want to or actually can cause harm to the target organization. Threats include organized crime, spyware companies, and disgruntled internal employees who start attacking their employer. Worms and viruses also represent a threat because they could cause harm in your organization even without a human directing them to do so by infecting machines and causing damage automatically. In this course, we often refer to threats generically as "attackers" or "bad guys."

A *vulnerability* is a flaw in the environment that an attacker can use to cause damage. Vulnerabilities can exist in numerous arenas in environments, including architectural design, business processes, deployed software, and system configurations.

Risk is where threat and vulnerability overlap. That is, a risk occurs when systems have a vulnerability that a given threat can attack.

An *exploit* is the vehicle by which the attacker uses a vulnerability to cause damage to the target system. The exploit could be a package of code that generates packets that overflow a buffer in software running on the target. Alternatively, the exploit could be a social engineering scheme whereby the bad guy talks a user into revealing sensitive information, such as a password, over the phone.

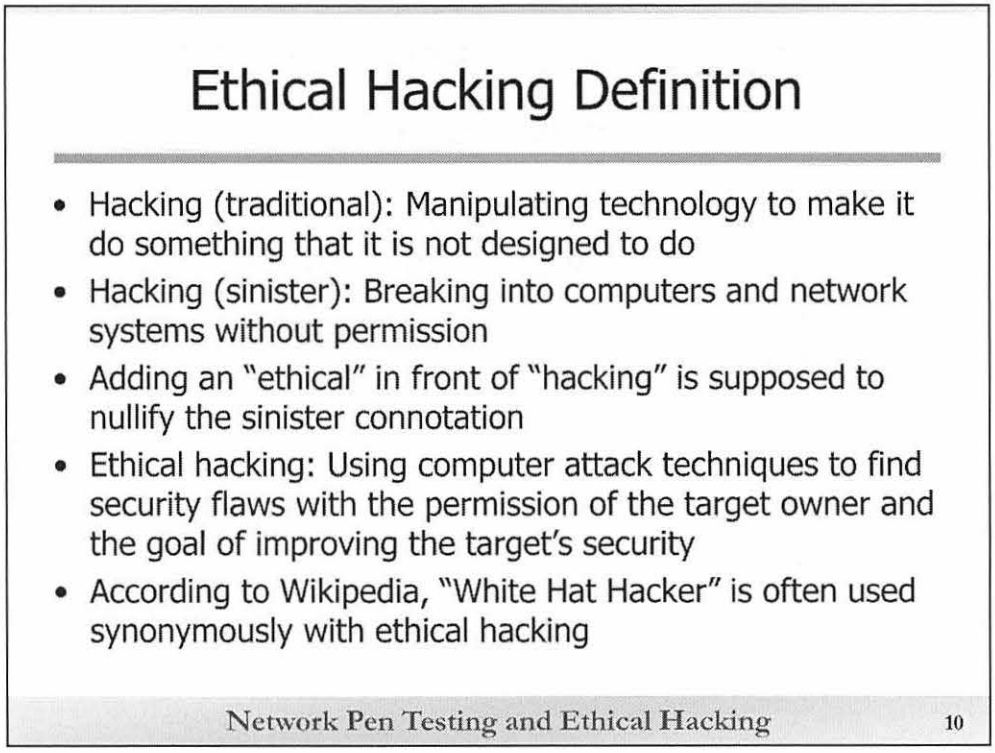
As security professionals, we have to work hard to minimize this risk by minimizing vulnerabilities and blocking threats. That's what penetration testing is all about: Model the activities of real-world threats to discover vulnerabilities. Then, through controlled exploitation, attempt to determine the business risk associated with these flaws. Then, recommend appropriate defenses. These recommendations must apply in light of the operations environment of the target organization. If you do this properly, you stand a significant chance of improving the security stance of the target organization.



There is another set of terms that many information security practitioners use interchangeably, which results in a lot of confusion. The following terms are associated with what an ethical hacker or penetration tester actually does on a day-to-day basis:

- Ethical hacking
- Penetration testing
- Vulnerability assessments (and security assessments)
- Security audits

Although these terms are often used interchangeably, they do have subtle distinctions that we should observe.



The term *hacking* means different things to different people. Traditionally, hacking refers to exploration of technology, trying to understand it at a deep level to manipulate it into doing something that it was not designed to do. Early hackers of this kind were often hobbyists or academics, with the noble goal of using technology in interesting and innovative ways.

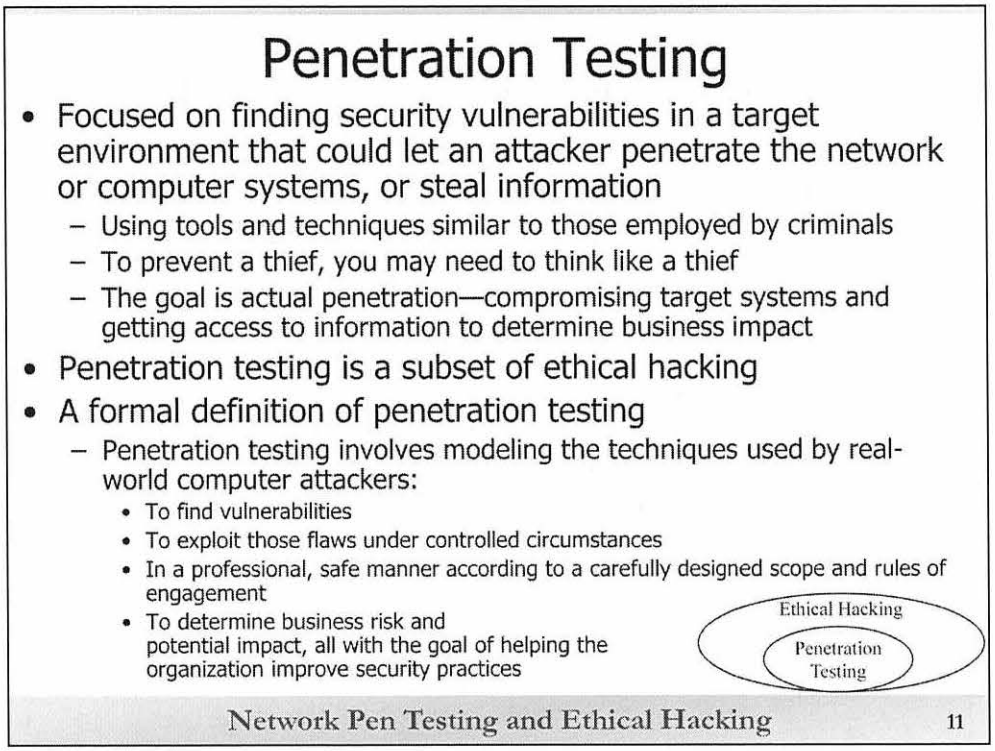
Unfortunately, most people today think of hacking in more sinister terms: breaking into computer systems and accounts without the permission of their owners to make money illicitly or cause damage.

People started to use the term *ethical hacker* to refer to individuals who applied the process of breaking into computer systems but with the wholesome purpose of finding security vulnerabilities so that they could be fixed. The hope is that the "ethical" prefix will nullify the sinister notions of hacking. *Ethical hackers* use some of the techniques and processes of the bad guys, but in a professional manner, with permission of the owners of the target systems, to try to improve the security of their targets.

Pulling all this together, we get the following definition of ethical hacking: *Ethical hacking* is the process of using computer attack techniques to find security flaws with the permission of the target owner and the goal of improving the target's security.

Throughout this class, we use the phrase in this non-evil sense.

According to Wikipedia, ethical hacking is synonymous with White Hat hacking. This is to differentiate it from the Black Hat hacking, which involves the sinister motive.

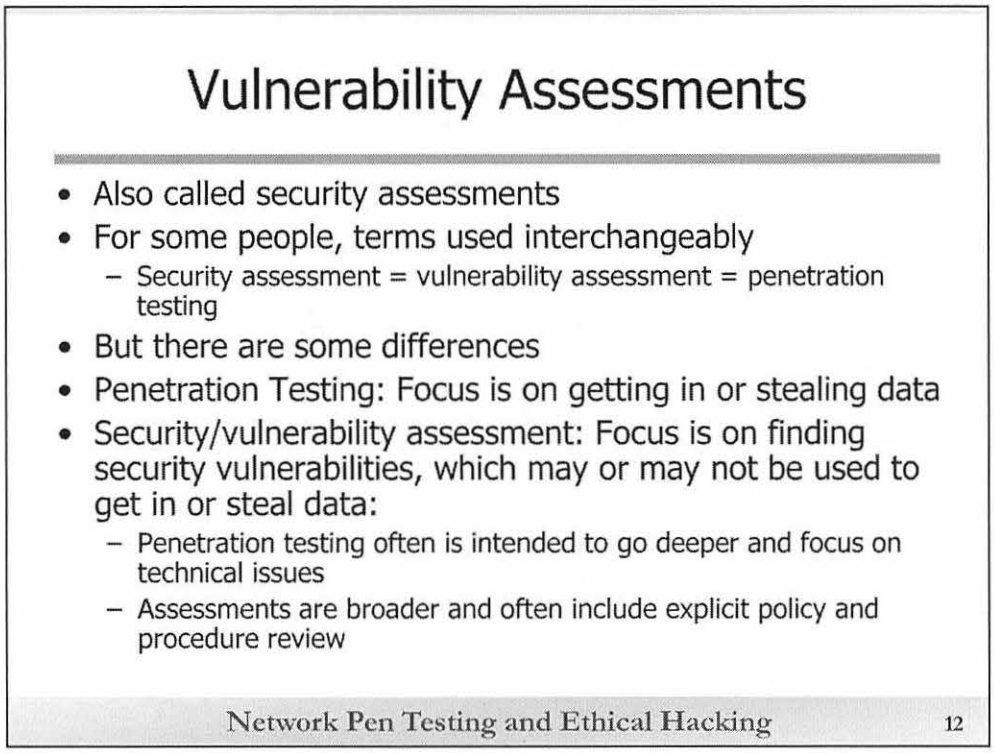


Penetration testing is closely related to ethical hacking. Indeed, throughout this course, we often refer to ethical hacking and penetration testing to bundle the terms together.

Some people use the term *ethical hacking* to mean the general process of using hacker techniques for good, which includes vulnerability discovery in a target organization's network, software product vulnerability research, and other tasks. In this view, *penetration testing* is a more narrowly focused phrase, dealing with the process of finding flaws in a target environment with the goal of penetrating systems, taking control of them. *Penetration testing*, as its name implies, is focused on penetrating the target organization's defenses, compromising systems, and getting access to information.

To summarize, ethical hacking is an expansive term encompassing all hacking techniques used for good, whereas penetration testing is more focused on the process of finding vulnerabilities in a target environment. In this view, penetration testing is a subset of ethical hacking.

According to the Top 20 Critical Controls, a formal definition of penetration testing is as follows: Penetration testing involves modeling the techniques used by real-world computer attackers to find vulnerabilities, and, under controlled circumstances, to exploit those flaws in a professional, safe manner according to a carefully designed scope and rules of engagement to determine business risk and potential impact all with the goal of helping the organization improve security practices.

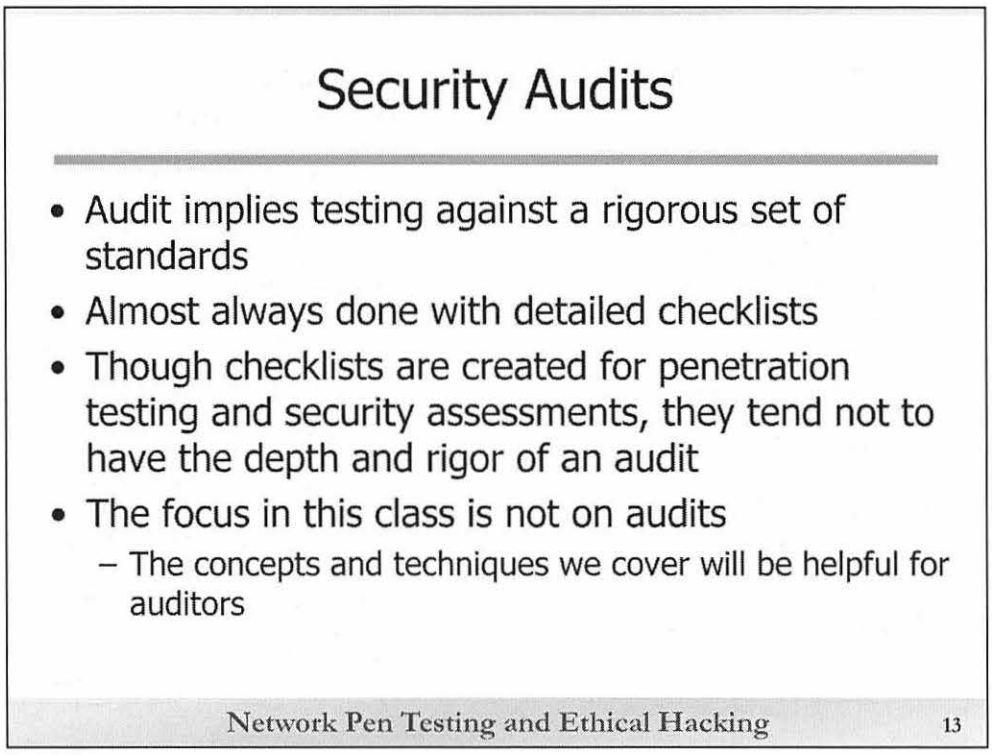


Many people use the phrases *vulnerability assessment* or *security assessments* to describe the work done by penetration testers and ethical hackers; but, there is a subtle distinction between a penetration test and a security assessment.

A *penetration test* is focused on getting in or stealing data. The emphasis is on penetration of the target environment by exploiting discovered vulnerabilities.

Vulnerability assessments and *security assessments* are focused on finding vulnerabilities, often without regard to actually exploiting them and getting in.

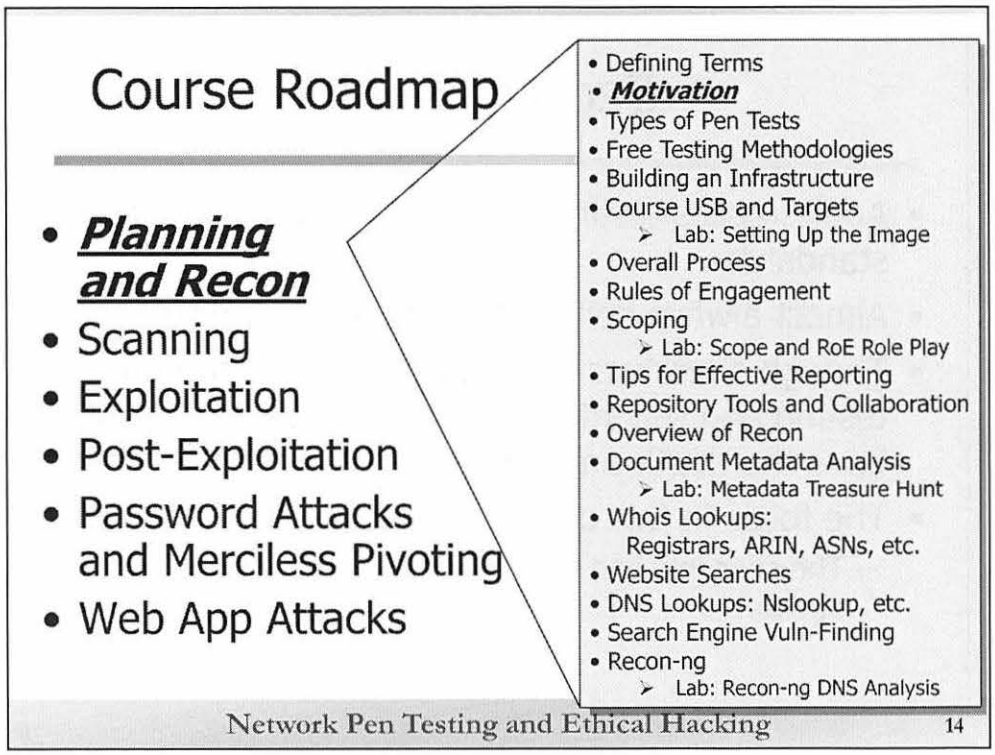
Thus, penetration testing often goes deeper, with its goal of taking over systems and stealing data, whereas security and vulnerability assessments are broader, involving the process of looking for security flaws. These assessments also often include policy and procedure review, which are usually not included in penetration testing.



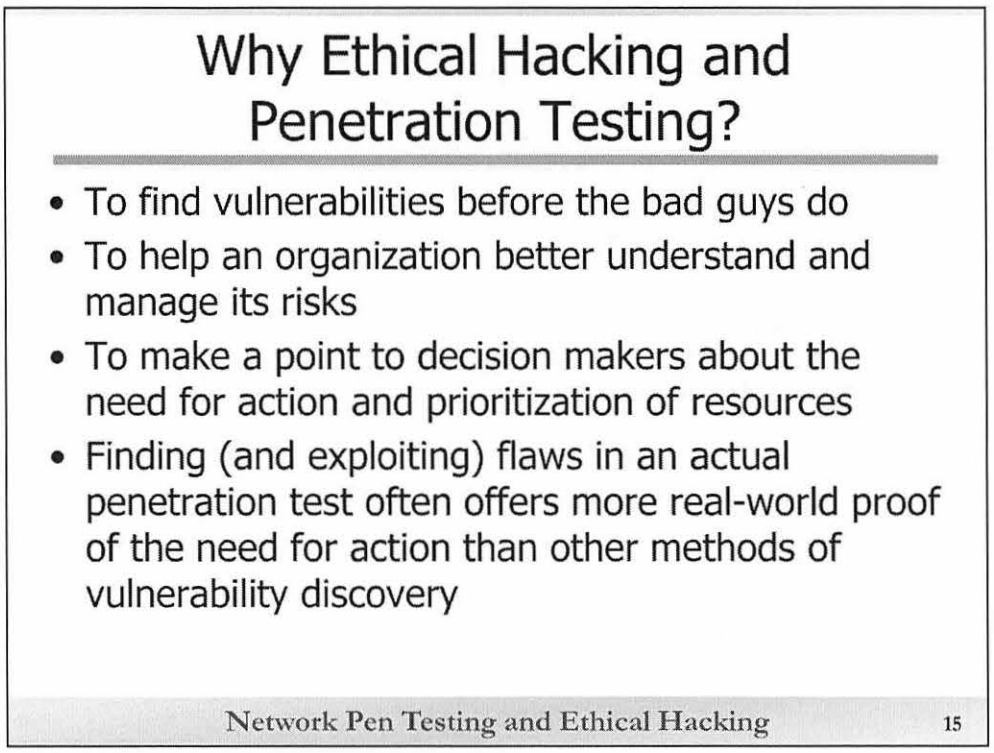
Finally, we have the phrase *security audit*. *Audit* implies that we are measuring things against a fixed, predetermined, rigorous set of standards. These audits are almost always done with detailed checklists.

Some penetration testing and ethical hacking organizations have created their own internal checklists of items that need to be covered in a test, but these checklists aren't as detailed as a comprehensive audit.

Our focus in this class is not on auditing. SANS has numerous other classes that address security audits in detail. Our focus is on ethical hacking and penetration testing.

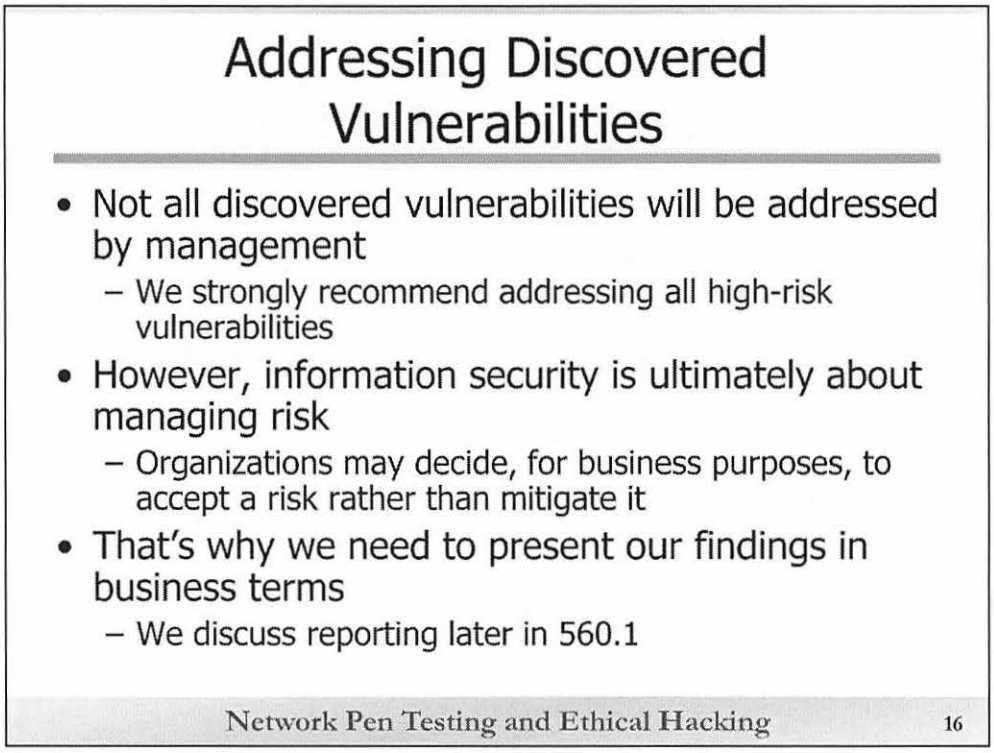


With terminology covered, let's now turn to the motivation for ethical hacking and penetration testing. What value do they provide to an organization? In short, why are we here?



Many organizations use ethical hacking and penetration testing to find security flaws before the bad guys do. After applying their security policies, procedures, and technology, organizations can use thorough penetration tests to see how effective their security actually is in light of an actual attack, albeit by friendly attackers.

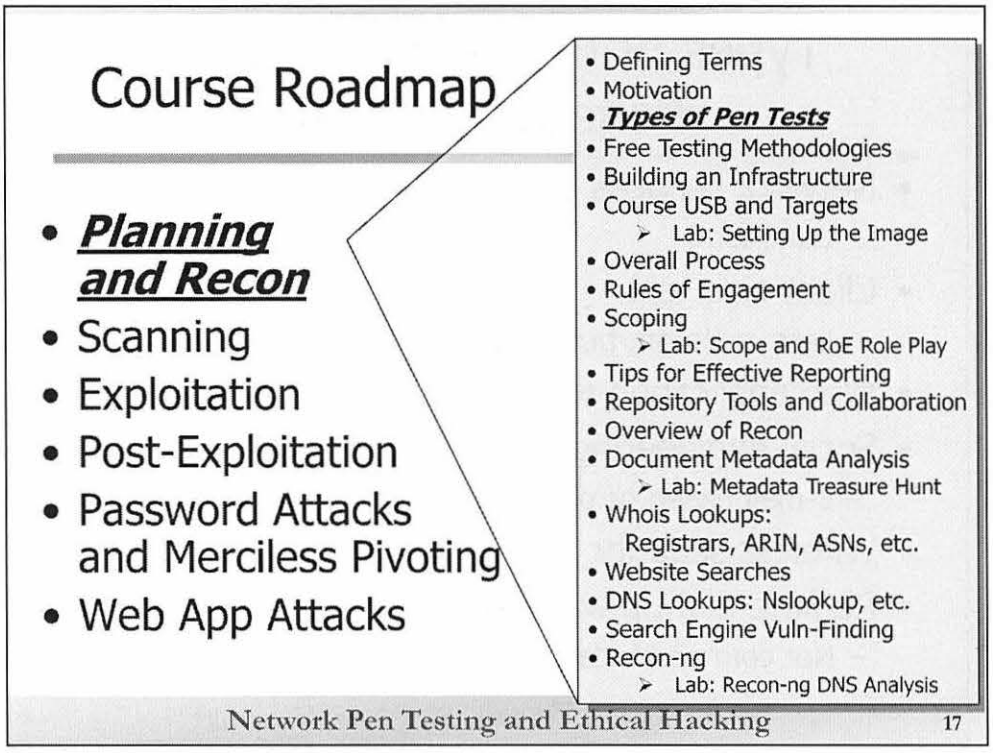
An added benefit of ethical hacking and penetration testing is that, because they show real vulnerabilities and indicate what a malicious attacker might be capable of achieving, they can get management's attention. Decision makers, when presented with the carefully formulated results of a test in business terms, are more likely to provide resources and attention to improve the security stance of an organization.



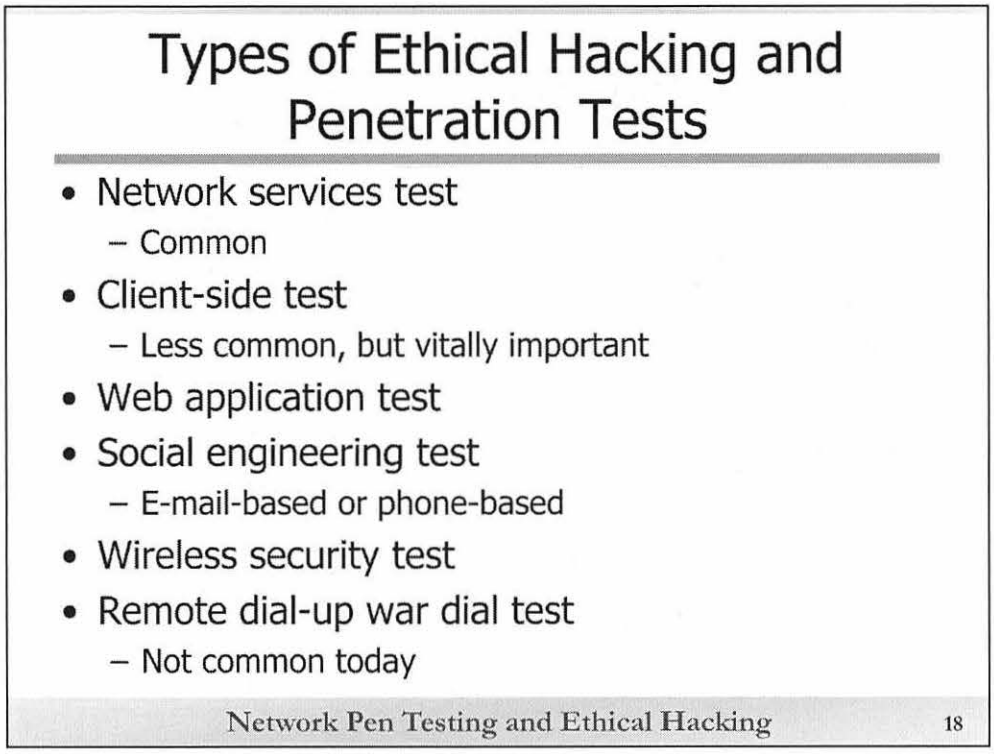
A major goal of penetration testing and ethical hacking is discovering flaws so that they can be remediated (by applying patches, reconfiguring systems, altering the architecture, changing processes, and more). However, it is important to note that in most tests, not all the discovered vulnerabilities are actually addressed.

We recommend that all high-risk vulnerabilities be addressed in a timely fashion, but the truth is that some vulnerabilities linger long after a test is complete, even high-risk issues. Remember, information security is all about managing risk, not eliminating it. Decision makers in an organization may conclude that, for business purposes, they will accept a given risk identified during a test, rather than mitigate the associated vulnerability. In the end, it's a business decision, informed by our input.

For this reason, we have to present our findings in both *business* and *technical* terms. That's an important principle to remember throughout this course. We'll get into reporting in more detail later.

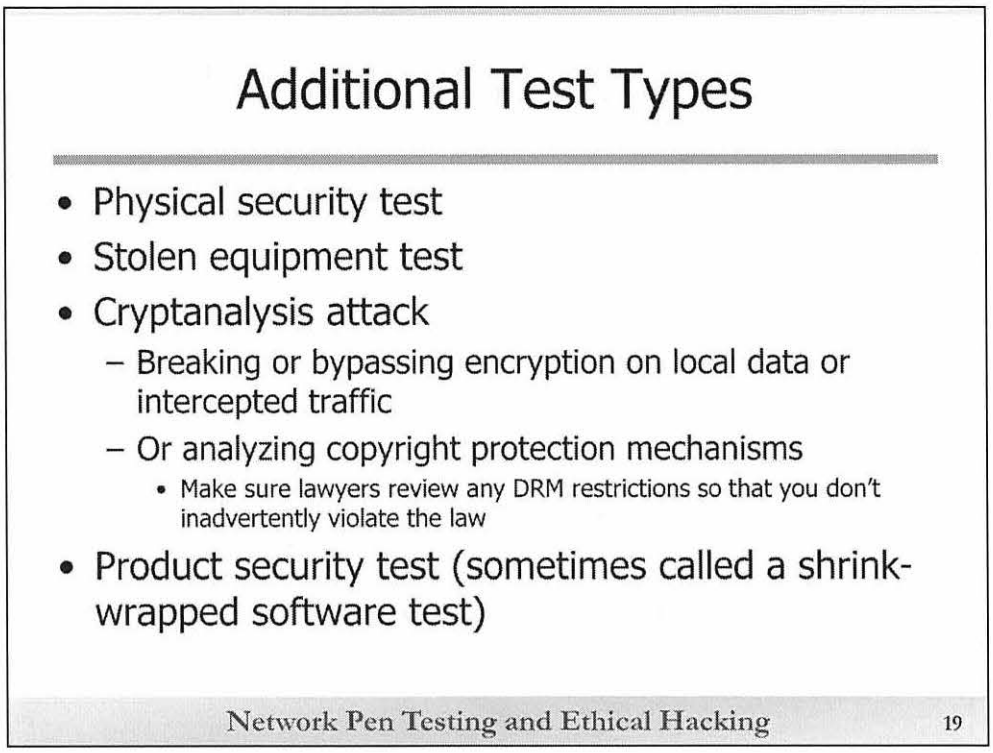


There is a large number of different types of ethical hacking and penetration tests. Let's now explore the different types, realizing that many of the tests we'll engage in for our jobs will be a mixture of a subset of these various types.



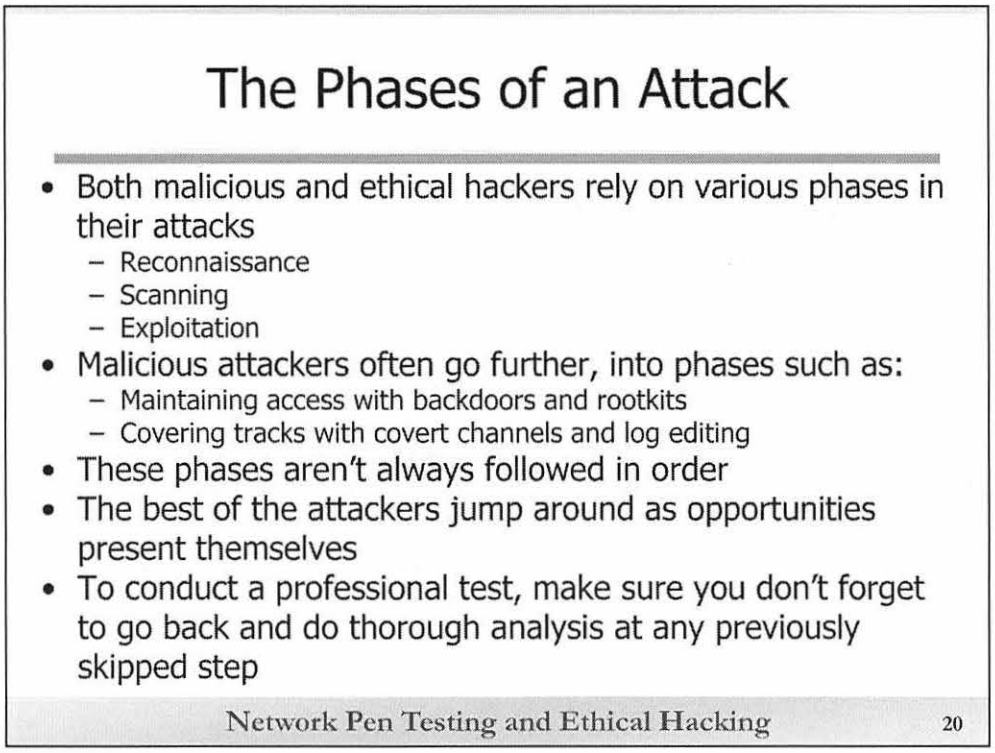
There are numerous kinds of ethical hacking and penetration tests:

- **Network services test**: This is one of the most common types of tests and involves finding target systems on the network, looking for openings in their underlying operating systems and available network services and then exploiting them remotely. Some of these network service tests happen remotely across the Internet, targeting the organization's perimeter networks. Others are launched locally, from the target's own facilities, to evaluate the security of their internal network or the DMZ from within, seeing what kinds of vulnerabilities an internal user could discover.
- **Client-side test**: This kind of test is designed to find vulnerabilities in and exploit client-side software, such as browsers, media players, document-editing programs, and so on.
- **Web application test**: These tests look for security vulnerabilities in the web-based applications deployed on the target environment.
- **Social engineering test**: This type of test involves attempting to dupe a user into revealing sensitive information such as a password, or possibly convincing a user to click a link in e-mail. These tests are often conducted via e-mail or over the phone, targeting selected help desks or users and evaluating processes, procedures, and user awareness.
- **Wireless security test**: These tests involve exploring a target's physical environment to find unauthorized wireless access points or authorized wireless access points with security weaknesses.
- **Remote dial-up war dial**: These tests look for modems in a target environment and often involve password guessing to log in to systems connected to discovered modems. Currently, they are not common.



Some additional types of ethical hacking and penetration tests include

- **Physical security test**: These tests look for flaws in the physical security practices of a target organization. Testers might attempt to gain access to buildings and rooms, or to take laptops, desktops, or recycling bins out of target facilities. Dumpster diving tests are a variation of a physical security analysis. Physical testing must be conducted carefully to ensure that the testers do not get hurt or arrested during their work.
- **Stolen equipment test**: This kind of test involves obtaining a piece of equipment from the target, such as a laptop computer, and then trying to extract sensitive information from it in a laboratory environment.
- **Cryptanalysis attack**: This test focuses on bypassing or breaking the encryption of data stored on a local system or across the network. Some of these tests also evaluate the strength of digital rights management (DRM) solutions. Due to legal restrictions regarding reverse engineering copyright protections (such as those imposed by the Digital Millennium Copyrights Act in the United States), any contract regarding the analysis of DRM software should be inspected by a lawyer to ensure that proper permission has been derived from the owners of the given DRM solution.
- **Product security test (sometimes called shrink-wrapped software test)**: In this kind of test, you look for security flaws in software products that can be installed in the tester's laboratory systems. Such tests look for flaws in the software, such as exploitable buffer overflow conditions, privilege escalation flaws, and exposure of unencrypted sensitive data.



Both malicious attackers and professional penetration testers/ethical hackers apply various phases in their attacks. Attacks are often separated into these phases:

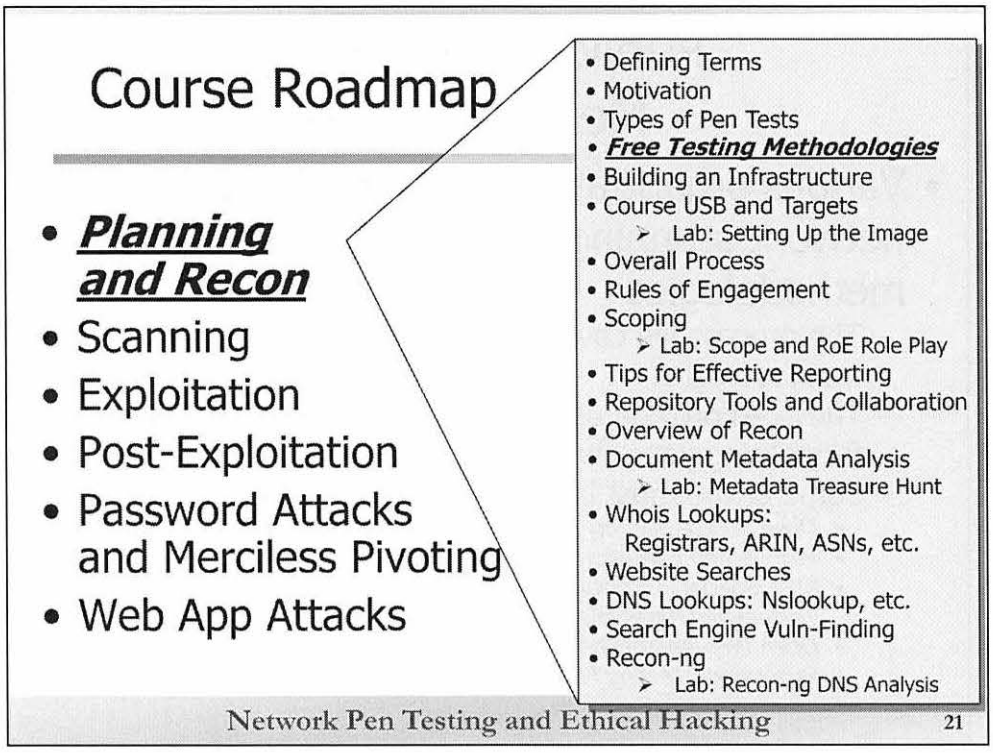
- *Reconnaissance* is the process of investigating the target organization to gather information about it from publicly available sources, such as domain registration services, websites, and so on. Some people include techniques such as social engineering and dumpster diving in the recon phase.
- *Scanning* is the process of finding openings in the target organization, such as Internet gateways, available systems, listening ports, and vulnerability lists.

In the *Exploitation* phase, attackers exploit target systems to compromise them, possibly getting control of them or causing a denial of service attack.

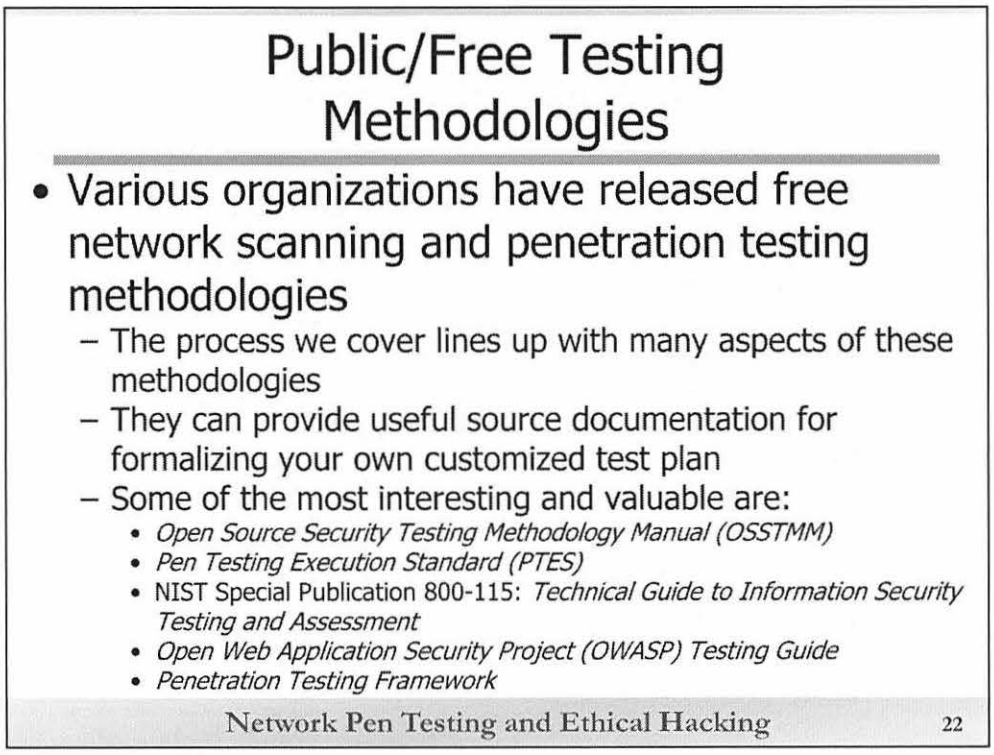
Although legitimate tests often include the previously listed phases, malicious attackers often go further than the Rules of Engagement allow for a professional penetration test. The next phase, often used by malicious attacker to maintain access and control of a target machine, involves setting up the compromised machine so that the attacker can keep control over it, with techniques such as installing backdoors and planting rootkits. Malicious attackers also often use a final phase, *Covering the Tracks*, in which they employ log editing, file hiding, and covert channels to hide their activities on a system.

Please note that the best of the attackers (both the good guys and the evil ones) are pragmatists. They don't always proceed from reconnaissance to scanning to gaining access and so on. Sure, they use these steps, but they are likely to jump around between them as events and discoveries warrant. For example, during the recon phase, attackers may discover an exploitable flaw that they will use to gain access directly, temporarily bypassing scanning. Then, after they gain access to one machine, they may go back and start scanning.

From a professional testing perspective, though, be careful when jumping out of order between these steps, making sure that you return to the earlier phases to conduct a comprehensive test.



Several organizations and individuals have released free ethical hacking and penetration test methodologies. You need to understand the freely available methodologies for several reasons. First, you need to recognize the people who invested their hard work in creating these methodologies and then provided them on a freely available basis to everyone. Secondly, these methodologies track nicely with the various topics covered in this course, so reviewing them can help to shore up the topics of this class, often from a slightly different perspective. And, thirdly, as you put together your own penetration testing process, you can utilize concepts and techniques from these documents as well as this course.



Several organizations have released high-quality, free penetration testing and ethical hacking methodology documents. The process we cover in this course addresses many topics also covered in these methodologies.

We recommend that you review each of these free documents because they provide useful insights into testing from various different perspectives. Also, when formulating your customized testing methodology, these document, together with this course, can act as useful sources to pull together wording for your documentation on your testing processes and findings.

Five of the best free documents on testing methodologies include:

- *Open Source Security Testing Methodology Manual (OSSTMM)*
- *Pen Testing Execution Standard (PTES)*
- NIST Special Publication 800-115: *Technical Guide to Information Security Testing and Assessment*
- *Open Web Application Security Project (OWASP) Testing Guide*
- *Penetration Testing Framework*

Let's briefly explore each one in more detail.

The *Open Source Security Testing Methodology Manual (OSSTMM)* was released by Pete Herzog and is distributed by the Institute for Security and Open Methodologies (ISECOM). This free document is focused on improving the transparency of enterprise security as well as the methodology of testers. Rather than making security testing a black art of mystery, this comprehensive document eloquently strives for repeatability, consistency, and high quality in numerous kinds of security tests. The document is written so that organizations and the testers they employ get the maximum business value for their activities. Earlier versions of the *OSSTMM* are available for free. The latest version and drafts of new updates are available to Silver subscribing members. Gold subscribers get those items as well, plus additional research, mailing lists, and contacts for business questions. Platinum subscribers get its logo and link on the ISECOM website, plus access to an exclusive mailing list.

The overall document is broad, covering numerous kinds of security tests. It does not go into depth with particular commands and tools but is still immensely useful.

Topics addressed in the *OSSTMM* include scoping, metrics, human security testing, physical security testing, wireless security testing, telecomm security testing, and data networks security testing.

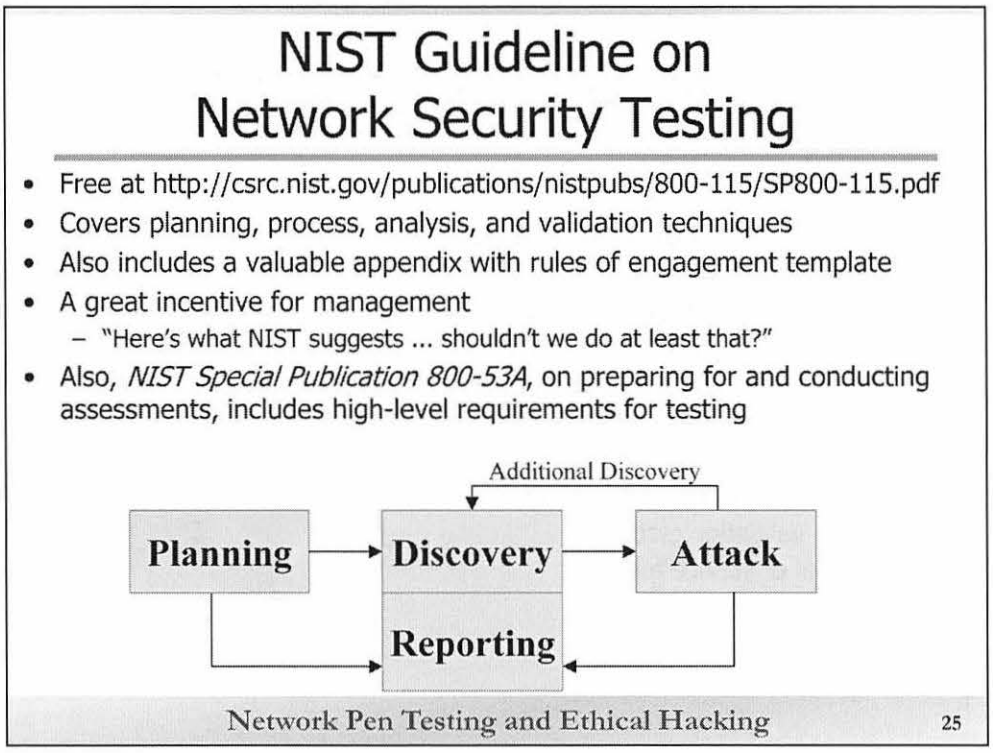
One of the best aspects of the *OSSTMM* is its detailed discussion of scoping a project in advance, as well the report templates that it includes. It has fill-in-the-blank templates for almost every kind of test it describes.

Penetration Testing Execution Standard (PTES)

- Available for free at www.pentest-standard.org
- Many contributors, spearheaded by Chris Nickerson of Lares Consulting
- Aims to create a standard so that organizations can understand what is involved in conducting a penetration test
- Includes information about:
  - Pre-engagement interactions (scoping and rules of engagement)
  - Intelligence gathering (recon)
  - Threat modeling
  - Vulnerability analysis
  - Exploitation and post exploitation
  - Reporting
- Currently, a great outline of an in-depth penetration test

The *Penetration Test Execution Standard (PTES)* is an interesting project that aims to improve the state of penetration testing projects by defining a set of activities that should be included so that a project can reasonably be called a penetration test. Chris Nickerson of Lares Consulting began the project and has involved numerous collaborators. The aim is to create a standard for penetration testing so that organizations who procure tests can be assured that they are receiving a test conducted according to an understandable, repeatable framework that provides them with business value.

The PTES includes information about pre-engagement interactions, including scoping and rules of engagement that we'll discuss in more detail in 560.1, intelligence gathering (reconnaissance activities), threat modeling, vulnerability analysis, exploitation and post-exploitation activities, and in-depth reporting. It's a great outline of a penetration test, with more detail being added on a regular basis.

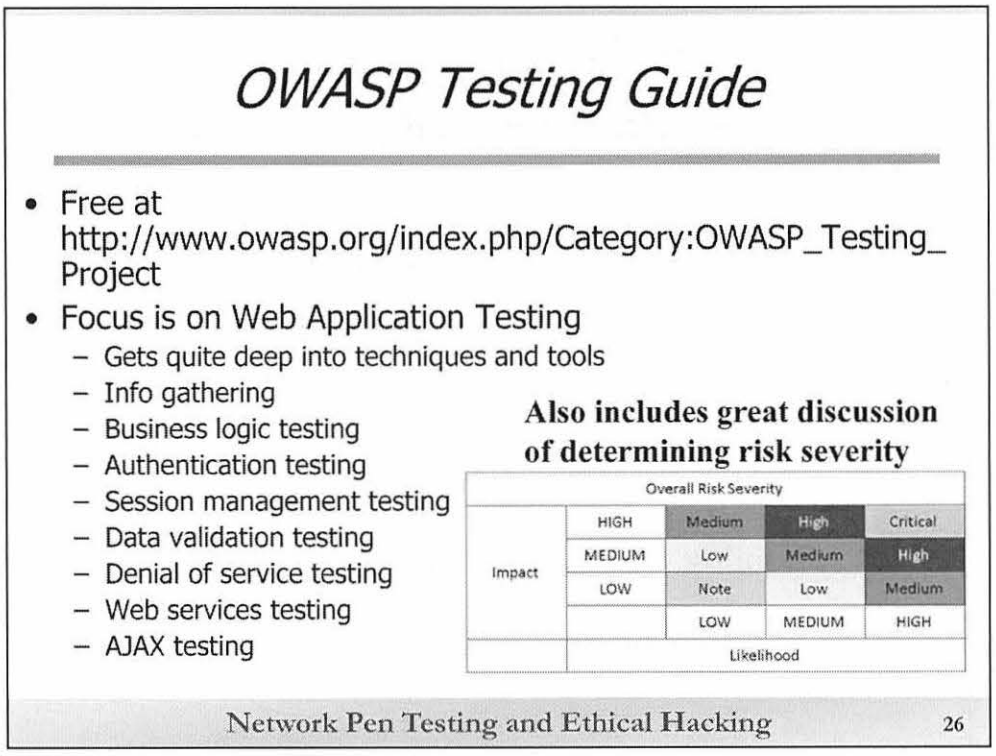


The United States National Institute of Standards and Technology (NIST) has released a document called *Technical Guide to Information Security Testing and Assessment* that covers network penetration testing methodologies at a high level. The document addresses the process commonly applied in testing, planning for tests, conducting detailed analysis, and dealing with validation of discovered issues. It also includes appendixes that cover some common tools used in vulnerability assessments and penetration tests.

Another useful appendix is a template for Rules of Engagement, helping testers and target system personnel agree upon various vital aspects of how the testing will be conducted. We'll spend some time later in this book discussing Rules of Engagement.

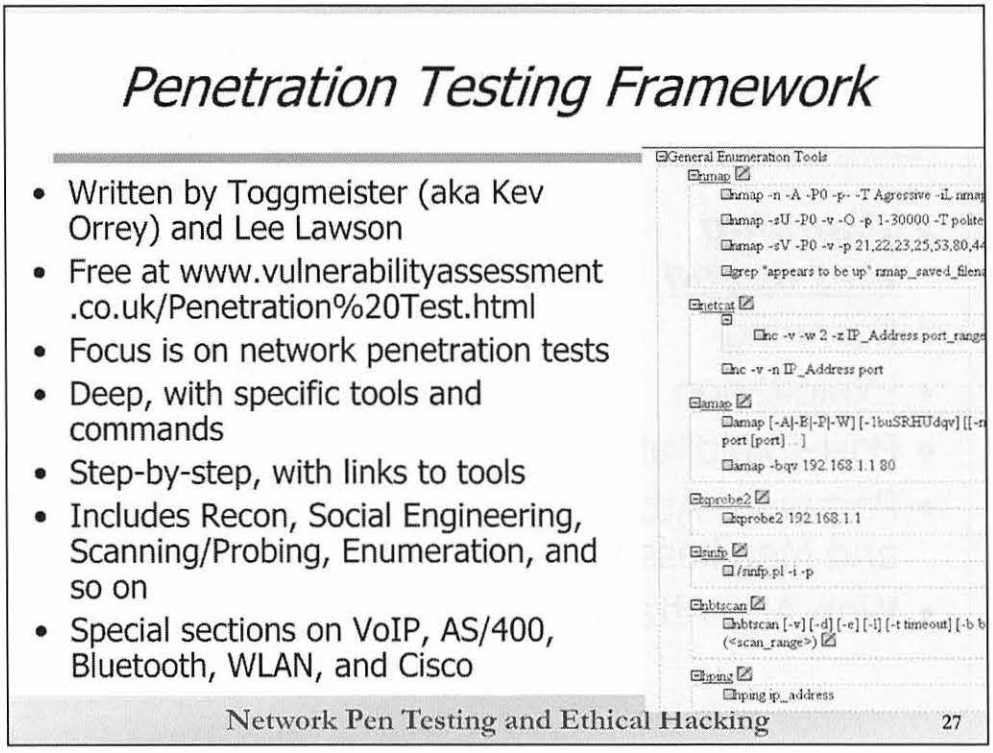
One of the most useful aspects of the *NIST Guide* is the motivation it can help us inspire in management. If management suggests that our testing methodology shouldn't include some vital component that NIST recommends, we can ask our management why they want to deviate from NIST's guidance. Management may then provide business rationale for doing so or decide that complying with the NIST document is a better practice than it originally anticipated. Either way, we get a better test more attuned to the business needs of our enterprises.

Another document from NIST also addresses measuring security in an organization. The *Guide for Assessing the Security Controls in Federal Information Systems,* Special Publication 800-53A, is more high-level than SP 800-115 but still provides some useful tips for planning assessments.



Next, is the *Open Web Application Security Project (OWASP) Testing Guide*. Unlike the other broad and general purpose methodologies we've touched on so far, the *OWASP Guide* focuses purely on web application security testing. From a web app perspective, this document is an excellent description of the various kinds of testing that need to be done, providing great depth and a wide variety of tools to use in the process.

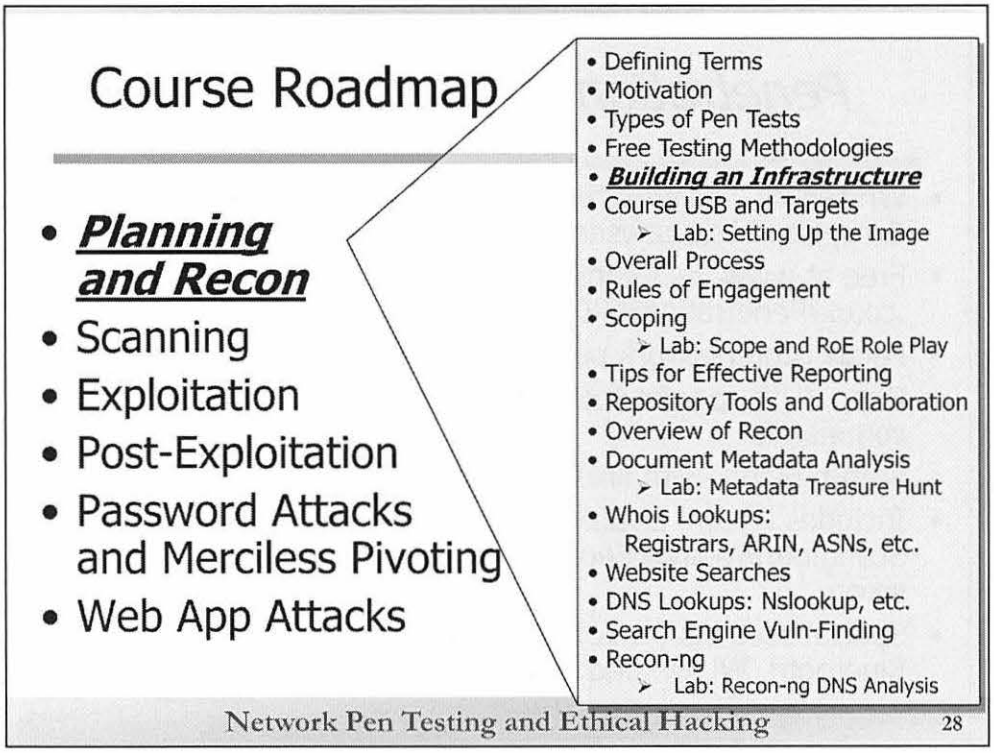
One of the best aspects of the *OWASP Guide* is its detailed description of determining the business risk posed by findings. The *OWASP Guide* rates risk based on the impact it could have to the business and the likelihood it has of occurring. From those two aspects, the overall risk rating of a given finding is derived, giving the enterprise appropriate guidance on prioritizing their findings.



And, finally, we have the deepest of the free testing methodologies that we cover: *The Penetration Testing Framework* by Toggmeister and Lee Lawson. This website provides a step-by-step walk through of every aspect of a network penetration test, including specific tools (with links to each and every tool) and the individual commands to use for each tool.

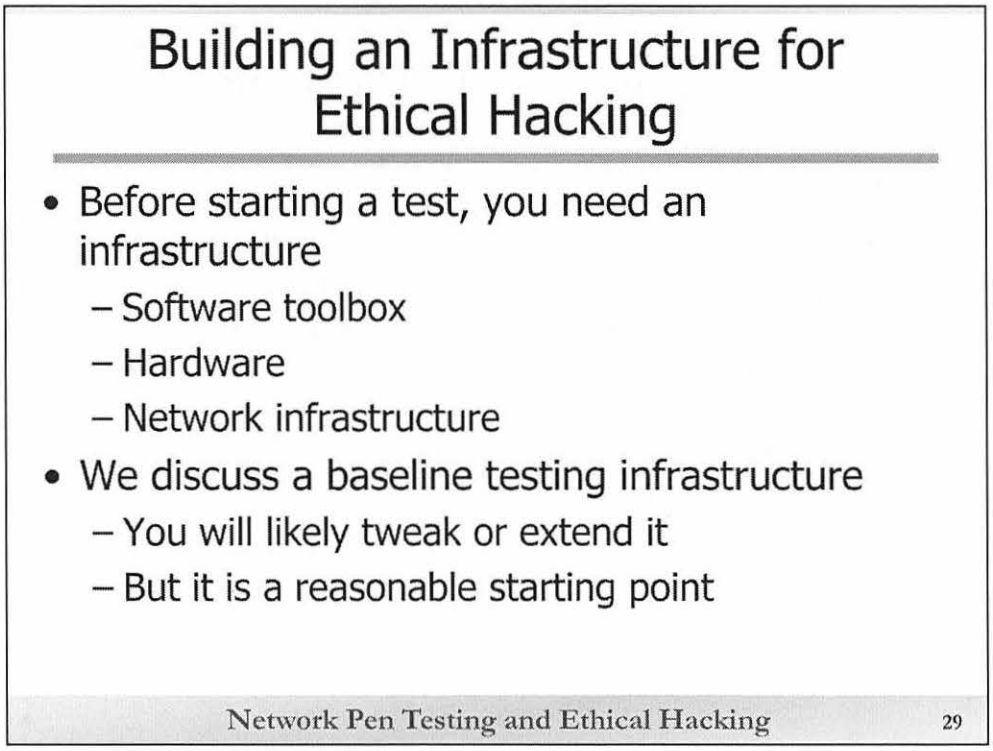
The document walks its reader through several concepts, step by step, covering reconnaissance, social engineering (via e-mail and the phone), scanning, enumeration of target systems, exploitation, configuration review, and more.

Several sections focus on specific technology, such as Voice over IP (VoIP), assessing the security of AS/400 machines, Bluetooth security analyses, and wireless LAN assessments. The section on analyzing Cisco routers and related devices is also quite helpful.

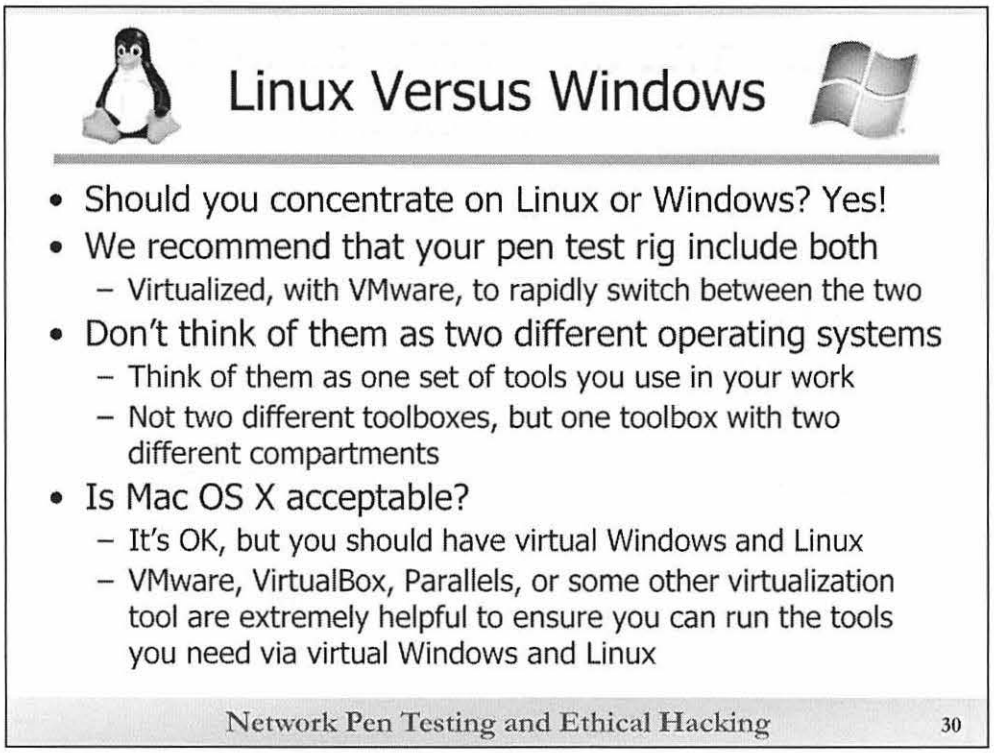


A well-stocked lab and an arsenal of testing tools are crucial to the success of an ethical hacker and penetration tester. Let's now discuss the hardware, software, and network connectivity used by testers in their work.

Keep in mind that these infrastructure items we discuss are not a one-size-fits-all proposition. Instead, we cover the areas of tools that you need, with some notable examples. Then, based on your budget, expertise, and test types, you can construct an appropriate arsenal to match your test regimen.



To conduct a thorough test, ethical hackers and penetration testers first have to establish an infrastructure from which to do their work. Detailed planning in advance is essential, pulling together the proper software, hardware, and network infrastructure. We discuss some tips for doing this properly, but keep in mind that the technical infrastructure necessary for performing these tests is not one-size-fits-all. Consider the items we discuss to be a baseline infrastructure, which you can tweak or extend to meet your own specific testing needs. We discuss a software toolbox, the hardware, and a network infrastructure you should consider when doing tests.



A common question among penetration testers and ethical hackers is, "Should I focus my skills and toolbox on Linux or Windows?" When confronted with this question, we recommend that your pen test toolset include both operating systems, side-by-side, working together to maximize your efficiency and capabilities. The truth is, some tools work better on Linux whereas others work better on Windows. Some tools work just fine on both, whereas other tools have been released only for one of those platforms. Thus, if you choose to work in only one OS, or at least just focus on that OS, you'll be missing out on a lot of useful tools and techniques. To improve productivity and streamline workflow, we recommend virtualizing one of these two OSs, perhaps using VMware, and running the two simultaneously on the same hardware so that you can quickly switch between them.

The entire question posed at the start of the preceding paragraph illustrates a mindset that should be transcended. Don't think of them as two different operating systems. Think of them as one set of tools that you use in your penetration testing and ethical hacking job. As a carpenter or plumber would use the best tool available and convenient for a given job, so should you. To continue with that analogy, don't think of Windows and Linux as two different toolboxes. Instead, they are two different compartments in your single toolbox.

Some of you are no doubt wondering whether Mac OS X is an acceptable platform for penetration testing and ethical hacking. It is, with remarkable stability and ease of use. However, there are some tools for Linux and Windows that will simply not run on Mac OS X, no matter how hard you try to get them installed. Thus, if you plan to use Mac OS X, make sure you get a virtualization solution for it (such as VMware Fusion or Parallels) so that you can also run both Windows and Linux on top of Mac OS X.

First, you need software for your testing regimen. With this course, you received a copy of the SANS Slingshot image, which is full of tools used in ethical hacking and penetration testing. Furthermore, this VMware image includes tools pre-installed, and in many cases, preconfigured so that you can apply them directly in your own testing.

Another useful source of tools are the bootable Linux distributions various people have made freely available, loaded with useful assessment and attack tools. A solid set of tools is included in Kali Linux, created and maintained by Offensive Security. Numerous similar Linux images for pen testing are also available, but Kali is one of the best because of its comprehensive set of tools, compatibility with a wide range of hardware, and carefully designed organization and layout.

In addition to the SANS Slingshot image and other free bootable Linux environments, a variety of websites offer vast arsenals of free tools and exploits, which can be incredibly helpful. The vast majority of professional penetration testers and ethical hackers rely on at least some of these free tools when doing their jobs. Before considering whether you can run such tools in your environment, you need to determine your organizations' policy regarding the use of such third-party security assessment and exploitation tools. Some organizations strictly forbid the running of any tools beyond a standard baseline of already-approved tools. Others allow additional tools to be used, but only if they are carefully vetted.

Consider this scenario: A tester scans a target environment, discovering a listening service that has a version number that is known to be exploitable. With a little research, the tester discovers a freely downloadable exploit for that specific version of the service from an exploit distribution site. Suppose further that the rules of engagement for the test allow actual exploitation of the target machine, and, furthermore, the tester's own organization allows for the use of third-party, free exploits. What should the tester do?

We strongly urge you to be careful with free, downloaded tools and exploit code. Historically, some of the tools and exploit code freely distributed on the Internet have included backdoors that let a bad person control any system on that the tool was run, or even control the target machine the tool was run against. Also, some of the tools may cause a crash in a target service or system.

Thus, we recommend that testers analyze all free tools carefully in a laboratory before using them in a test. If you have the skills, review the source code for the tool before using it, making sure it does exactly what it says it does, with no hidden backdoor functionality or other trojan horse capabilities in the tool. If you cannot review the code, then, at a minimum, run the free tool in a laboratory environment, carefully reviewing the traffic it sends across the network (looking for unexpected packets going to unexpected destinations) and any changes it makes in the file system of the attack and target machines. The free Microsoft Sysinternals processmon tool is helpful in analyzing file system and Registry interactions. Processmon has subsumed the earlier tools, filemon and regmon, extending their functionality in a single tool.

Although there are numerous exploit and attack tool repositories on the Internet, some of the most comprehensive archives that are updated on a regular basis include The Exploit Database and Packetstorm Security. Several other sites come and go on a regular basis, but these sites are long-standing and tend to have relatively higher quality tools.

The Exploit Database (exploit-db for short) is maintained by the same group that maintains the Kali Linux distribution, Offensive Security. Its site hosts more than 10,000 exploits and sorts them into useful categories such as Remote Exploits, Local Exploits, Web Applications, Denial of Service/Proof of Concept, Shellcode, and papers. For each exploit in these categories, they list the platform (Windows, Linux, PHP, and such) and the author.

The Security Focus BID website also has information about various vulnerabilities along with exploits for some of them, available at www.securityfocus.com/bid. The older Security Focus news site was shut down, but the useful BID search is still available.

Also, the SEBUG site has hundreds of categories of vulnerabilities, including exploit code for many different issues that they inventory.

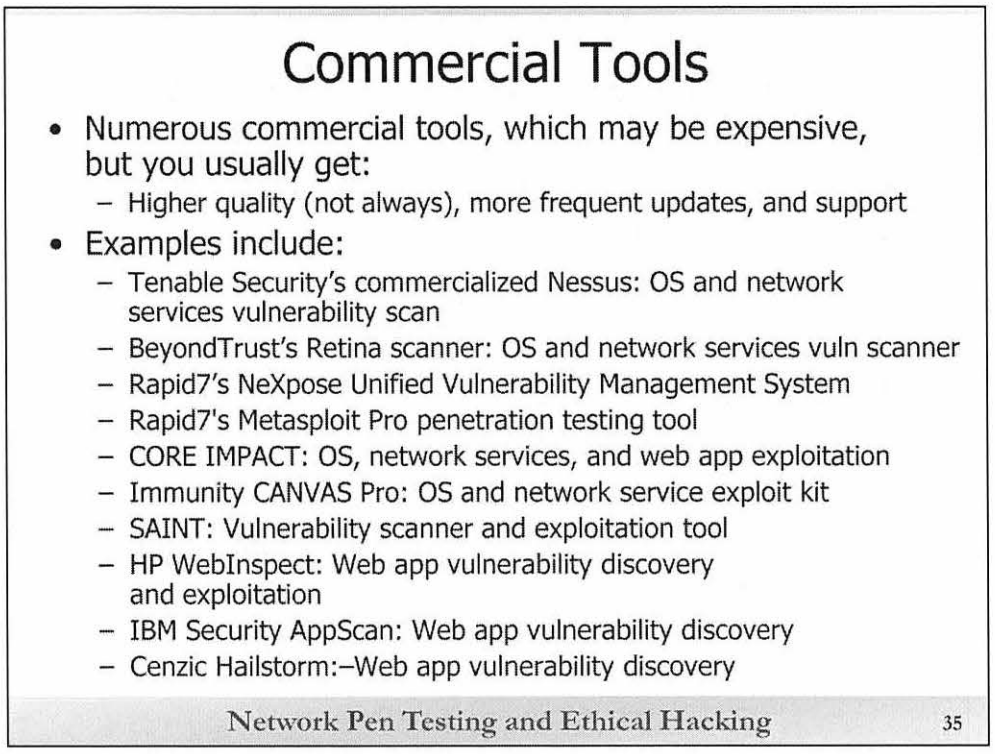
Packetstorm Security has an archive of attack and defense tools that spans over a decade. It's quite an impressive assortment of useful tools, exploits, and security research papers.

Note that we are not endorsing these sites or the tools that they distribute. These sites have been quite controversial, and you need to be careful with any code you download from them. Still, ethical hackers and penetration testers need to know about these sites to do their jobs.

Beyond the tool and exploit sites, numerous vulnerability research sites are also available. Although these sites do not distribute exploit code freely, they do publish information about vulnerabilities. These detailed vulnerability descriptions are invaluable in letting a tester know that there is an issue with a system type or service version discovered in a test. Even though an exploit might not be available (in fact, an exploit may have never been publicly released or even created), the tester still needs to understand the vulnerabilities so that they can be included in the test report.

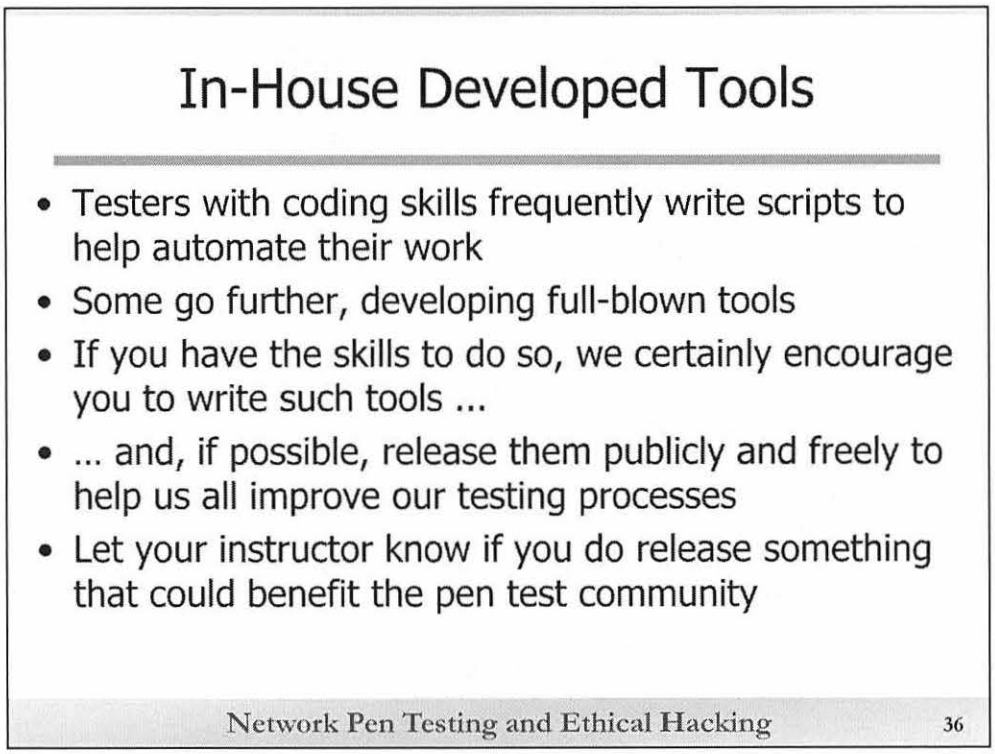
Some of the best sites with vulnerability research and detailed descriptions are the following sites:

- The United States Computer Emergency Readiness Team (US-CERT), maintained by the U.S. Department of Homeland Security (DHS)
- The Common Vulnerabilities and Exposures (CVE) repository operated by Mitre
- The Secunia vulnerability list, a solid list of issues provided by a vendor of software security solutions
- The Hackerstorm website, which includes a free, downloadable Open Source Vulnerability Database tool that can be stored locally by a tester for searching even without Internet access
- ExploitHub sells exploits for nonzero-day vulnerabilities (that is, for vulnerabilities that are already publicly disclosed). They act as a clearinghouse for authors to sell their exploits for such flaws.



In addition to the free tools we've been discussing, some penetration testers and ethical hackers rely on commercial tools for testing. There are a large number of commercial tools, with new ones released on a regular basis. The advantages of commercial tools include generally higher quality (but not always), typically more frequent updates (given the vendors' paid teams of software developers), and technical support if issues arise during testing.

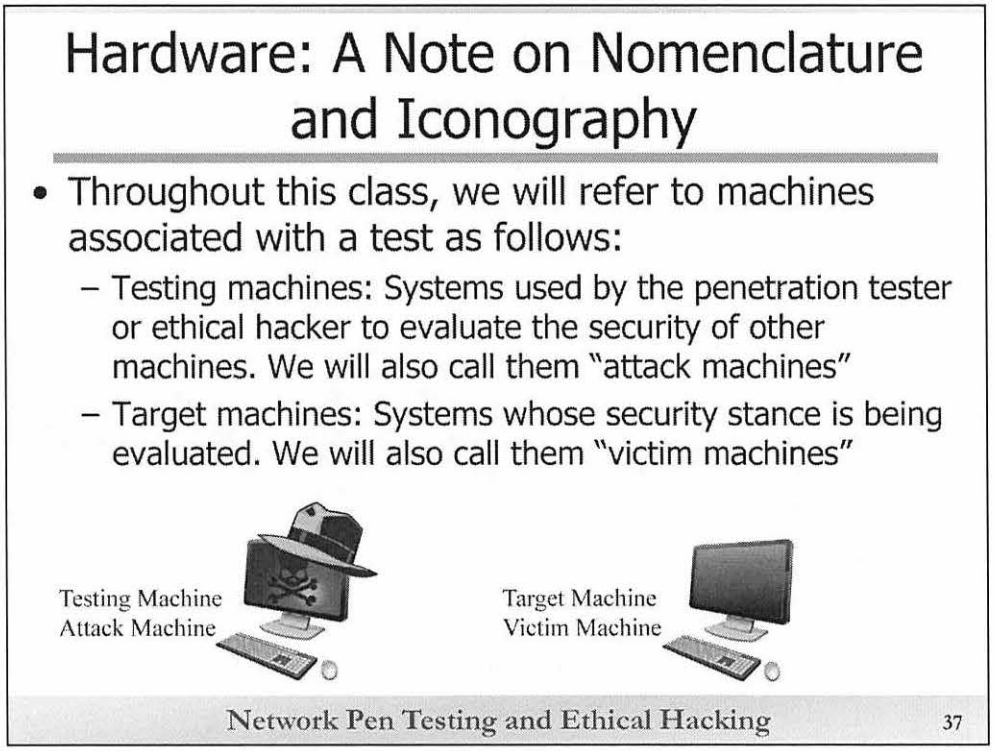
Although this course is taught from a vendor-neutral perspective, professional penetration testers and ethical hackers do need to know about some of the commercially available tools, even if they don't use them. That way, they can make sure that their test regimen made up of noncommercial (free or in-house) tools includes similar concepts and capabilities of the commercial tools. This slide lists a few of the more popular and comprehensive tools for testing.



Some penetration testers and ethical hackers with coding skills develop scripts that automate some portions of their test to help improve efficiencies. Some go even further, writing full-blown tools of their own that improve upon publicly available tools or conduct specific tests tuned to their target environment.

If you have the development skills to do so, we encourage you to create custom scripts and tools to help in your testing. If you do create a high-quality tool that would benefit others, we encourage you to release it publicly.

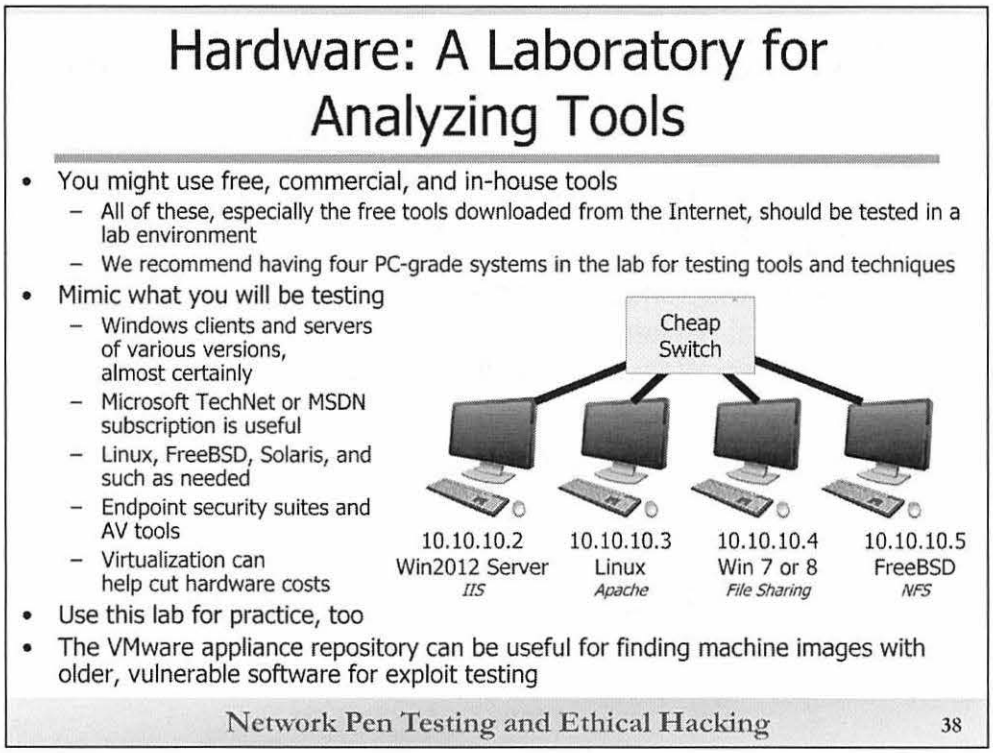
If you do release a tool, let your instructor for this class know about it. It could help us improve our testing processes and may be added to a future version of this course.



For the remainder of this class, we need to carefully differentiate between the machines used by the penetration tester or ethical hacker and the machines whose security is being evaluated.

We use the terminology *testing machines* and *attack machines* to refer to the systems that the tester uses to evaluate the security of other systems. These testing machines often run the attacker's scanning and exploitation tools. In figures, these machines will be represented with a red screen and black hat. We use these pictorial clues to help you rapidly identify where the attacker's machines are in a diagram. However, please do not think that the black hat on this computer implies that the attacker is somehow evil. The attackers we refer to here are professional ethical hackers and penetration testers. The black hat just makes this system easier to quickly locate in the figure.

The machines whose security is being evaluated are referred to as *target machines*, and, occasionally, as *victim machines*. They are represented pictorially as a standard machine, with no hat.



As we've seen, your tests might rely on free, commercial, and/or in-house tools, depending on the policies of your organization. Whichever tools you use, you should test them in a laboratory environment to make sure you understand how they work and their potential impact on a target machine. Such laboratory testing and analysis is especially vital for free tools downloaded from the Internet because of concerns about quality, the potential to crash a target, and hidden functionality that could compromise the test systems.

In creating a test lab, we recommend that you get at least four inexpensive PC-class systems. That way, you can have a variety of different kinds of operating systems and software applications to analyze. Because the speed of these machines isn't vitally important, inexpensive 2- or 3-year-old PCs can be used. If you want a more portable environment and can afford higher performance computers, you also may want to consider using virtual machines running in VMware or other virtualization products for your test-bed.

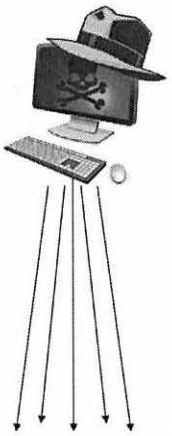
Your test-bed should include operating systems that mimic what you will encounter in your actual tests, likely at least a variety of Windows clients and servers. A Microsoft Technet or MSDN subscription comes in handy because it provides access to many Microsoft products for laboratory testing purposes (not production use). In addition, include Linux, FreeBSD, and other operating systems that make sense for what you will test. You should also have access to the most popular end point security suites and antivirus tools (especially those used in your target environments) to evaluate how your attacks will function against them and to tweak your approach in light of the target's defenses.

A well-equipped lab will also be useful for practice hacking to help improve your skills over time.

Another useful source of vulnerable applications pre-installed on their own operating systems is VMware's virtual appliance repository, at www.vmware.com/appliances. They have a vast archive of virtual machines freely available for download (typically focused on freely redistributable operating systems, such as Linux and FreeBSD). Many of the older images include applications that are known to have vulnerabilities, which you can test and evaluate in the lab.

Hardware: Systems Used for Testing

- You also need one or more machines to use for testing
  - Ideally, use machines dedicated to the testing
    - Use systems that you do not use for day-to-day Internet surfing or reading e-mail
    - Systems without any sensitive information on them ... only temporary results as they are pulled together
    - These machines may have to stay online scanning for an extended period of time, tying them up
    - These testing machines will likely not be protected by firewalls
  - For extensive scans, you may want to set up a scanning server co-located with a fast Internet connection
    - Possibly accessed via SSH or terminal services

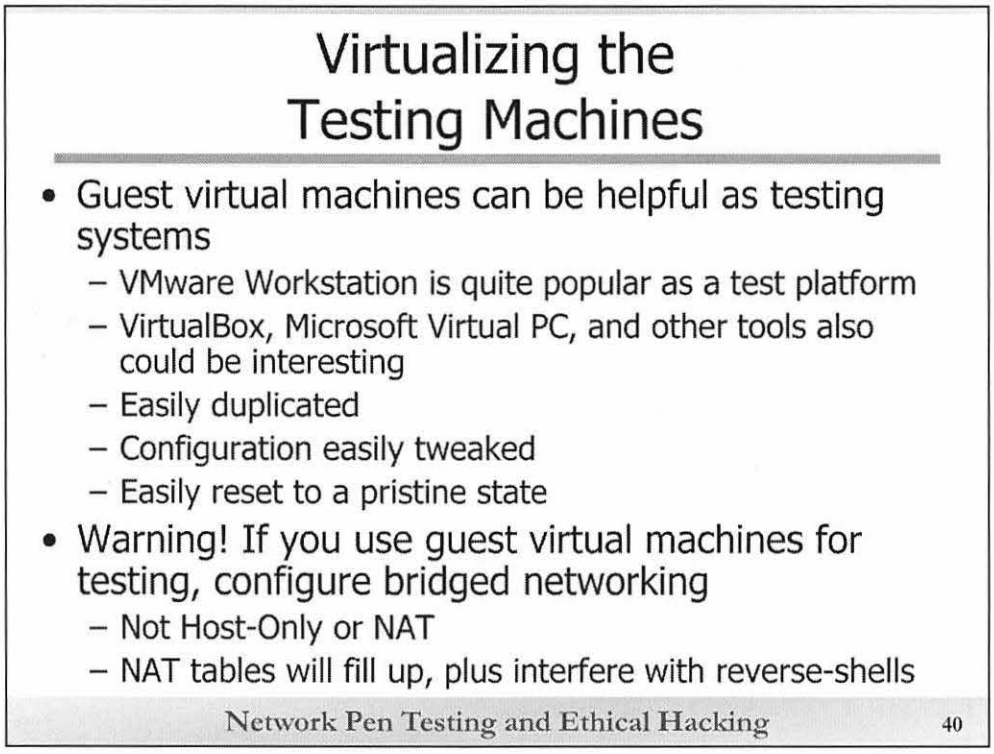


Network Pen Testing and Ethical Hacking 39

Next, you need the actual systems that will be doing the testing. We recommend that you use one or more machines that are dedicated solely to testing. That is, ideally, your testing machines should not be used for routine web surfing and e-mail reading, or even vulnerability research during a test. In addition, these machines used for testing should not be storing any sensitive information, other than the temporary tools output as the test is running. In other words, these machines should be focused exclusively on testing.

There are several reasons for this exclusivity. First, these machines may have to stay online for a significant period of time, as you wait for a lengthy scan to complete. Second, these machines used for testing are often not firewalled, either by a personal firewall or a network firewall. We address this firewall issue in more detail shortly. And, third, any additional software that you run on the testing machine could slow down the progress of your test or otherwise impact the results.

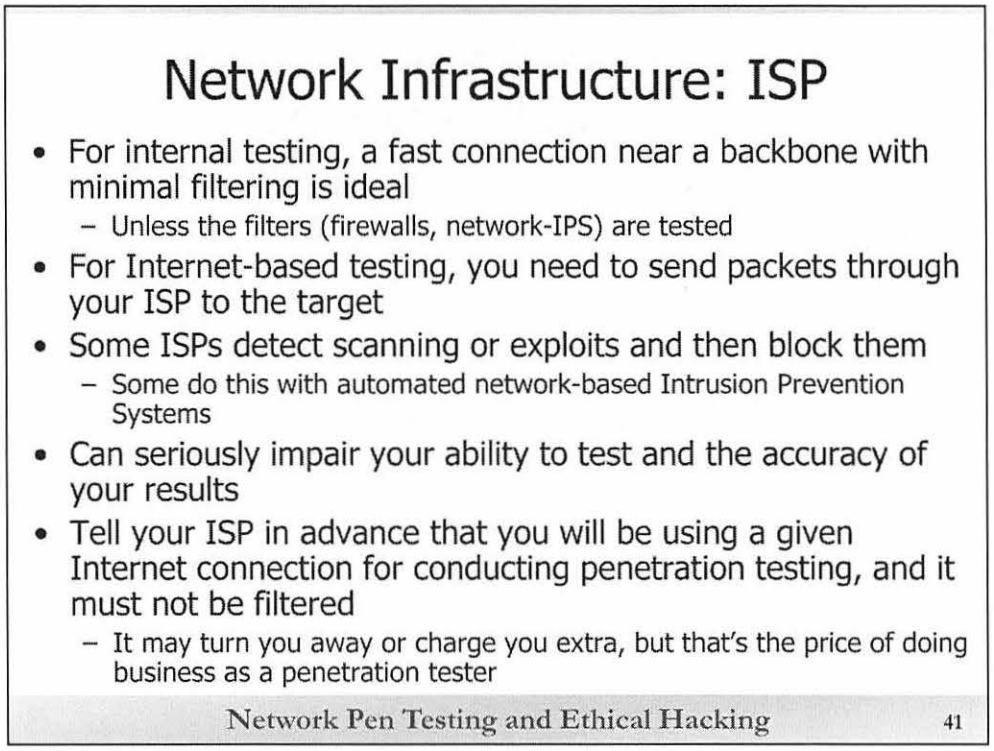
For extensive, long-term scans that might run for several days, you may want to consider buying a server-grade machine that is housed in a secure location, either at your own facilities or in a locked cage at a co-location facility with high bandwidth. You could then use Secure Shell or terminal services to control testing software on that machine that is dedicated to testing.



Many penetration testers and ethical hackers use guest virtual machines as their testing systems. Most rely on VMware Workstation because of its rich set of features and relatively low cost. Some use VirtualBox, a free virtualization platform. Others rely on Microsoft's virtualization products; although, they are less common in penetration testing than VMware's products. This difference is due to the early market lead VMware established with a lot of useful features. Microsoft has been playing catch-up in virtualization and may one day surpass VMware, but the jury is still out.

Virtual machines are useful for penetration testers for many reasons. First, they are easily duplicated. A tester can simply replicate the disk image of the VMware guest and have another identical testing guest machine ready to run. Also, the tester can easily modify the configuration of a guest machine, altering network settings, the amount of RAM, disk image sizes, and so forth. And, with VMware's Revert to Snapshot feature, a guest machine can be quickly reset to a pristine state, in case a tool causes problems with the machine.

There is one important note about using guest virtual machines for testing: Configure them to use Bridged Networking. Many VM environments offer other networking alternatives that will get in the way of your testing. In addition to Bridged Networking, VMware also supports Host-Only and NAT Networking. Bridged Networking makes the guest look like it is on the same subnet as the host machine, a desirable property. As a tester, we want our packets to get out from our testing system to the target with as little interruption and alteration as possible. Having the guest connected via a virtual bridge has little impact on the packets. NAT Networking, however, will perform Network Address Translation on the packets, altering them and potentially dropping them if the NAT table fills up. Finally, Host-Only Networking just doesn't make sense in an across-the-network test because the guest can only reach the host and no other systems. Thus, when using guest VMs as the testing systems, use Bridged Networking or you will miss things in your findings!

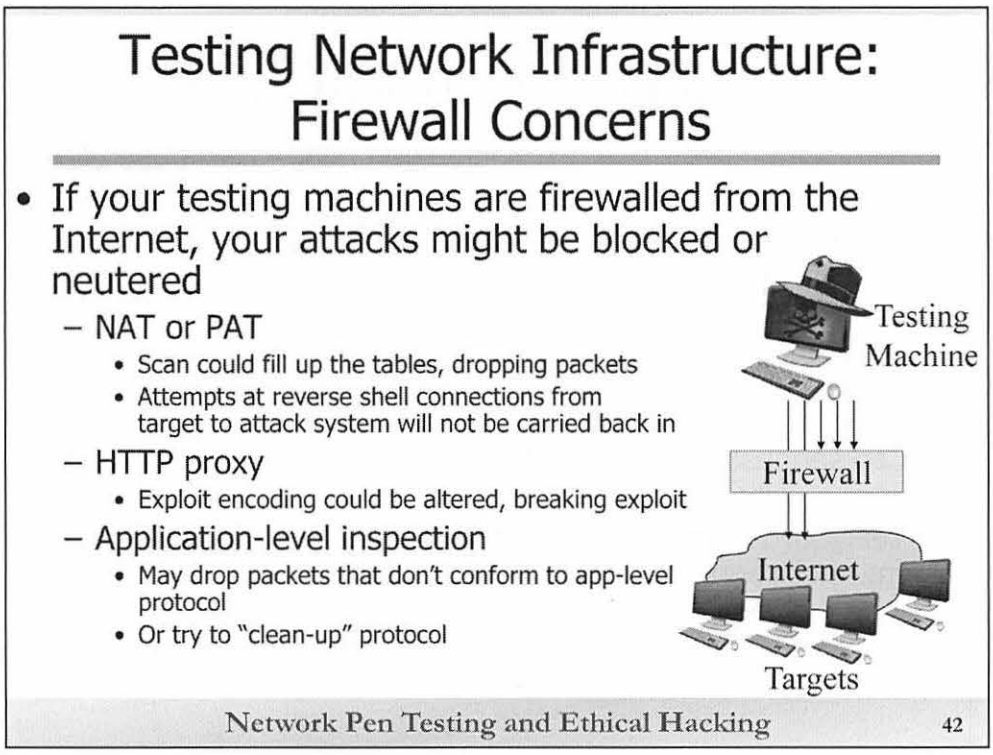


For internal testing, the testers should have a fast network connection near a network backbone with minimal filtering between it and the target systems, unless the filtering devices themselves, such as firewalls and network-based Intrusion Prevention Systems (IPSs), are tested.

For testing across the Internet, the testers will, of course, need an Internet connection from an ISP. We recommend that they get a stable, relatively high-speed connection, such as a T1 or better. Cable modems, DSL, and FIOS lines are also a possibility, but they are often less reliable than a T1. Regardless of the connection type, the testers will be sending some unusual traffic across that line. Their scanning will generate a large number of unusual packets. Furthermore, they may run actual exploits against targets across that ISP. This could be a problem, unless it is cleared with the ISP first.

Some ISPs detect network scans and throttle them, slowing down a test. Others block certain TCP and UDP ports for consumer-grade connections (like consumer cable modems, DSL lines, and FIOS) to help protect consumers on their networks. For example, connections to or from TCP port 25 are blocked by some ISPs to lower the chance that e-mail-relaying bots can be installed on their subscribers PCs. Such port blocking will prevent the tester from evaluating the security of services on those ports. Some ISPs even identify exploitation attempts and then block the exploit packets using network-based IPS tools. Such an action would prevent the tester from moving forward on that part of a test. Any of these issues would result in inaccurate results. Furthermore, performing scanning or exploitation through your ISPs network could be a violation of their terms of service. Violating those terms could result in service termination and a possible lawsuit.

Because of the concerns associated with using the ISP connection for testing, it is vital that you tell the ISP in advance that you plan on using the connection for penetration testing and security assessments. (You may want to avoid the phrase "ethical hacking" as it may misunderstand the term "hacking.") It may tell you to seek another ISP, but it is better to determine its policy in advance than to find it out in the middle of a test. It may charge you extra for the connection or force you to buy a different grade of service. Still, such expenditures are a reasonable price of doing business.



For the testing machines scanning across the Internet, you may want avoid personal and network firewall technologies. Firewalls may block inbound or outbound packets, yielding inaccurate results for a test. That's why most testers who need accurate, professional results don't use a firewall on their testing machines. Paul Asadoorian, host of the Paul Security Weekly podcast that covers various security and hacking issues, refers to the practice of hacking without a firewalls as "hacking naked." Without a firewall, the testing machine is not as protected as other systems in your network. Thus, don't store sensitive data on the testing machine, other than the temporary test results, which must be moved to another box in a timely fashion.

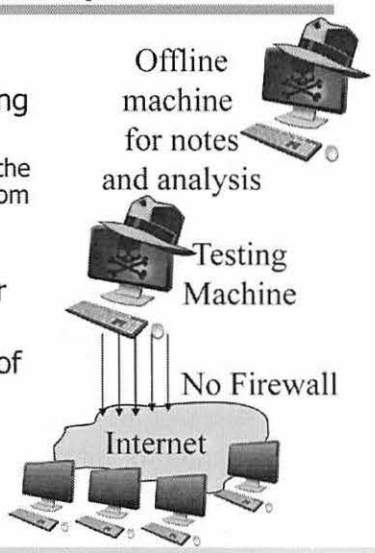
Network Address Translation (NAT) of firewalls cause problems for testers because they alter the source IP address of packets on their way out of the network and map any response packets back to the original IP address. Port Address Translation (PAT) functionality not only alters source IP addresses of outbound packets, but it also alters source port numbers so that the response to the packet can be associated back with the originating host. Both NAT and PAT rely on tables in the firewall. A scan that creates millions of packets may overwhelm the NAT or PAT tables, causing them to drop packets, including subsequent probes and their responses. Thus, the scanning tool won't see all the results, missing open ports and vulnerabilities.

Furthermore, HTTP proxies sometimes alter the encoding of various web traffic that passes through them, which could break carefully calibrated exploitation code. An exploit passing through an HTTP proxy may simply not function at all or could cause a target system to crash.

The application-level inspection technology of some firewalls drops packets that do not properly confirm to the protocol, or even clean up some packet settings, forcing them to match the protocol spec as interpreted by the firewall vendor. Again, these changes could neuter an exploit or cause damage to the target.

Avoid Firewall and IPS on Testing Network and Systems

- We strongly recommend not using a network firewall (and even a personal firewall) on the testing network and testing system(s)
  - Note that we are ***not*** talking about removing the firewall from the target environment ... just from the tester's machine and tester's network
- Even a host-based IPS on your testing machine could block outbound exploits or scans
- Record notes on a separate machine off of the network
  - Also, use this separate machine for detailed analysis
- Copy information to the notes machine using a USB flash drive



Network Pen Testing and Ethical Hacking 43

For all these reasons, we recommend that the testing machine not be located behind a network firewall, nor should it use a personal firewall on that specific system. Of course, the target network likely has a firewall, if it is tested across the Internet. We're not talking about removing that firewall. Instead, we're referring to the pen tester's testing systems connected to the Internet and how they should not be located behind a firewall. Even a host-based IPS on your testing machine could block outbound exploits or scans. For example, some commercial endpoint security suites with IPS functionality automatically filter Metasploit exploits and payloads on the outbound when a penetration tester uses a machine with the given endpoint suite installed. For this reason, many penetration testers opt to test without any IPS or firewall functionality on their own systems.

In addition, use a separate, offline system not connected to the testing network to take notes while you perform a test. This separate machine provides a more secure location for housing your notes, plus it gives you flexibility to analyze interim results while the testing machine is tied up with performance-draining scans and other activities. You can move files between the testing machine and the offline machine using a USB flash drive.

Harden Testing Systems Carefully

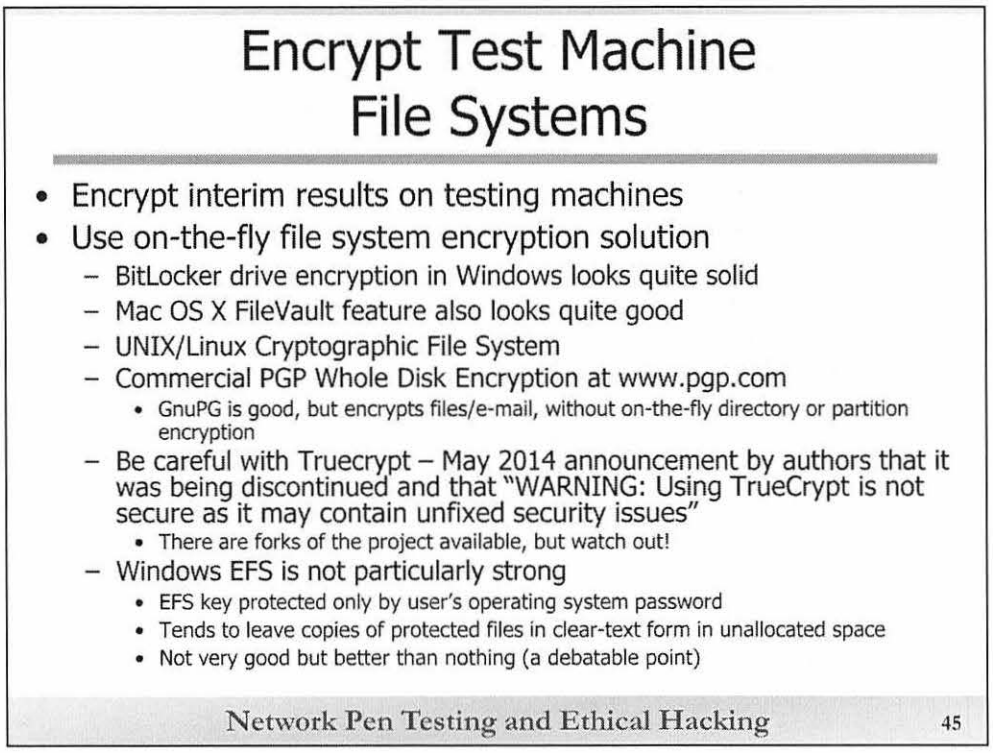
- Make sure you thoroughly harden the testing machines
  - You don't want them to get compromised during a test:
    - By a third-party malicious attacker
    - By an over-exuberant admin on the target network
    - Such a compromise could be embarrassing or even catastrophic: exposed test results

- Shut off unneeded services
- Increase security settings, but not to the point in which you inhibit the functionality of your testing tools
  - Example: Windows LMCompatibilityLevel Registry key requiring NTLMv2 may make the tester's machine less likely to find some flaws in some Windows servers ... testing machine needs to support LM Challenge-Response, NTLMv1, and NTLMv2
- The Center for Internet Security has free templates for hardening Windows and Linux (as well as other operating systems and environments)

Although we recommend that you avoid using a network or personal firewall to protect your testing machines because of the potential impact on your test results, we do caution you. Make sure you carefully harden the testing machines before starting a test. You must guard against compromise of the attacking machines during a test, by either a third-party malicious attacker or even an over-exuberant system administrator in the target organization. There have been cases of an administrator on a target network launching a counterstrike during a test, hacking back to the penetration tester's machines and compromising them. If someone compromises your testing machines, they could steal your interim test results and even alter the results they leave behind. Such exposure of test results could be, at best, embarrassing, and at worst, catastrophic for your career as a penetration tester or ethical hacker.

For this reason, keep patches up to date on all your testing machines, and shut off unneeded services. For a penetration testing system, you likely need no or only minimal listening services on the machine. The only services that should be listening are specific ones you need for your test, such as a web server set up to deliver a client-side exploit or a file server needed to serve up files to compromised target machines. You want to increase security settings of the testing machines beyond the defaults for the given operating system, but make sure that you don't inhibit the functionality of your testing tools. Harden the boxes, but verify on lab systems that your hardening process doesn't break needed functionality of your test tools. One area of hardening involves configuring a Windows machine to speak the stronger NTLMv2 protocol for authentication using the LMCompatibilityLevel Registry key. To attack less secure Windows servers, the tester's machine needs the capability to speak the older LANMAN Challenge-Response and NTLMv1 authentication mechanisms in addition to NTLMv2. We explore the differences between these authentication mechanisms in more detail in 560.4. For now, keep in mind that hardening a tester's machine to avoid these older and weaker protocols may hamper the system's capability to attack weakly configured targets.

To help with this hardening process, the Center for Internet Security (www.cisecurity.org) has a large number of free templates for hardening various kinds of systems, including Windows and Linux. Download and use these templates.

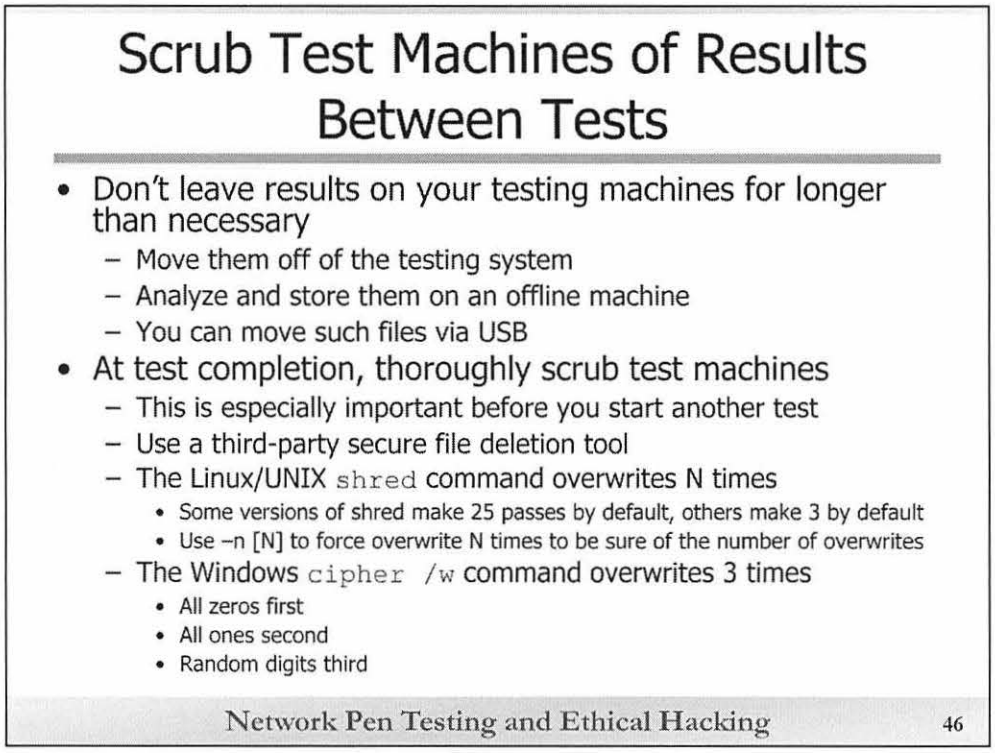


In addition to hardening your testing machines, you also need to be careful with the data on those machines. Professional penetration testers and ethical hackers should consider using a file system encryption solution on their testing machines to lower the probability of test results exposure. Ideally, you'll use an on-the-fly encryption solution that encrypts an entire directory or entire partition, letting you seamlessly drop files into a directory to have them automatically encrypted.

Numerous file system encryption solutions are available. The Windows BitLocker feature included in Windows looks quite solid; although, it functions only on Windows machines. The Mac OS X FileVault feature likewise provides a good boost of security in protected files but is again focused on a single platform: Mac OS X. The Cryptographic File System (CFS) has been ported to most Linux and UNIX variations. From a commercial perspective, the PGP Whole Disk Encryption tool provides useful functionality, with vendor support. Gnu Privacy Guard (GnuPG) is a fine free, open-source tool for encrypting files and e-mail. However, as of this writing, GnuPG does not support on-the-fly directory or partition encryption.

Unfortunately, in May 2014, the authors of the free Truecrypt tool announced that they were discontinuing development, and that, "WARNING: Using TrueCrypt is not secure as it may contain unfixed security issues." There are projects that have forked the original TrueCrypt code, but be careful in trusting such tools.

Windows has built-in file system encryption functionality called the Encrypted File System (EFS). Unfortunately, the security of EFS leaves a lot to be desired. It protects crypto keys only with the user's operating system password, which can be extracted and cracked, as opposed to the separate passphrase most other solutions employ. Furthermore, EFS often leaves clear-text copies of recently deleted or encrypted files in unallocated space of the file system, until they are wiped. Although EFS is not a particularly strong solution, many people believe it is better than nothing. However, some worry that EFS, with its problems, may lull people into a false sense of security. Thus, we recommend that you use a stronger solution, such as TrueCrypt, PGP, or BitLocker.

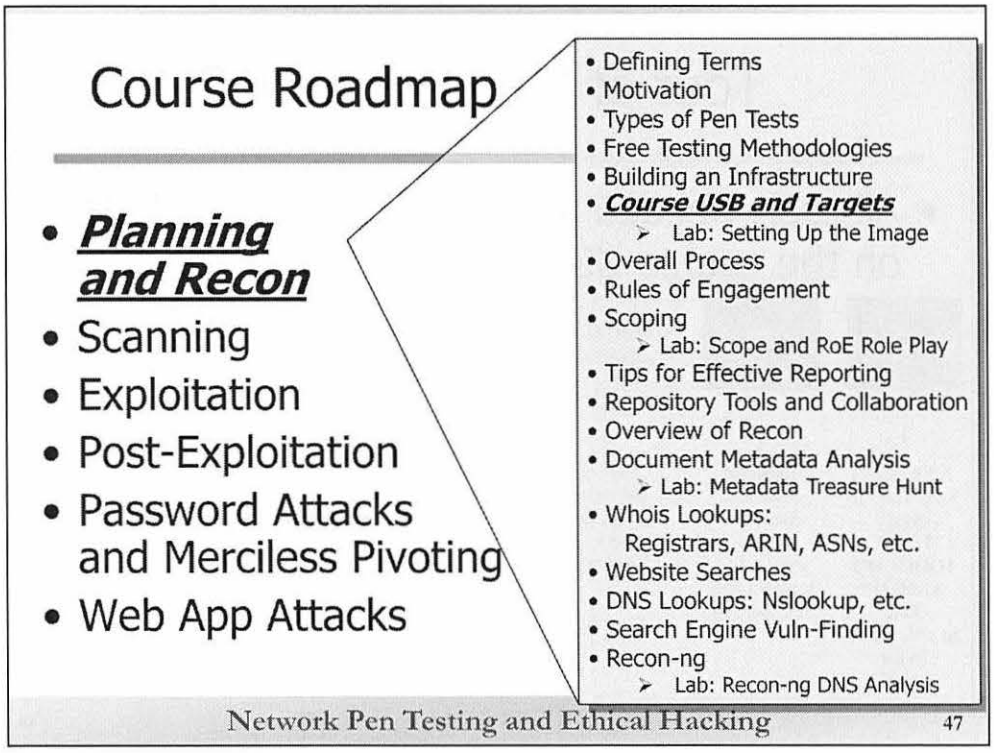


By itself, encrypting your sensitive test results information is not enough. You must also securely remove your test results from your testing machines. Don't leave data on the testing machines for any longer than is necessary during a test. Move results files, including output from scanning tools, files containing password guessing results, and any notes that you create, to your notes and analysis machine on a regular basis using a USB flash drive.

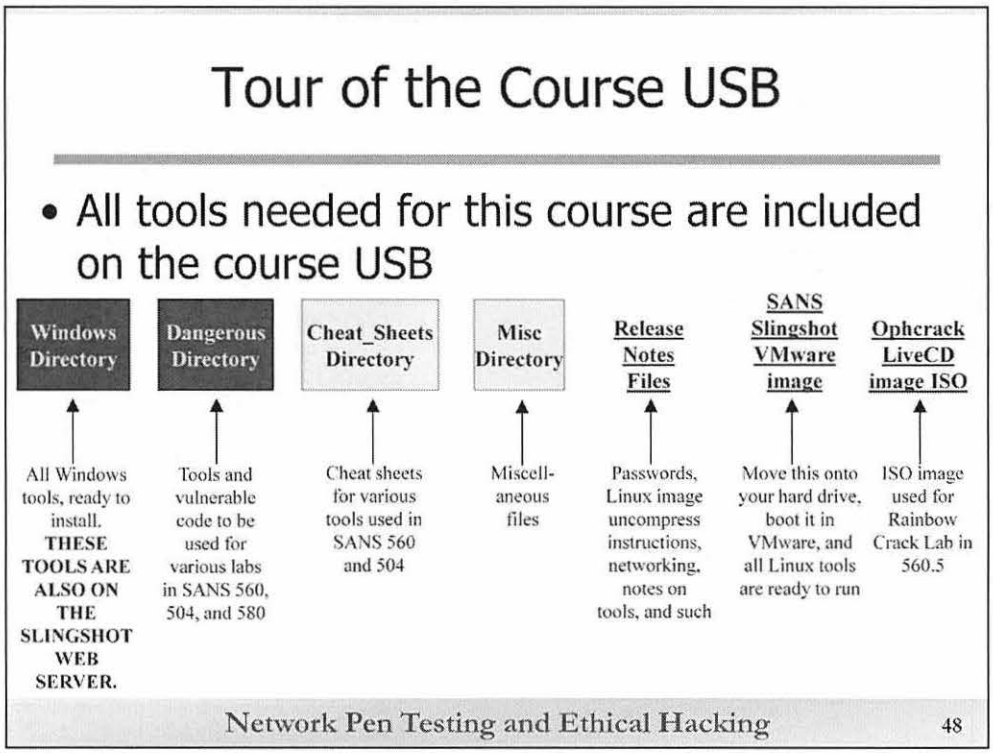
Periodically during a test, and especially when a test is completed, you must securely wipe interim results files from the testing machines. This is especially important before you start another test of another target environment because you don't want to mix up your results or have the results of one target exposed to another target.

Merely deleting the files isn't enough because that simply moves them into unallocated space, making them still recoverable. Instead, use a secure file deletion tool that overwrites files with alternating zeros and ones multiple times to make sure the file cannot be recovered. Most versions of Linux have the `shred` command, which overwrites files with alternating zeros and ones. Some versions of shred overwrite files 3 times by default, whereas others overwrite 25 times by default, making files almost impossible to recover. To be sure of the number of overwrites, you should specify –n [N], where N is the number of overwrite passes.

Windows has a built-in tool called `cipher`, which can be used with the /w:[file] option to overwrite all free space on the partition where [file] resides with all zeros, followed by all ones, followed by random numbers (3 passes total). Although 3 passes is less than we'd like to see, it is acceptable for most environments.



We'll now go over some of the technical infrastructure components of the class, including the course USB and the target machines we'll be working with all week. Make sure you have a copy of the course USB handy as we go through the next several slides.



On the course USB, at the top of the directory structure, there are a handful of files and directories. Among the most important of these files are the Release Notes for the course image, which are named "Release Notes for SANS [version_number]." They are included in both .doc and .txt format, so you can read them in any doc-format-compatible word processor or even any text editor. The Release Notes include the userIDs and passwords for the VMware image for the course, as well as information about getting that VMware image uncompressed, booted, and networked. They also contain additional notes about some of the individual tools. ***As long as you have the course USB, you will also have these Release Notes, and therefore, you will have access to the root password for the course VMware image. Please remember this so you won't be stranded without the password.***

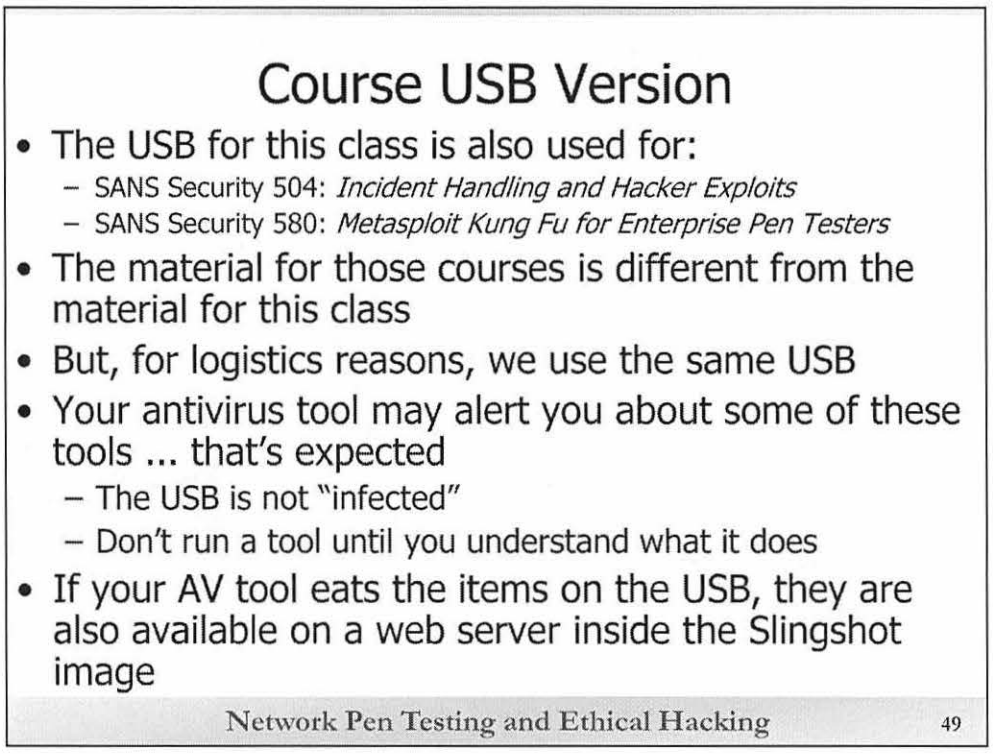
Next, we have the course Windows directory. All Windows tools that you need for the course are included here. You have to install them on your Windows machine when the time comes for each lab. Do wait, though, until we start a given lab so that you can understand a tool before installing it. Also, all these Windows tools are included on a web server on the Slingshot Linux image on the course USB.

The next directory is called "Dangerous" because it includes programs that could introduce vulnerabilities on your system or open backdoors. We use such programs in labs at various points in the course to illustrate pen testing techniques for exploiting such flaws. Do not run these items until you understand what they do.

Another directory, called Cheat_Sheets, contains cheat sheets for various tools covered in this class. You can look through it. These sheets can be helpful in labs throughout the course and the pen test workshop in 560.6. And a Misc directory contains miscellaneous files, including a sample pen test report.

The next element of the USB is a large file called SANS_Slingshot[Version].zip. This zipped file contains a VMware image of a Linux machine, ready to boot in VMware Workstation, VMware Player, or VMware Fusion (the Mac OS X product). The course laptop requirements document specify bringing a copy of VMware with you. VMware is not included on the course USB, due to redistribution limitations imposed by VMware.

And, finally, we have an ISO image of a bootable CD called the Ophcrack LiveCD. We use this image in a lab in 560.5 to conduct Rainbow table password cracking.

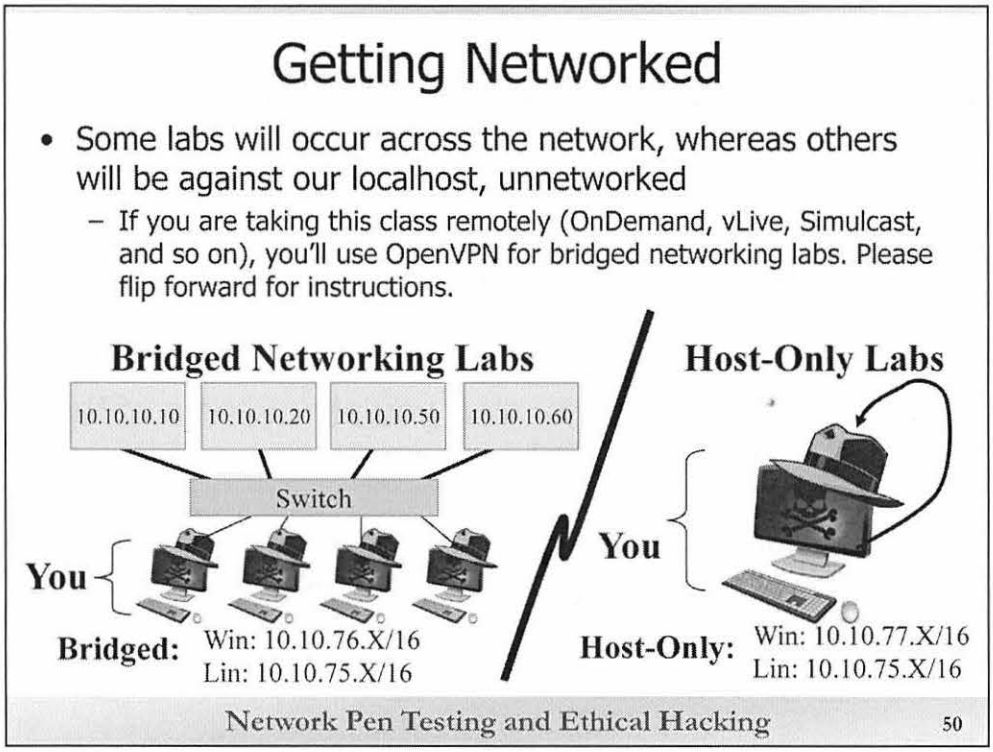


The USB that you received for this course is also used for other SANS courses, including SANS Security 504 (the Incident Handling and Hacker Exploits course) and SANS Security 580 (Metasploit Kung Fu for Enterprise Pen Testers).

The material and topics covered for each of these courses is different, but we keep them all on the same USB for logistics reasons. (Making sure that the appropriate USB is in the right place at the right time would be more complicated with different USBs for each course.) This arrangement has positive implications for you because you get all the tools for each of the courses, all on one handy USB, with the SANS Slingshot Linux image preconfigured to run them all.

However, note that your antivirus and antispyware tools may alert you regarding some of the items on the course USB. The classes that the USB supports deal with computer attacks and exploits, and some antimalware vendors classify particular tools as malicious code. The course USB is not "infected." Instead, it merely contains software that can be used in a malicious fashion, and therefore it triggers the signature-based detection of some antivirus and antispyware tools. Don't run a program that your antivirus tool alerts you about unless you understand what the program is designed to do and until you understand how to use the program safely.

If your antivirus tool deletes any of the Windows files on the USB, we have included a copy of them inside the Slingshot virtual machine on a web server running. You can simply download those tools using a browser on Windows accessing your Slingshot Linux image.



Throughout this course, we'll have numerous hands-on labs, so you can gain experience by practicing the various techniques we'll describe. Most of the hands-on labs for this class will occur across a network, but some will occur against your local host.

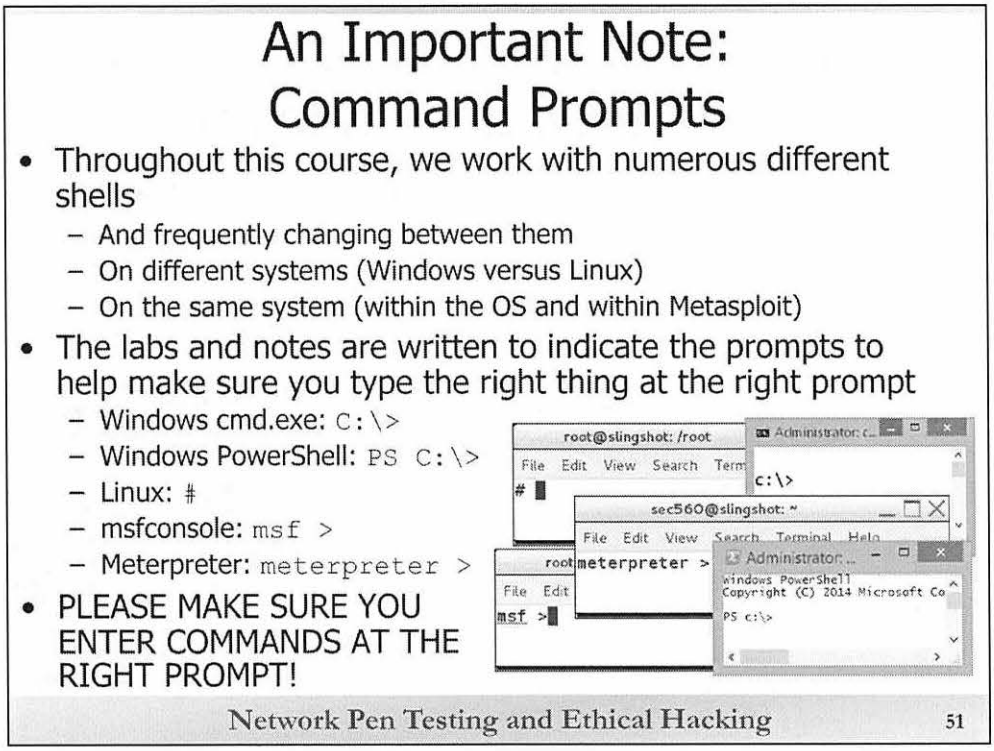
For networked labs, you'll be attacking four target machines on the 10.10.10 subnet, including 10.10.10.10, 10.10.10.20, 10.10.10.50, and 10.10.10.60. YOU ARE ALLOWED TO ATTACK ONLY THESE MACHINES. DO NOT ATTACK YOUR FELLOW ATTENDEES' SYSTEMS. IF YOU DO ATTACK OTHER MACHINES OUTSIDE OF THE 10.10.10 NETWORK, YOU MAY BE DISMISSED FROM THE CLASS, AND THERE COULD BE LEGAL IMPLICATIONS.

If you are taking this class at a live conference, you will be connected to these targets using one or more switches provided in the room. If you are taking this class across the Internet (via vLive, OnDemand, or other offering), please flip forward to the slide that describes your VPN configuration.

For *Bridged Networked Labs*, your virtual machine configuration (VMware) should be set to Bridged networking, with your Windows IP address being 10.10.76.X (X will be given to you by your instructor or a room facilitator) with a netmask of 255.255.0.0. No default gateway or DNS is required. Your Linux IP address will be 10.10.75.X (the same X you received from the instructor or facilitator) with the same netmask.

For the *Host-Only Labs*, you will be running attacks from your virtual machine guest against your host machine. In these labs, you'll need to configure your virtual machine for host-only networking, and your Windows IP address will be 10.10.77.X. (Note that it is 76.X for Networked Labs and 77.X for Host-Only Labs.) Your Linux IP address will always be the same: 10.10.75.X.

We will now cover how to configure your systems for these labs.



Throughout this course, we use numerous different shells, both in our operating system and within Metasploit. We frequently change between these different shells as we switch back and forth between Linux and Windows, and within different aspects of Metasploit. Sometimes, even on a single page in the book, you use two or even three different types of shell to do something and then observe the results.

All the labs and notes were carefully written to indicate the proper shell you are supposed to use at any given time by including the shell prompt right before each command you are supposed to type. That is, each lab command is preceded by the prompt indicating which shell to use. The shell types you encounter throughout this class include

A Windows cmd.exe with the prompt:

```
C:\>
```

A Windows PowerShell with the prompt:

```
PS C:\>
```

A Linux bash shell (which we'll run as root) with the prompt:

```
#
```

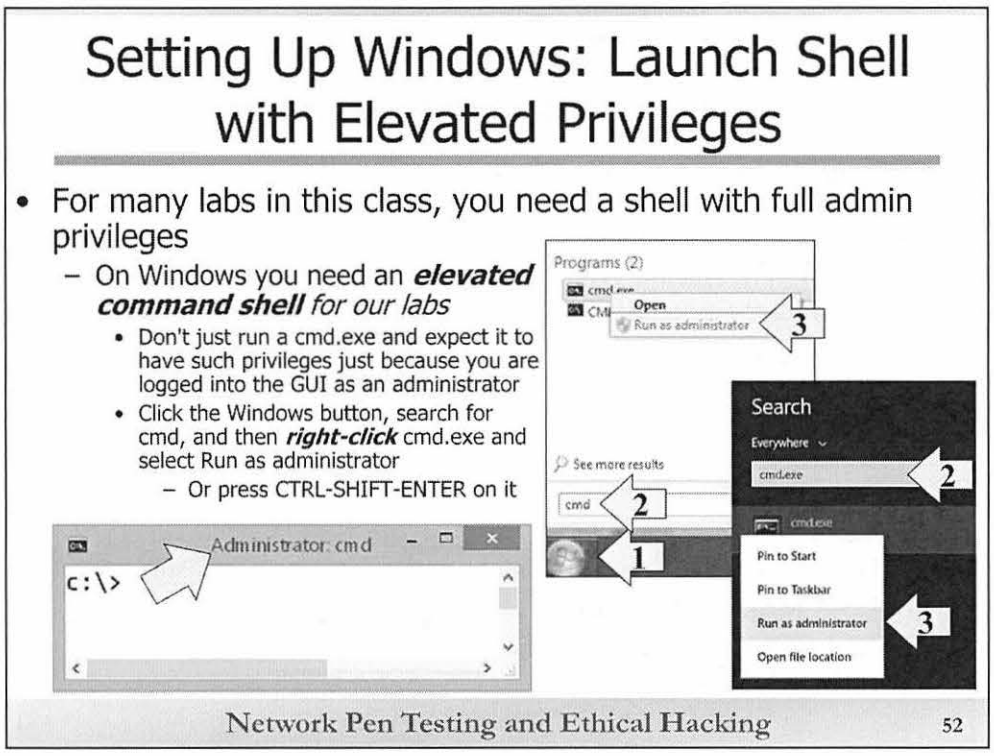
The Metasploit Framework Console with the prompt:

```
msf >
```

A Meterpreter shell with the prompt:

```
meterpreter >
```

DOUBLE-CHECK AT EACH LAB STEP THAT YOU ARE ENTERING THE PROPER COMMAND INTO THE PROPER SHELL. Otherwise, a given lab step will not work for you properly.

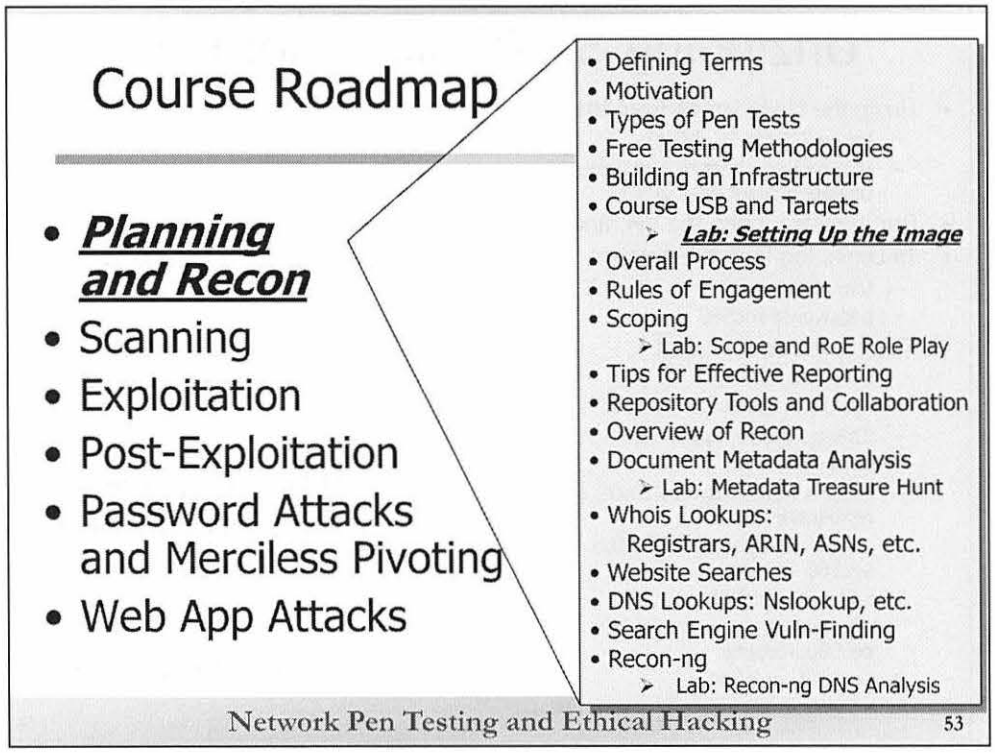


Regarding Windows shells, for many labs in this class, you need a command shell with full administrator privileges. On Windows XP or 2003, getting such shell access is easy. If you are currently logged into the GUI with admin privileges, simply go to Start→Run and type **cmd.exe**. Or if you aren't currently logged in as admin, simply launch a non-admin cmd.exe and then use the runas command to launch a cmd.exe via the command runas /u:[AdminUser] cmd.exe.

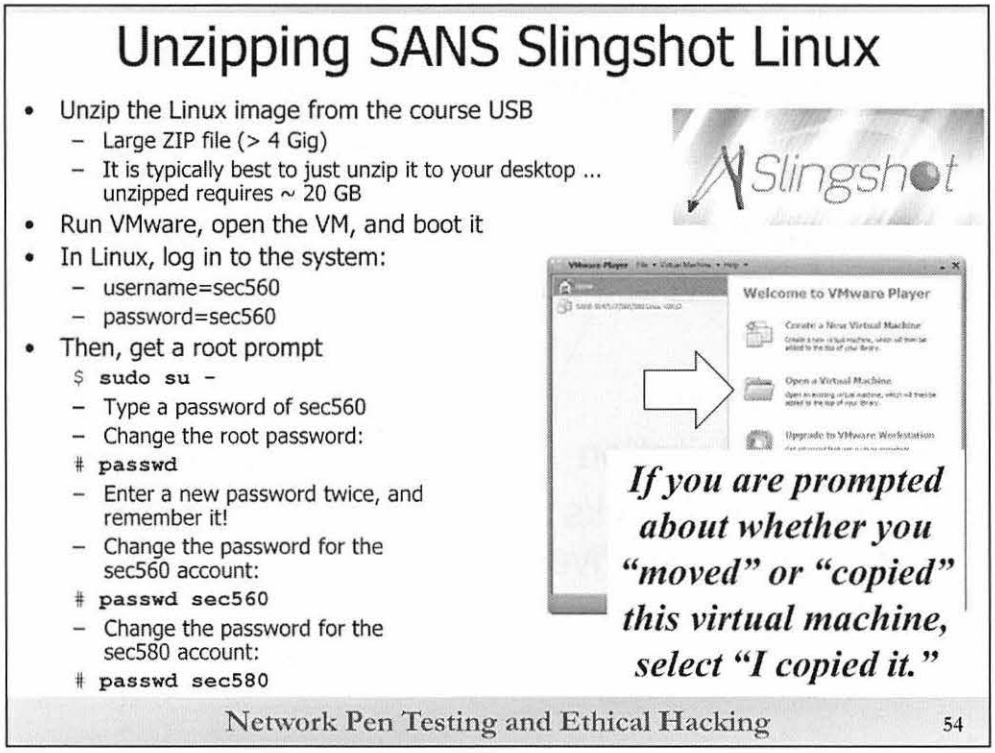
Special Note for Windows Vista, 7, 8, 8.1, and 10 users:

If you use Windows Vista, 7, 8, or 8.1, when you simply invoke a cmd.exe via the GUI, you won't have full administrator privileges, even if you are logged into the GUI as an administrative account. That's because Windows is trying to protect your system from you, not giving you full administrator privileges at the command line unless you specifically demand it. For many of the commands you are going to run in labs throughout this class, you need an **elevated command shell**, which has full admin privileges. To get such shell access, go to your Windows icon (Step 1 in this slide). Do a search for cmd (Step 2). And in Step 3, right-click cmd.exe, selecting Run as administrator. Now you have an elevated command shell, which you can use for all the labs in this class that require admin access.

At any time beyond that point, you can tell that a shell is running with elevated privileges because its title bar will have the word "administrator:" in it, as shown on this screen.



Let's now conduct a lab to network your Windows and Linux systems. In this lab, we walk through setting your Windows and Linux IP addresses as well as configuration options within VMware.



Start by unzipping the Slingshot Linux guest VM image from the USB onto your hard drive. Unzip all the files included in the large ZIP image on the course USB. It is usually best to just unzip its entire contents into a directory on your desktop for easy access throughout the course. This file requires approximately 20 GB of space on your hard drive when unzipped.

After the file is unzipped, run VMware, select Open a Virtual Machine, and boot your Slingshot Linux guest system. ***If VMware prompts you about whether you "moved" or "copied" this virtual machine, select "I copied it." If it doesn't prompt you, that's okay.***

When prompted, log in to the guest machine using the following credentials:

Username=sec560

Password=sec560

Now, become root by running

```
$ sudo su -
```

Type in a password **sec560** to finish su'ing to root.

Now, change root's password to a value you'll remember but that isn't easily guessed or cracked. We'll be connected to a network with other students in this course, and you do not want them to know the password for your Linux VMware image.

```
# passwd
```

Enter your chosen password once, and then again to set it.

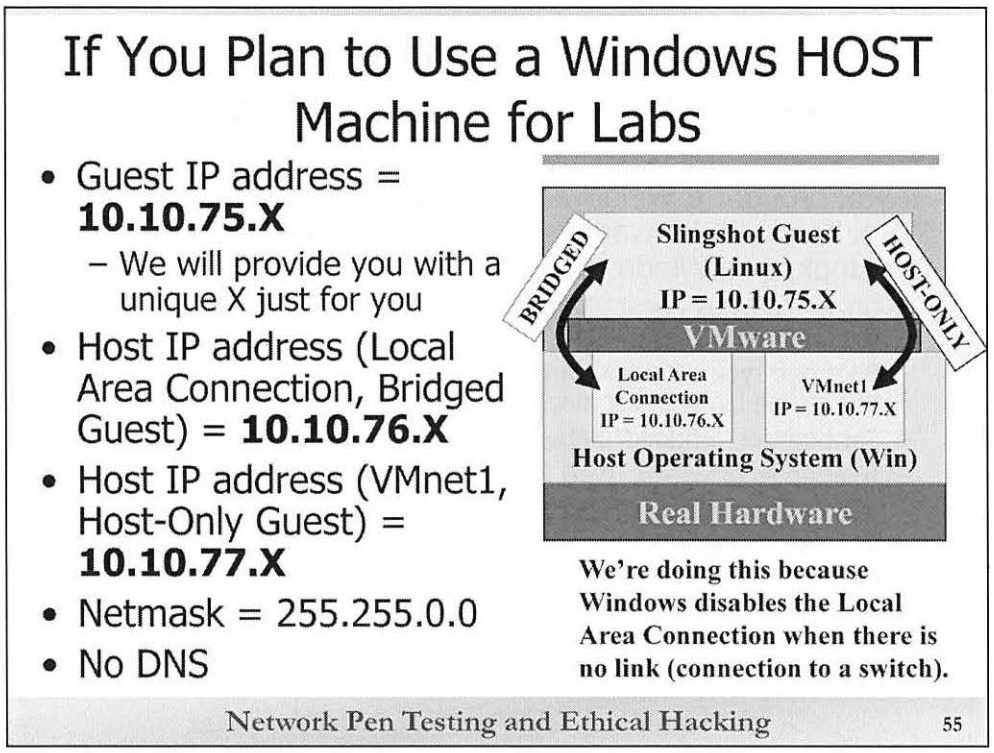
Change the password for the sec560 account:

```
# passwd sec560
```

Again, please change the password for the sec560 account to something you will remember.

Finally, change the password for the sec580 account:

```
# passwd sec580
```



Many students who take this class use a Windows host machine (not a virtual machine) for the labs, along with our Slingshot Linux guest machine. We will now discuss the networking configuration for that approach.

If you plan on using a Windows GUEST machine (that is, a virtual Windows machine), please flip forward to the next slide, titled, "*If You Plan to Use a Windows GUEST Machine for Labs (with Mac OS X, Linux, or Windows Host Machines")*.

Your Linux Guest will have an IP address of 10.10.75.X, where X, the last octet of your IP address, will be given to you by the course instructor. Remember this X. In fact, write it down so that you have it with you for the remainder of the course.

Your Windows Host machine will have a Local Area Connection address of 10.10.76.X. This will be the address you'll use when VMware is configured for Bridged networking for a Networked Labs.

Your other Windows Host IP address will be for VMnet1, a network interface created when VMware was installed. Its address should be 10.10.77.X. You'll use this address when performing labs against your own machine (Host-Only Labs) when not connected to a switch, using VMware in host-only mode.

The netmask for all your interfaces should be 255.255.0.0, a /16 network. We have no DNS server or default gateway for the labs. There is a DNS server located at 10.10.10.60, but you do not need to configure your machine to use it. We are a flat (nonrouted) network here, so there is no default gateway for the labs in 560.1 through 560.5. We will have a separate network configuration with routers and a target DNS server for 560.6, when we conduct our final penetration testing workshop.

If You Plan to Use a Windows GUEST Machine for Labs (with Mac OS X, Linux, or Windows *Host* Machines)

- If you brought a Windows GUEST virtual machine, with a Mac OS X host with VMware Fusion or a Linux host with VMware for Linux, or a Windows *host*
  - Unzip the Linux guest VM from the course USB
  - Boot both your Windows and the course Linux guest machines
  - Set both for "Bridged" networking
  - IP address of Win = 10.10.76.X
  - IP address of Linux = 10.10.75.X
  - You can configure your host machine's IP address to 10.10.78.X.

Network Pen Testing and Ethical Hacking 56

Some people who take this class do not use a Windows host machine, but instead rely on a Windows *guest* running on VMware Fusion on Mac OS X or VMware Workstation on Linux. Alternatively, some people have brought a Windows *host* that they don't plan to use for labs, but instead plan to use their own Windows *guest* for labs. Such systems will still work with our guest Linux machine, and the networking becomes a little bit easier.

If your have a Mac OS X, Linux, or Windows *host* machine (and your own Windows *guest* machine), you will still need to unzip the VMware Linux system we provided on the course USB to your hard drive. This will be one of your guest machines. You were required (in the course laptop instructions on the registration page for the course) to bring a Windows guest machine with you.

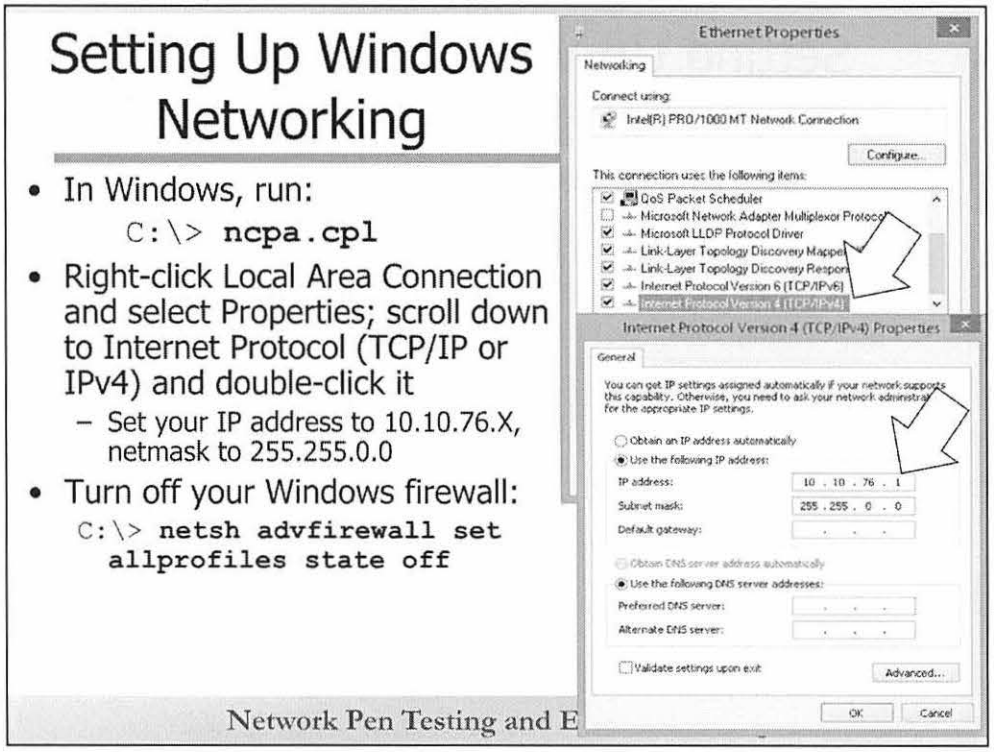
Boot both your Windows guest VM and the Linux guest VM unzipped from the course USB.

Set both of your guest machines to "Bridged" networking.

Your Linux IP address will be 10.10.75.X, with a netmask of 255.255.0.0.

Your Windows IP address (for your Windows GUEST) will be 10.10.76.X, also with a netmask of 255.255.0.0.

For your host machine (Linux, Mac OS X, or Windows HOST), use the IP address 10.10.78.X (again, with a 255.255.0.0 netmask).



You can set up your Windows networking by opening up your network interfaces. One of the easiest ways of doing this is to launch (at an Administrator cmd.exe prompt):

```
C:\> ncpa.cpl
```

You should see all your networking interfaces. Right-click your *Local Area Connection* interface and select Properties. Then, scroll down to where it says TCP/IP or TCP/IPv4 and double-click. Then, set your IP address to 10.10.76.X and netmask to 255.255.0.0. Click OK and then Close.

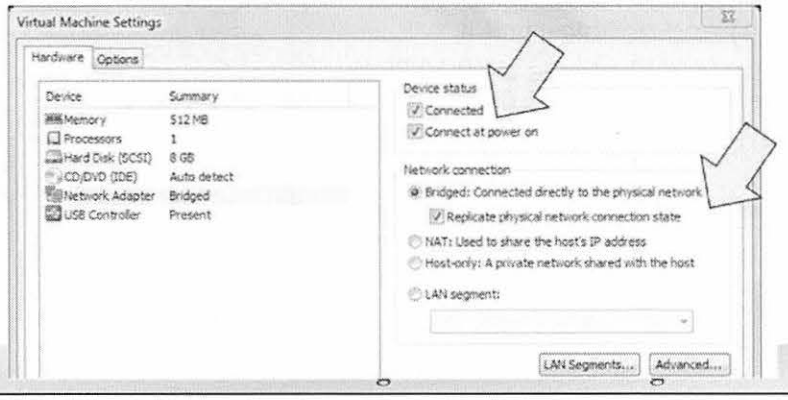
And, finally, we need our Windows firewall to be disabled so that we can get unfettered access to and from our machines. Disable the built-in Windows firewall by running the following at an elevated command prompt:

```
C:\> netsh advfirewall set allprofiles state off
```

If you have a third-party firewall on your Windows box, you need to disable it, or create exceptions to allow your guest and various applications to communicate for our labs.

Setting Up VMware Networking

- In VMware, go to VM→Settings ... Or Player→Manage→Virtual Machine Settings... Or, with VMware having your focus, hit CTRL+D
- Look at the "Network Adapter" settings
- For Networked Labs, use Bridged networking
  - Make sure Connected and Connected at power on are selected
  - Also, select Replicate physical network connection state
- For Host-Only Labs, use Host-only networking



58

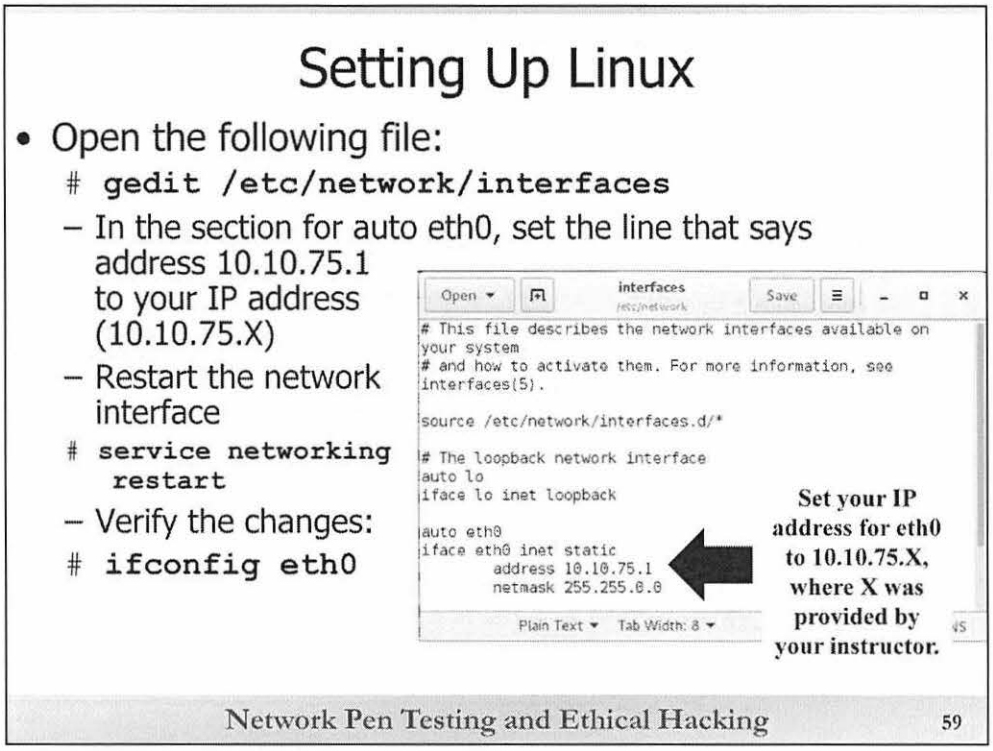
Next, we'll set up VMware. You can set your network adapter's configuration by going to VM→Settings or Player→Manage→Virtual Machine Settings. Alternatively, if VMware has the focus of your GUI, you can hit CTRL+D.

In VM→Settings, click Network Adapter.

For Networked Labs, select Bridged networking. Also select Replicate physical network connection state. Make sure that Connected and Connected at power on are both enabled.

When we switch to a Host-Only Lab, you need to go back into this setting (VM→Settings ...) and select the Host-only radio button.

For now, make sure you are set to Bridged networking for our first set of labs.



Now, we'll set up our Linux networking. In your Linux guest virtual machine, set the IP address by editing a file, which you can access by running this command (at a root-level shell with a # prompt):

```
# gedit /etc/network/interfaces
```

Let the tab-autocomplete do most of that typing of this command for you, so you can make sure you avoid any typos.

Inside the file, find the line in the section under auto eth0 that says "address " and change 10.10.75.1 to the IP address for your Linux machine (10.10.75.X), again using the X value provided to you for the class.

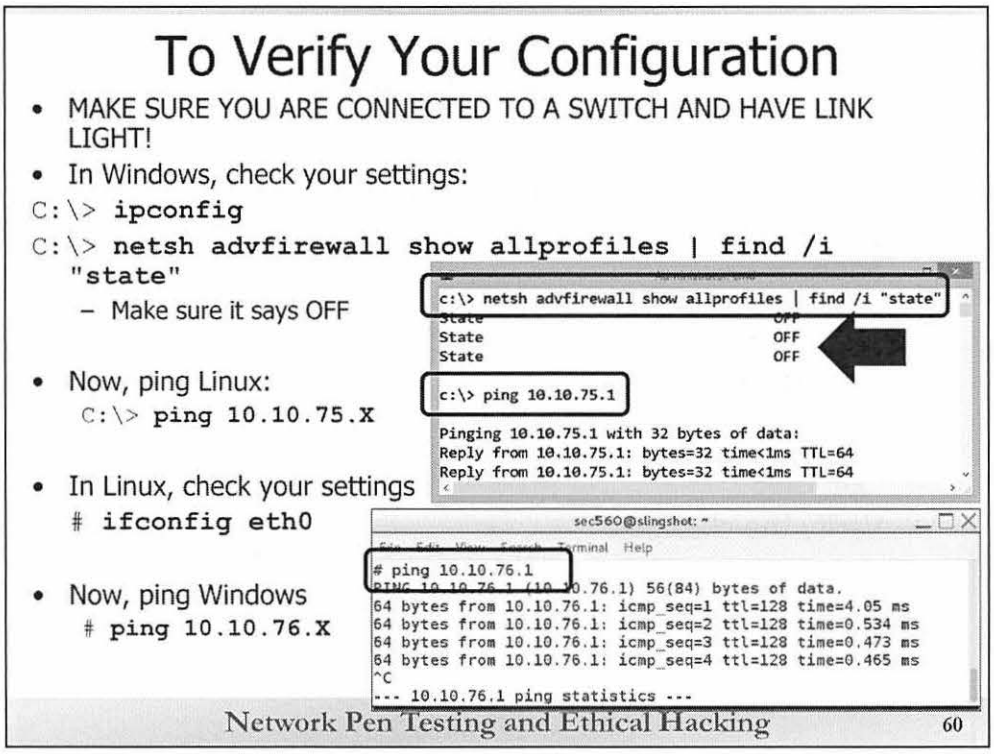
Restart your network interface to apply the changes:

```
# service networking restart
```

Let's verify that the changes were applied; look at the IP address in the output:

```
# ifconfig eth0
```

Your IP address in the output of this command for eth0 should be 10.10.75.X.



Finally, let's verify our configuration and make sure we have connectivity between our guest and host machines. We'll start from Windows and then work our way to Linux.

FIRST, BECAUSE WE ARE USING BRIDGED NETWORKING, YOU NEED TO MAKE SURE YOU ARE CONNECTED TO A SWITCH. YOU MUST HAVE LINK LIGHT FOR BRIDGED NETWORKING TO WORK WITH AN ETHERNET INTERFACE ON A WINDOWS HOST MACHINE.

In Windows, first check your IP address for your Local Area Connection:

```
C:\> ipconfig
```

Then, check your firewall settings, making sure all the output lines say OFF:

```
C:\> netsh advfirewall show allprofiles | find /i "state"
```

Now, from Windows, try to ping Linux:

```
C:\> ping 10.10.75.X
```

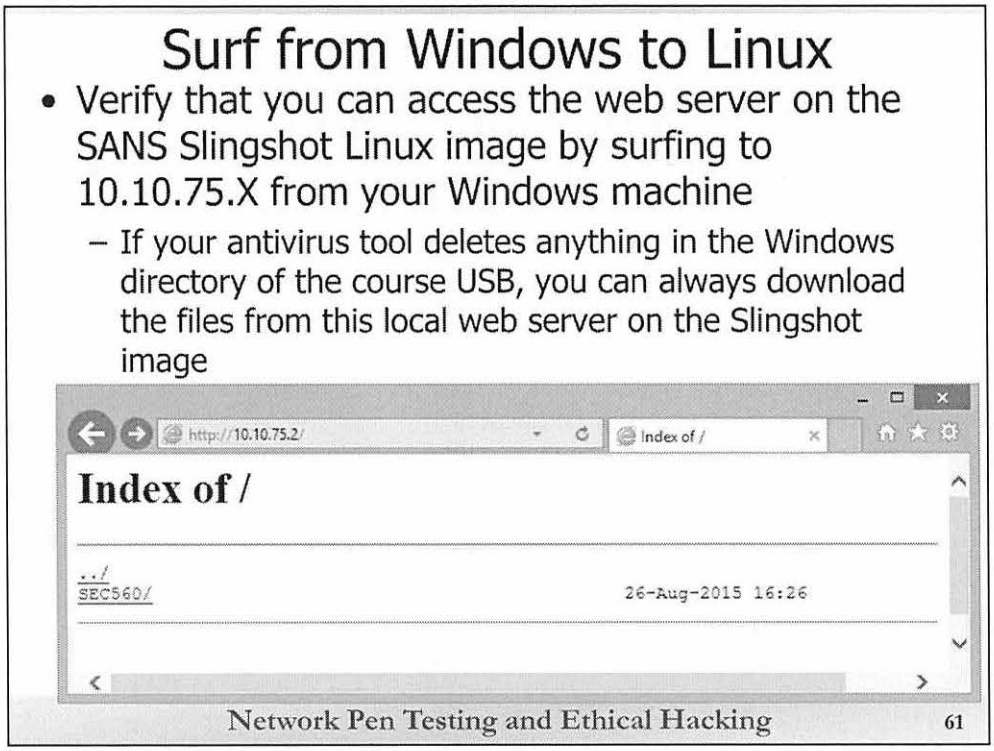
Then, in Linux, verify your network configuration:

```
# ifconfig eth0
```

And, finally, try to ping Windows:

```
# ping 10.10.76.X
```

If you see ping responses from Windows to Linux and from Linux to Windows, you are configured and ready for the labs. If not, contact the instructor or TA.



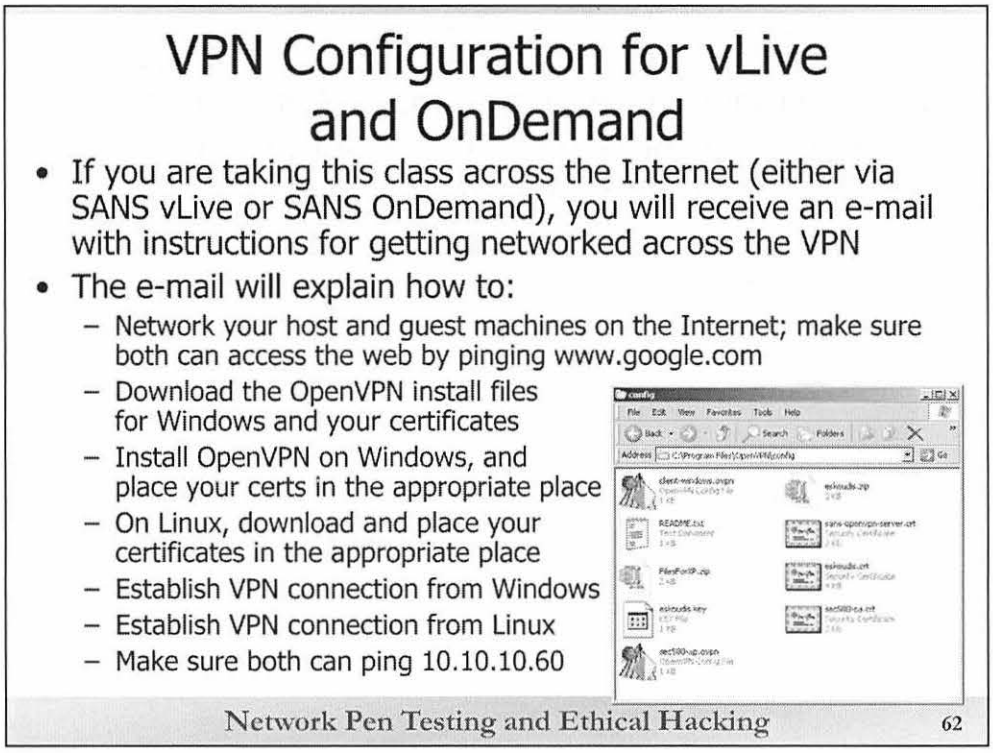
The Slingshot Linux distribution includes a web server listening on TCP Port 80, which serves up all the Windows tools and various other files associated with course labs.

If, during class, your antivirus tool deletes a Windows tool from your system or the course USB drive, you can retrieve the given file from this local web server in your own copy of the Slingshot image.

If your Windows and Linux systems are both networked, you can test this web server by running a browser on Windows and surfing to

```
http://[YourLinuxIPaddress]
```

On the web page, you can find a directory associated with SEC560. In that directory, you can find subdirectories for CourseFiles and WindowsTools.

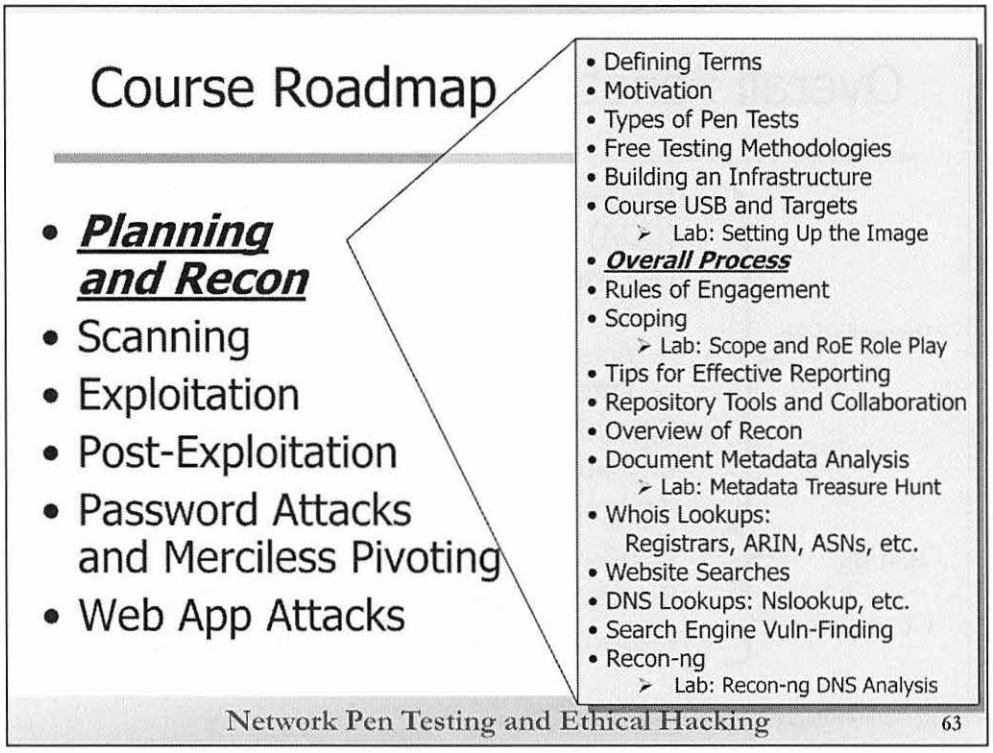


If you are taking this course across the Internet (either via SANS vLive or SANS OnDemand), you need to set up OpenVPN on your Linux and Windows machines to conduct the bridged networking labs in the class so that you can reach target systems we have prepared.

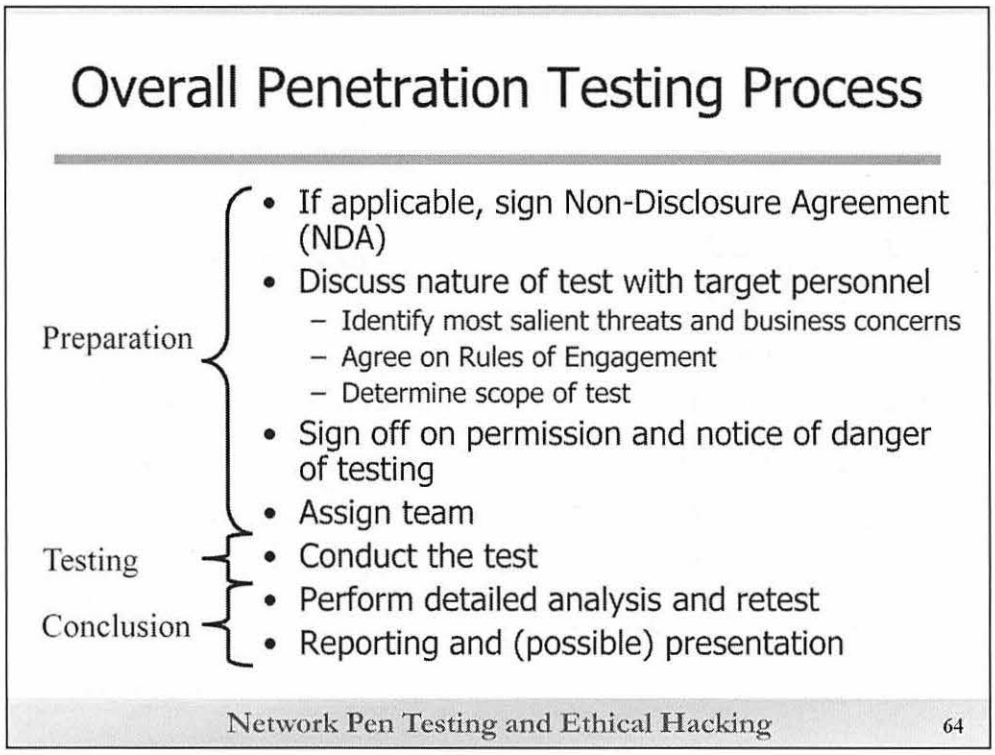
You will receive an e-mail from SANS NOC personnel that describes in detail the process for configuring your system to use the VPN. The e-mail will explain various steps, including

1. Set up your Linux guest and Windows host machine on the Internet. Both machines must reach Internet destinations. For Linux, use bridged networking, and configure your guest machine with an IP address for your environment or pull one using DHCP (edit /etc/network/interfaces). If you use hard-coded IP addresses, simply set it in the line that shows the IP address for eth0. If you use DHCP, make sure you change static to dhcp. Make sure both your Windows and Linux machines can ping some site on the Internet, such as www.google.com.
2. Download the OpenVPN install files for Windows, along with your certificates, as described in the e-mail from the SANS NOC. Put your certificates in the appropriate place (C:\Program Files\OpenVPN\config). You do not need to install OpenVPN software on the Linux guest image we provided because this software is already installed.
3. On Linux, place your downloaded certificates in the appropriate place (/etc/openvpn).
4. Establish the VPN connection from Windows (by right-clicking the OpenVPN icon in your tooltray and selecting Connect). Provide your SANS portal password to connect.
5. Establish the VPN connection from Linux (by running service openvpn start). Again, provide your SANS portal password when prompted.
6. Make sure both Windows and Linux can ping 10.10.10.10 while the VPN is connected.

Note: To communicate between your Linux guest and Windows host while connected to the VPN, you could use the IP address assigned to you by the VPN (viewable via the OpenVPN tooltray client in Windows and as the tap0 interface displayed by the ifconfig command on Linux) or the IP address of your network adapter (Local Area Connection in Windows and eth0 in Linux).



With our infrastructure in place, let's now go over the overall ethical hacking and penetration testing process. This process should be applied to all the testing that you do. Be careful in skipping any of the steps we describe. Some steps are designed to ensure that you've conducted a test with appropriate legal protections. Furthermore, other steps help ensure that you are providing demonstrable value to the organization you are testing.



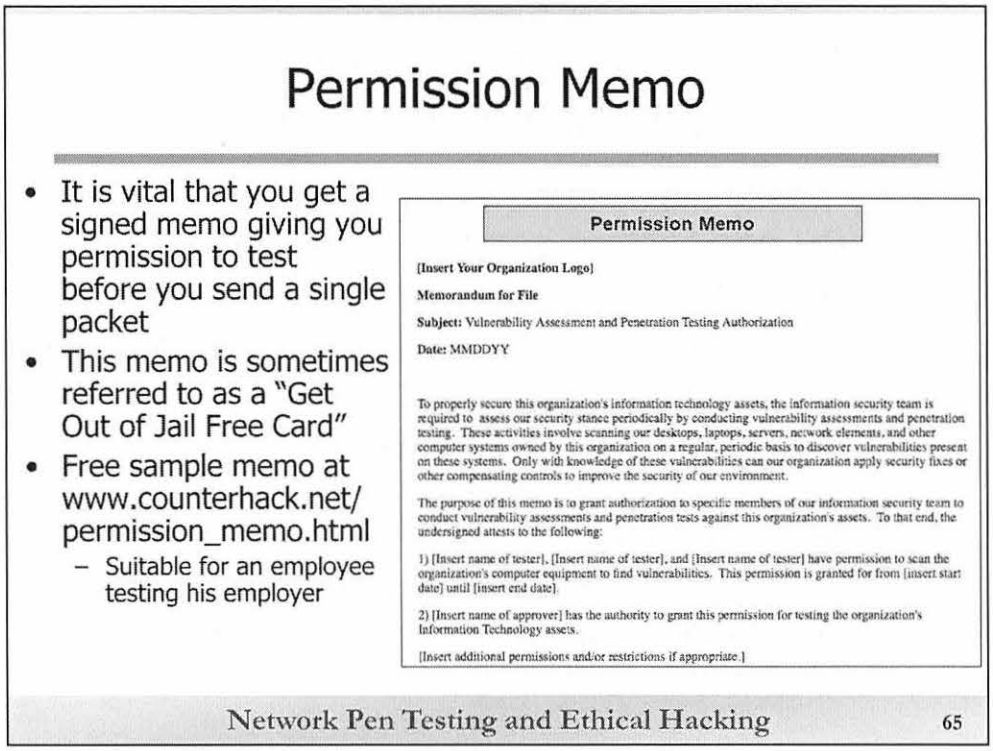
The overall penetration testing process involves preparation, testing, and conclusion phases.

During the preparation phase, the parties participating in the test may sign a non-disclosure agreement (NDA), especially if the test is conducted by a third-party organization. Then, the testers and the target personnel discuss the most significant concerns of the target organization. What are the biggest threats? Which systems are the most sensitive? What kind of information is the most valuable? We also agree on Rules of Engagement that describe how the testing will occur. Next, the scope of the test is determined, a process we'll discuss in depth later.

The next step is absolutely crucial. You need to get official, written permission to conduct the test, even if it is against targets in your own organization. This permission should notify the personnel associated with the target systems that there is some danger of their systems being crashed or impaired by the testing. We'll discuss this permission memo in more detail shortly. Then, based on the nature of the test, a team of appropriate testers is assigned, based on its technical areas of expertise and business knowledge of the target environment.

The test then occurs, potentially lasting from a day to many months.

To conclude, the team then analyzes the results, trying to discern its business implications. The technical details and business implications are described in detail in a final report. As findings are addressed, single issue retests could occur, or an entire comprehensive retest may happen. Some tests conclude with a wrap-up final presentation.



Let's zoom in on that one crucial element of the previous slide: the permission memo. It is absolutely vital that you get a signed memo from a leader of the target organization giving you permission to test its environment. This memo is sometimes called a "Get Out of Jail Free Card," or GOOJFC for short.

There is a free sample memo on the Counter Hack website at www.counterhack.net/permission_memo.html. Among other things, this memo states:

"The purpose of this memo is to grant authorization to specific members of our information security team to conduct vulnerability assessments and penetration tests against this organization's assets. To that end, the undersigned attests to the following:

1) *[Insert name of tester], [Insert name of tester], and [Insert name of tester] have permission to scan the organization's computer equipment to find vulnerabilities. This permission is granted for from [insert start date] until [insert end date].*
2) *[Insert name of approver] has the authority to grant this permission for testing the organization's Information Technology assets."*

Have your legal team review, tweak, and approve this language. Then, print it on corporate letterhead and have a chief information officer or similar level of management sign-off on it.

Note that this memo is suitable for employees to test the security of computer equipment owned by their employers.

Pen Test Companies: Limitation of Liability and Insurance

- By itself, that memo is not suitable for third-party pen testing companies to test their customers
  - That requires a limitations of liability agreement and contractual language
  - Should be drawn up by a lawyer
  - The liability is commonly limited to not exceed the value of the project
  - Furthermore, most penetration testing companies carry liability insurance and errors and omissions insurance

Network Pen Testing and Ethical Hacking 66

Although the permission memo on the previous slide is acceptable for employees testing their employers, it is not, by itself, suitable as a vehicle for a penetration testing company to test its customers' environments. It can act as a starting point for that more comprehensive document. But this third-party penetration testing agreement for client networks must also include a limitation of liability agreement and a contract. These items should be drawn up by a lawyer associated with either the penetration testing company or the client.

Most penetration testing companies include a limitation of liability agreement that caps the liability for any problems associated with the project at the price of the project itself. You wouldn't want one strangely configured target environment or errant test to destroy the testing organization. Thus, if the test costs $ 50,000, the limitation of liability agreement caps the liability at that value.

To help address this issue further, most penetration testing companies also carry liability and errors and omissions insurance in addition to getting the limitation of liability agreement. Some common insurance levels are $ 1 Million, $ 2 Million, and $ 5 Million. In fact, some clients require their testers (or anyone else doing business with the organization) to get such insurance.

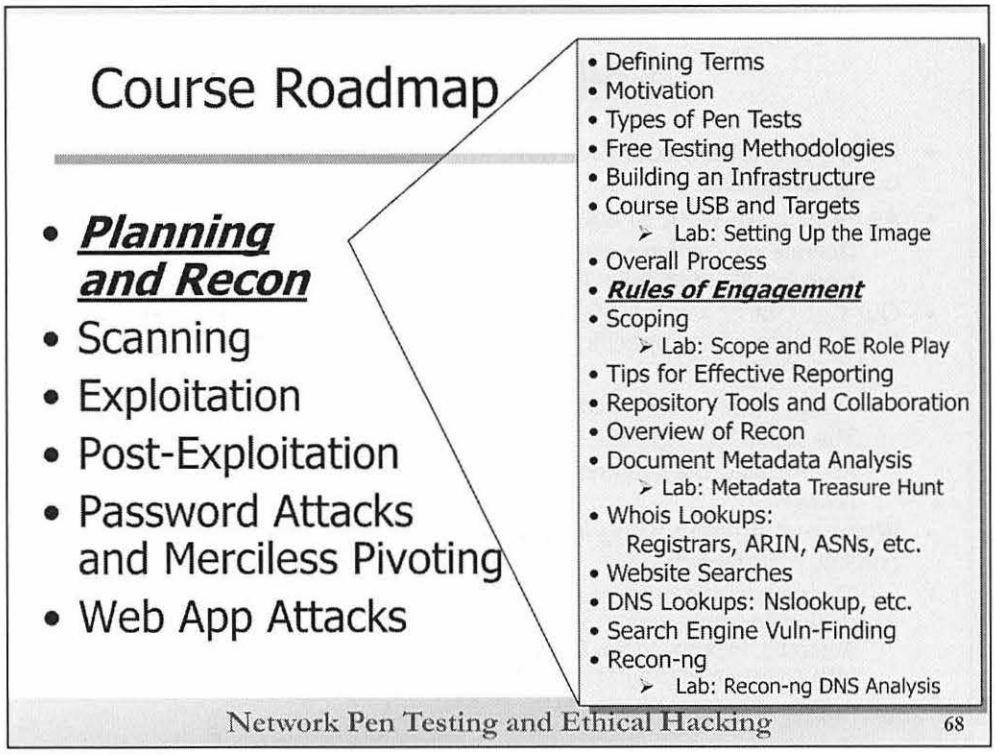
Many countries have instituted laws for dealing with crimes committed using a computer, so-called "cyber crime" laws. Not all countries have such laws, and indeed, attackers sometimes move to countries or operate through countries without such laws or where cyber crime laws are not enforced.

As penetration testers and ethical hackers, we want to make sure we carefully adhere to the laws of the countries in which we operate. In essence, conduct all your tests according to agreed-upon Rules of Engagement with a clearly defined scope. Attack only machines for which you have explicit permission, in written form, from both their owners and operators.

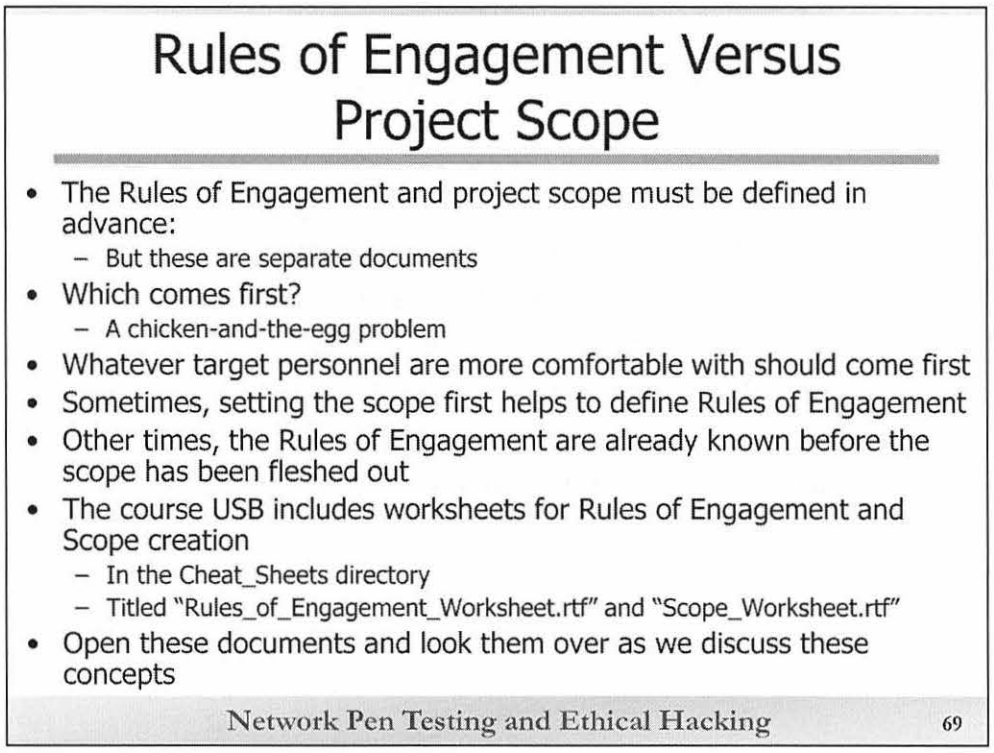
Your Permission Memo (the Get Out of Jail Free Card) is a helpful thing in ensuring that you have permission of the target organization that owns and operates the systems you will test.

It is important to note that the tester not only has to follow the laws where she is located, but also the laws in the country where the target machines are located. Furthermore, some countries (such as the United States) have indicated that any packets associated with a computer crime that traverse the country's borders fall within their law enforcement jurisdiction, regardless of where the packets originate or terminate. In other words, an attack from Germany against targets in Japan that traverse U.S. networks would be subject not only to German and Japanese law, but also to U.S. law.

Because some of the legal issues associated with testing in different countries can grow complex, we strongly recommend that you consult a lawyer when testing in a country where you haven't tested before. The lawyer can help explain the rules of the road to you, helping to make sure you don't run afoul of the law. For the most part, with a carefully documented Permission Memo, Rules of Engagement agreement, and Project Scope, you can operate in most countries legally without incident. Still, a lawyer's review can be helpful in establishing peace of mind and making sure you've taken any late-breaking legal issues into account when formulating your test plans.



Our next topic is Rules of Engagement, a set of practices that must be defined before a penetration test or ethical hacking project can begin. Both the people responsible for the target environment and the testing team must agree on these rules. Without proper Rules of Engagement agreed upon in advance, a penetration test or ethical hacking project could go seriously awry, resulting in devastating consequences for the target organization and the testers, including system downtime, financial damage, reputational damage, personnel firing, and possibly civil lawsuits or even criminal prosecutions.



The Get Out of Jail Free Card, limitation of liability agreement, and insurance help protect the testers legally. But, these documents must be shored up with a carefully decided Rules of Engagement memo that is documented in advance.

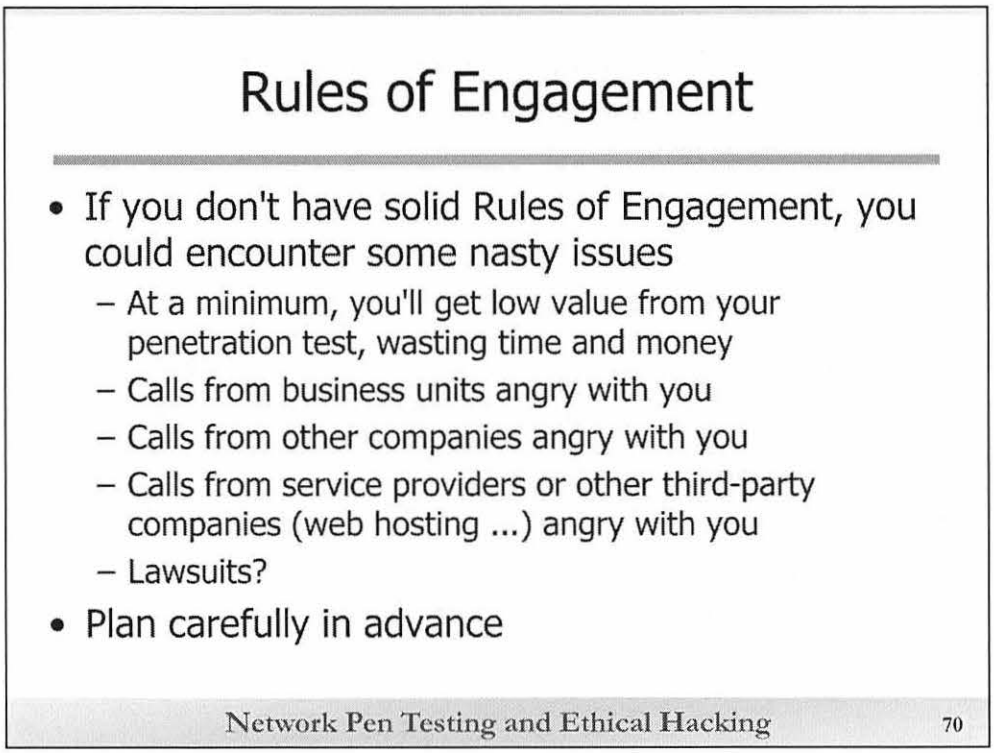
Some testers define the Rules of Engagement with a client before they devise a detailed scope of the test. That way, the target organization can have in mind the way the test will be conducted to help make decisions about what is in and out of scope.

Others reverse the flow, defining the scope before agreeing to Rules of Engagement so that they know what they will test and can then craft the Rules of Engagement around the given test targets.

Either approach is acceptable: defining the rules of engagement first followed by scoping, or scoping the project first and then defining rules of engagement. The important point is that both issues be covered in advance. In fact, these two phases may be iterative. That is, an initial Rules of Engagement is discussed, followed by a scoping conversation. Then, when the scope is agreed upon, the Rules of Engagement may be further tweaked, until the scope and rules line up appropriately.

The course USB includes worksheets for helping to define the Rules of Engagement and Scope of a penetration test. These documents, in the "Cheat_Sheets" directory of the USB, are in RTF format. You can open them up on your computer to follow along with the next section of the course as we explore the concepts more thoroughly.

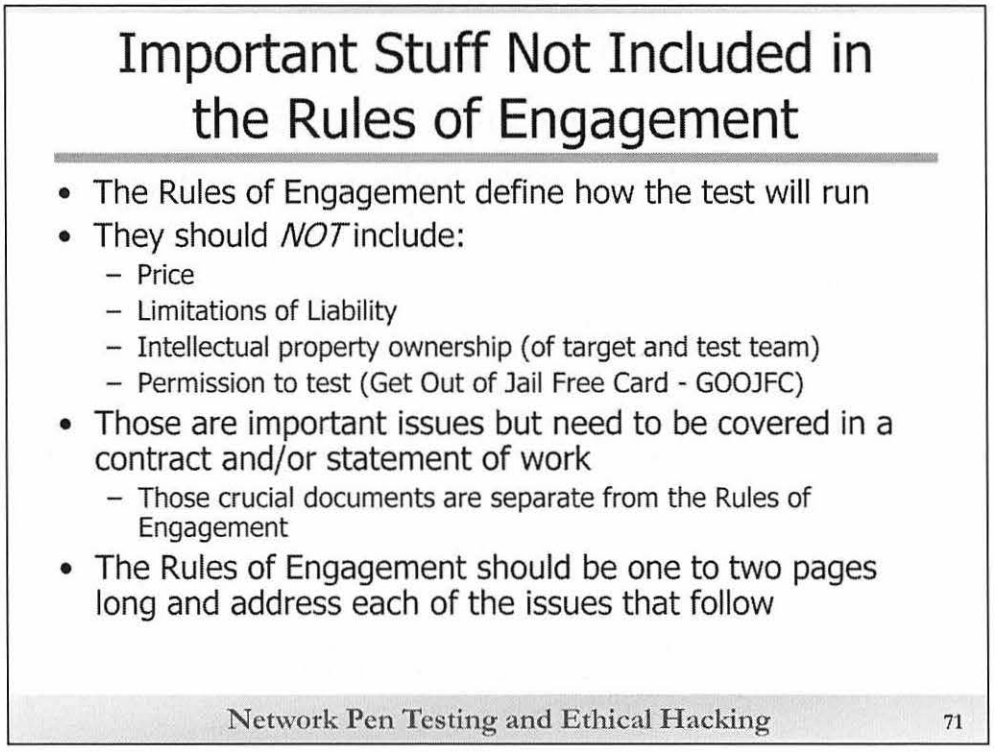
Please consider these worksheets as a starting point, which you can modify or add to in creating your own worksheets customized for your organization's testing.



If you don't nail down written Rules of Engagement in advance, you could run into significant trouble.

The minimum bad news you'll face without good Rules of Engagement is a wasted effort. That is, your penetration test results will be of low value, with little insight into your security stance. You'll have wasted your time and money.

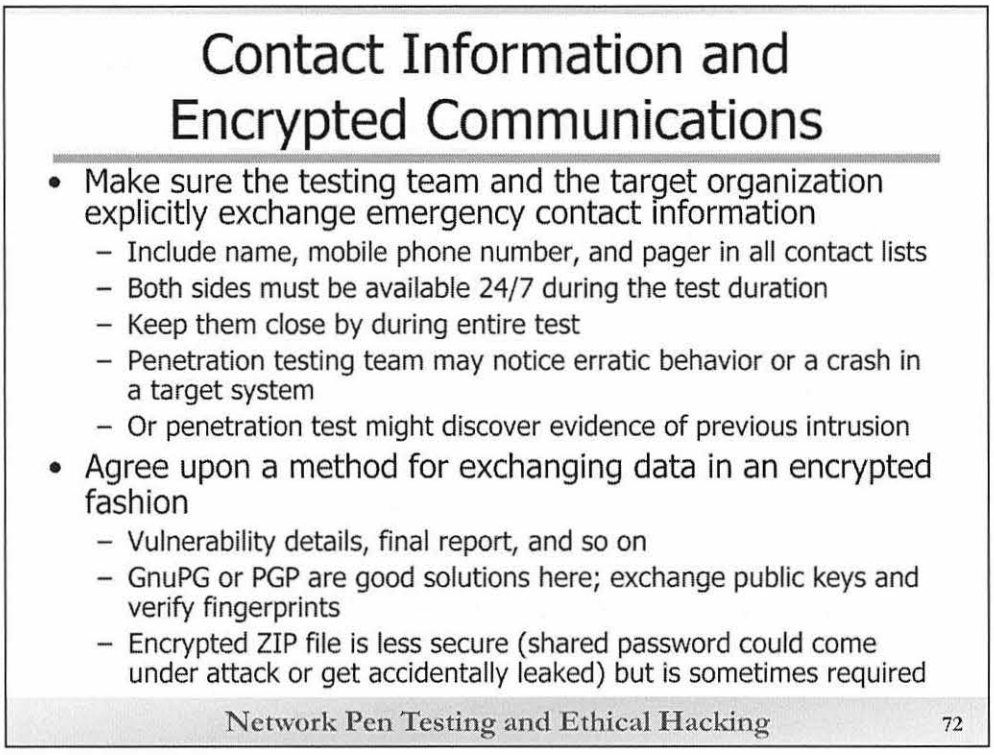
But the consequences could be far more dire than that! Without good Rules of Engagement, you could receive nasty calls from business units, other companies, or even your own service providers complaining that they did not authorize any test of your environment. In effect, someone attacked them, possibly giving them legal standing for a suit against you! Plan carefully in advance to minimize that risk.



The Rules of Engagement are intended to focus on how the test will be conducted and not to cover all the details between the target organization and the tester. The Rules should be kept short and focused; we recommend one to two pages, with pithy sentences dealing with each of the issues we'll discuss in this section.

In particular, the Rules of Engagement should not include the following items:

- **The price of the test service**: Such information shouldn't be in the Rules of Engagement.
- **Limitations of liability**: Who is liable and for how much money if a system or service is accidentally disabled, resulting in financial loss? This sticky issue is better handled in the contractual agreement instead of the Rules of Engagement.
- **Intellectual property ownership**: Do the resulting report and the methodology behind it belong to the target organization, the tester, or both? Again, this concept should be included in the contract, not the Rules of Engagement.
- **Permission to conduct the test**: The target organization should explicitly authorize all testing in writing, even if the tests are conducted by in-house personnel. This is the Get Out Of Jail Free Card, we discussed earlier.

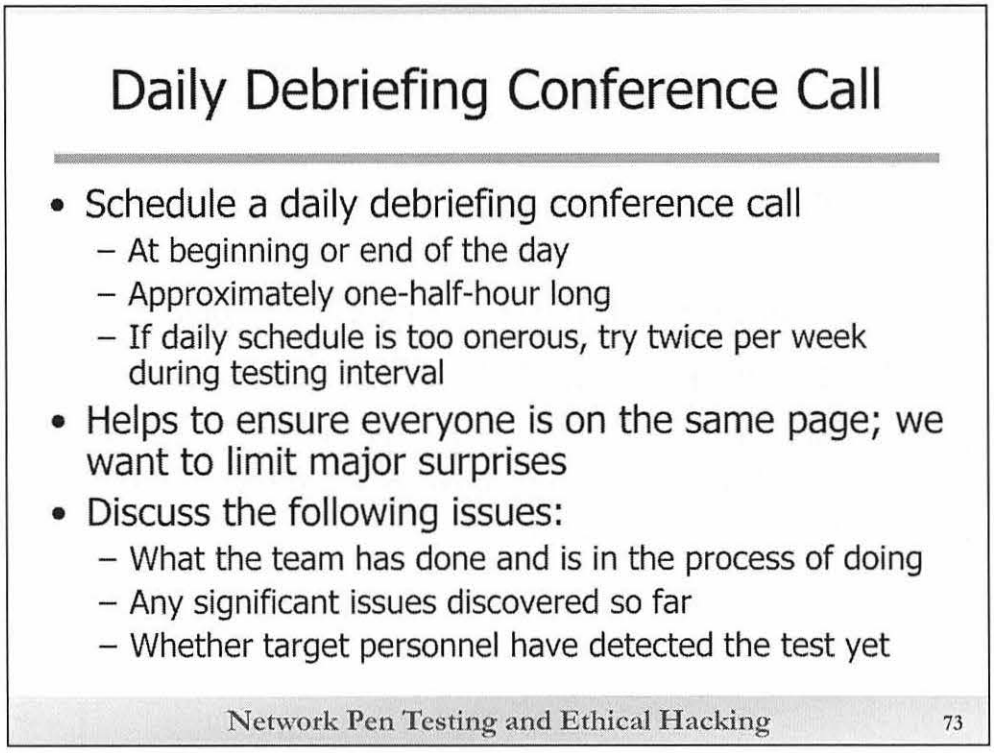


Another crucial element of the Rules of Engagement involves exchanging contact information between the testing team and the target organization. During a test, the testing team may crash a target, discover an urgent, high-risk vulnerability, or find evidence of a previous intrusion. In such cases, it may need to contact personnel immediately in the target company to report the issue.

Sometimes, the target company personnel needs to reach the testing team to verify that an observed attack is coming from the testers. What if an evil attacker starts hitting the target at the same time that the penetration testers begin their work? The target organization needs to contact the testing team 24X7 during the duration of the test. Therefore, both sides need to be available around the clock during the test.

After you identified points of contact, be sure to agree upon a secured method of communication regarding vulnerability details and the final report. The penetration testing team will be handling some sensitive data, which you may need to e-mail to the points of contact among target system personnel. You do not want to send this information in clear text. Instead, choose a suitable encryption solution. The free, open source Gnu Privacy Guard or commercial PGP are good solutions here. If you choose either, make sure to exchange public keys and verify their fingerprints.

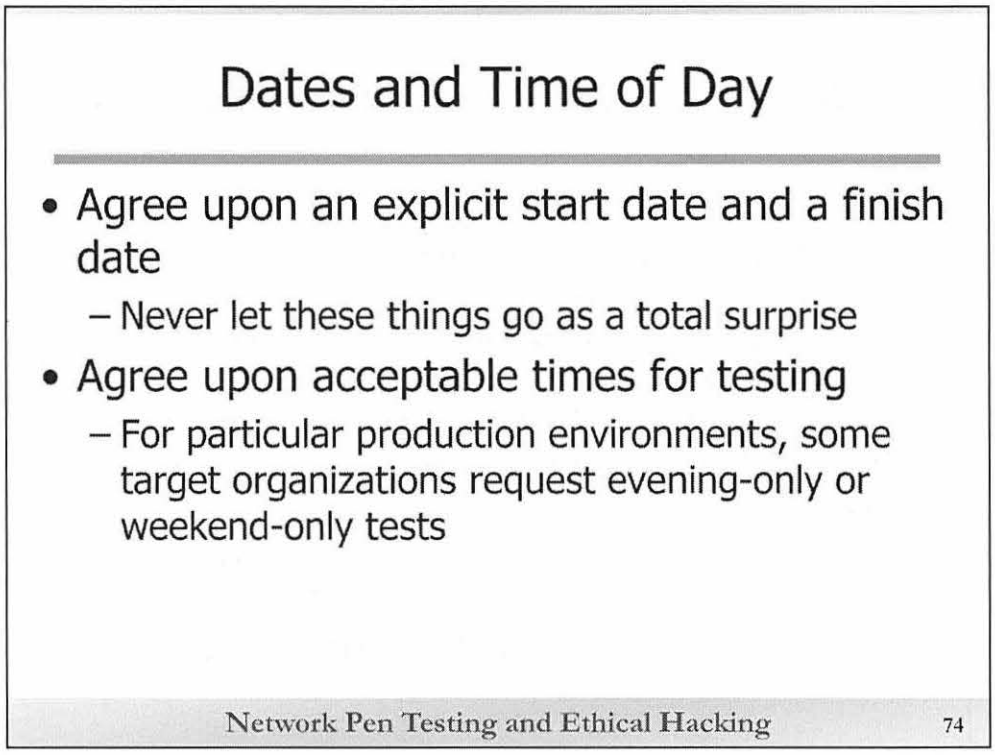
Encrypted ZIP files are less secure because the shared password used to protect the symmetric key in the ZIP file could come under attack. A bad person could steal the shared password, or it could leak if it is shared among too many people. For example, sometimes, target system personnel may be tempted to send the password for the ZIP file in e-mail, potentially exposing it. Alternatively, an attacker who gets the ZIP file could stage a brute-force attack against the password for the ZIP file. However, even though encrypted ZIP files are less secure, they are sometimes the only option we have, given that other encryption solutions are forbidden or are not accessible to target system personnel.



Another element that we find useful to define in our Rules of Engagement is to require a daily debriefing conference call. At the beginning or end of each testing day, schedule a brief session between the testing team and one or two representatives of the target organization. These calls help to make sure that everyone understands the progress of the test and to identify any issues early on.

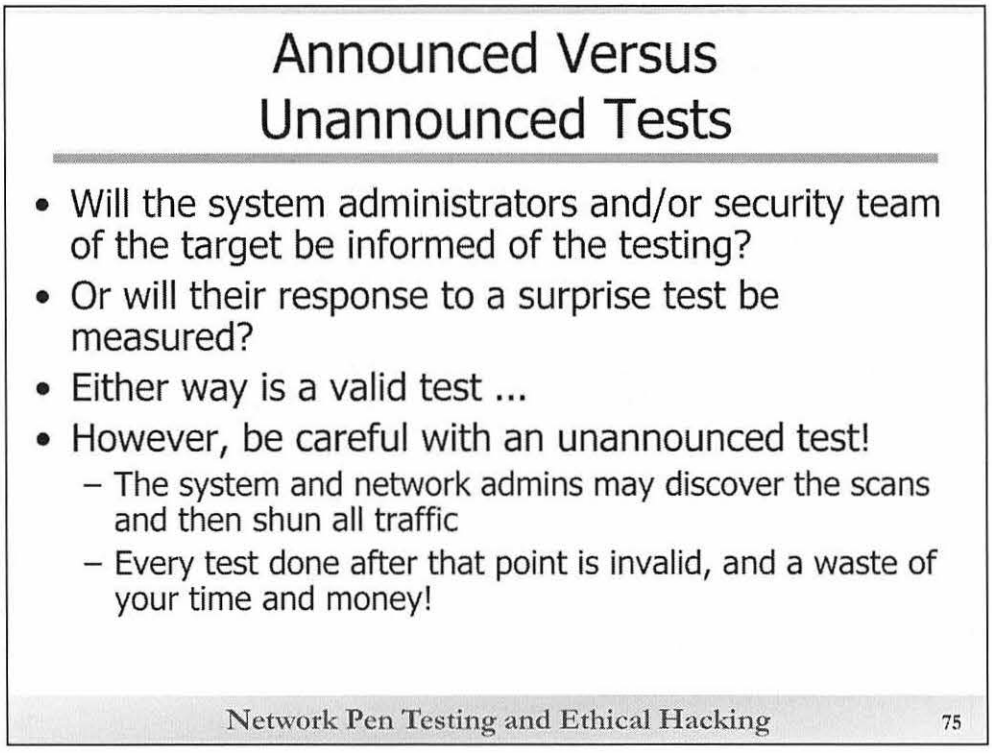
During the debriefing, the team should discuss what the team has done so far, and what they plan to do next. In addition, any major findings can be reviewed. Finally, the target organization can confirm whether its detection mechanisms (IDS, IPS, log review, and so on) have been triggered by the test.

If a daily call is too onerous given the busy schedules of target environment personnel, consider conducting a debriefing call twice per week during the duration of the test.



Of course, the Rules of Engagement should explicitly state the start and end dates of the test, as well as valid test times.

Some penetration tests and ethical hacking projects run around the clock, whereas others with more sensitive infrastructures and business needs require testing during off-hours or weekends only. When such off-hours tests are conducted by a third-party company, there is typically a slight additional charge for such off-hours testing, but it is usually quite reasonable. Also, such limited testing time windows require a longer total duration to complete the test.



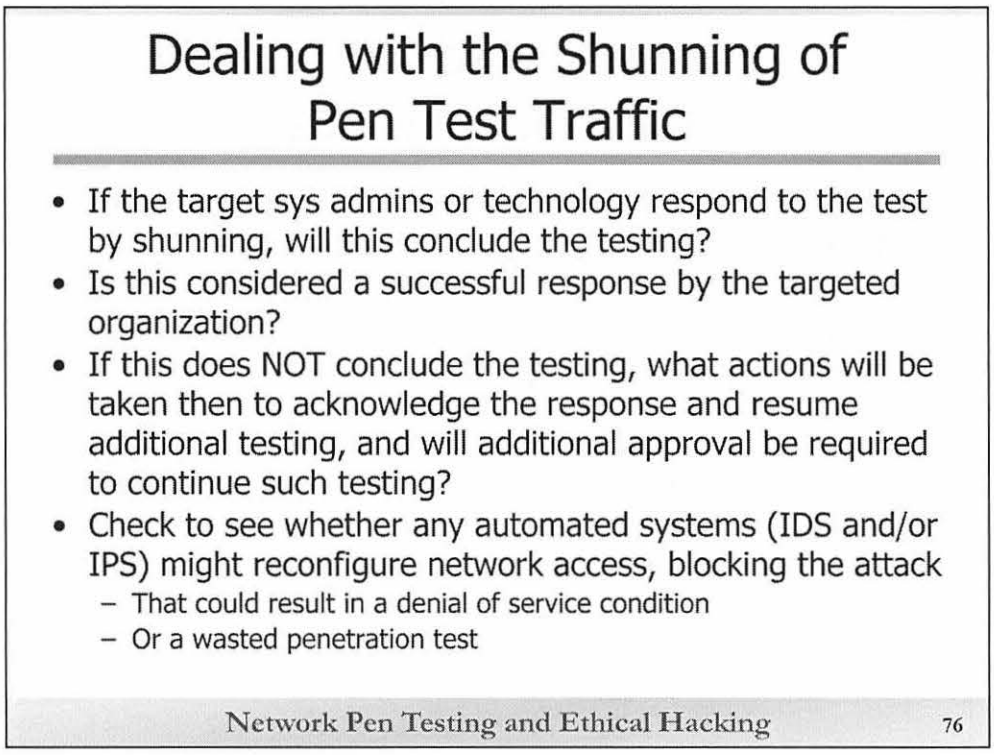
Here's another point that can cause controversy with some target organizations: Should the penetration test be announced to the people responsible for running the target infrastructure? Or will the test be a surprise to them?

Performing an unannounced test does have some advantages. First, if any of your admins are purposely running backdoors or side businesses on your servers, the testing team might find them during an unannounced test. If you announce the test, the admins will likely temporarily shut off their shifty activities during the test duration. Some penetration testers have discovered deliberate, sys-admin-planted backdoors during an unannounced test. Other testers have identified pay-for-porn services on target web servers that were run by the web administrator. Such findings are important results of a penetration test.

Secondly, an unannounced test gives you a chance to evaluate the detection and response mechanisms and processes of the target organization. Does information flow properly through the organization concerning a computer attack?

However, unannounced tests also have a downside. The system or network administrators might detect the attack and start shunning the traffic, blocking it from reaching the target systems. Then, any testing activities after the shunning is applied are invalid, a waste of time and money. If you do perform an unannounced test, make sure the system and network administrators are watched to verify that no shunning occurs.

Also, to prevent controversy about such tests, make sure that target personnel know that they could occur at any time and are just a normal part of the way the organization does business.



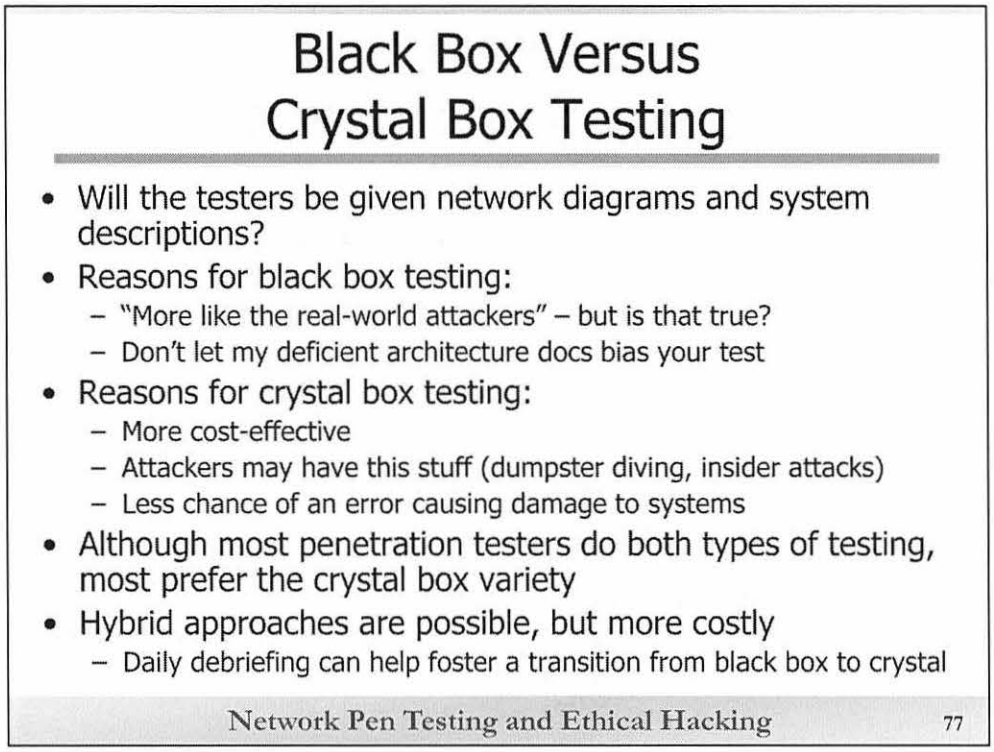
Another aspect to take into account with shunning involves what to do afterward. If manual or automated shunning occurs, will that conclude the test? And, if so, was the test successful?

Also, if and when shunning is engaged, make sure the target organization contacts the testing team to let it know and to determine which of the three options on the previous page you will utilize.

Some organizations have deployed automated shunning functionality. If an IDS spots an attack, it could reconfigure a firewall to automagically block it. Such functionality could lead to a denial of service attack, perhaps even inadvertently by the penetration testing team.

The Rules of Engagement must explicitly take into account any automated shunning functionality the target organization may have. Auto-shunning that is too widely applied could turn a run-of-the-mill penetration test into a denial of service attack. Or narrowly focused blocking would just render much of the test useless. To avoid these conditions, you should do one of the following, and document it in the Rules of Engagement:

- Have target organization personnel disengage shunning functionality.
- Include an exception for the source addresses of the testing organization.
- Announce the test, and get an alert when shunning starts so that it can be manually deactivated, allowing traffic from testers through.

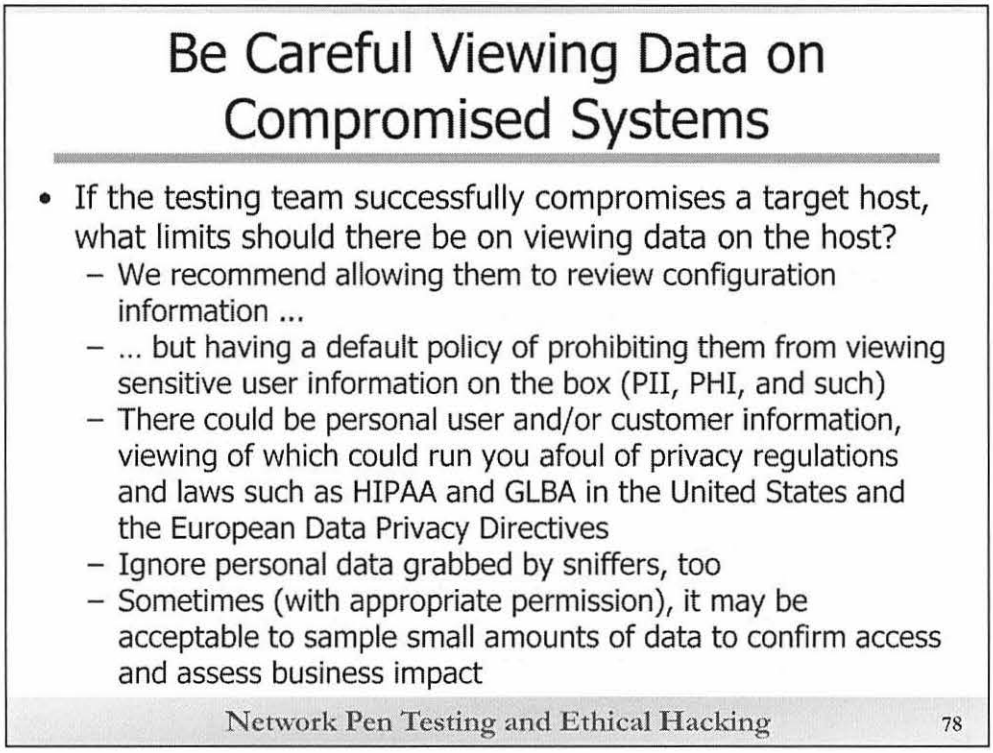


Here's another point of some controversy. Should the testing team be given a copy of the network diagram, listing hosts, topology, operating systems, and so on? Such crystal-box testing allows testers to see inside the target before launching any scans. Alternatively, the test might be a black-box engagement, where the testing team is given only a domain name and is expected to figure out the network topology and targets through reconnaissance and scanning.

The reasons some organizations say that they opt for black-box testing is that it is, "More like what a real-world hacker would see in our environment." Unfortunately, that's not necessarily true. A real-world attacker might have a picture of your network architecture, stolen from a dumpster, faxed from a duped employee, or swiped by a temporary employee. Working from a network diagram lets the penetration testing team analyze a worst-case scenario from the target's perspective.

A better argument for black-box testing is that some organizations are worried that a network diagram may bias the testers so that they miss items. Many network diagrams have significant limitations or errors, missing a lot of detail and sometimes overlooking hosts or whole networks.

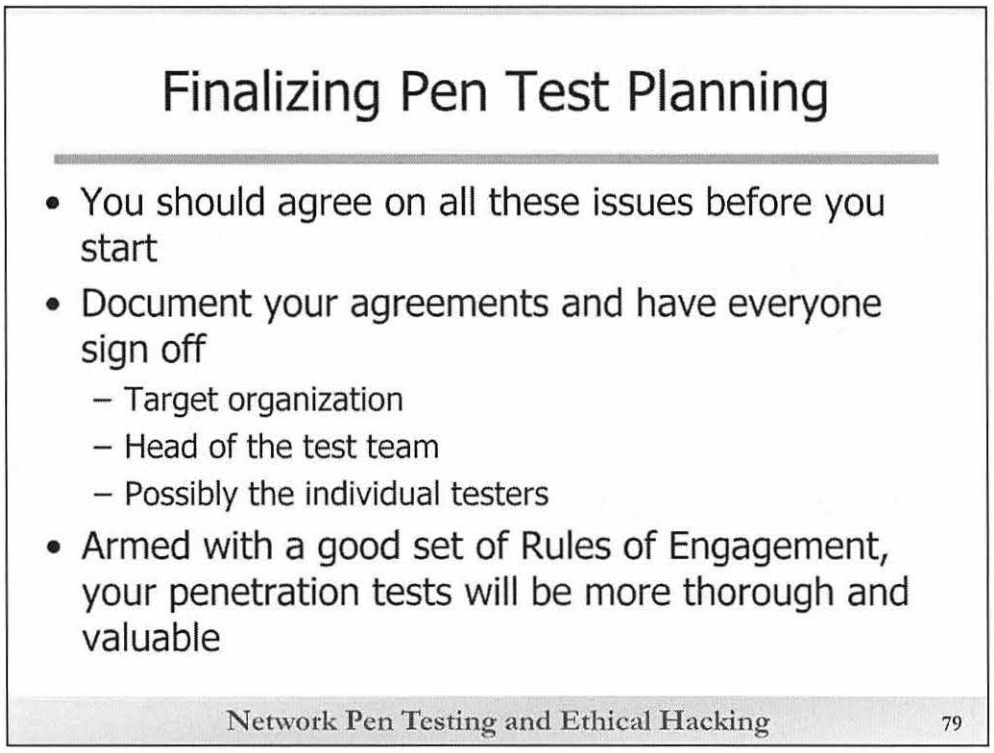
In the end, crystal-box testing tends to be more cost-effective because the attackers can quickly perform their reconnaissance and scanning. There is also a lowered chance of an error causing damage to an out-of-scope system. Although most penetration testers perform both black-box and crystal-box testing, the latter approach is usually recommended. Of course, you could take a hybrid approach in which the testing team starts out with nothing more than a domain name and performs detailed recon and scanning. Then after a week or so, the target organization provides a detailed diagram for further testing and analysis. When performing such hybrid testing, explicitly label in your report the elements the testers discovered and which items were given to them by the target organization. Hybrid tests usually take a little longer and therefore cost somewhat more. You can use the daily debriefing conference call on a black-box penetration test to start asking more questions, which could transition the project into a more crystal-box style test.



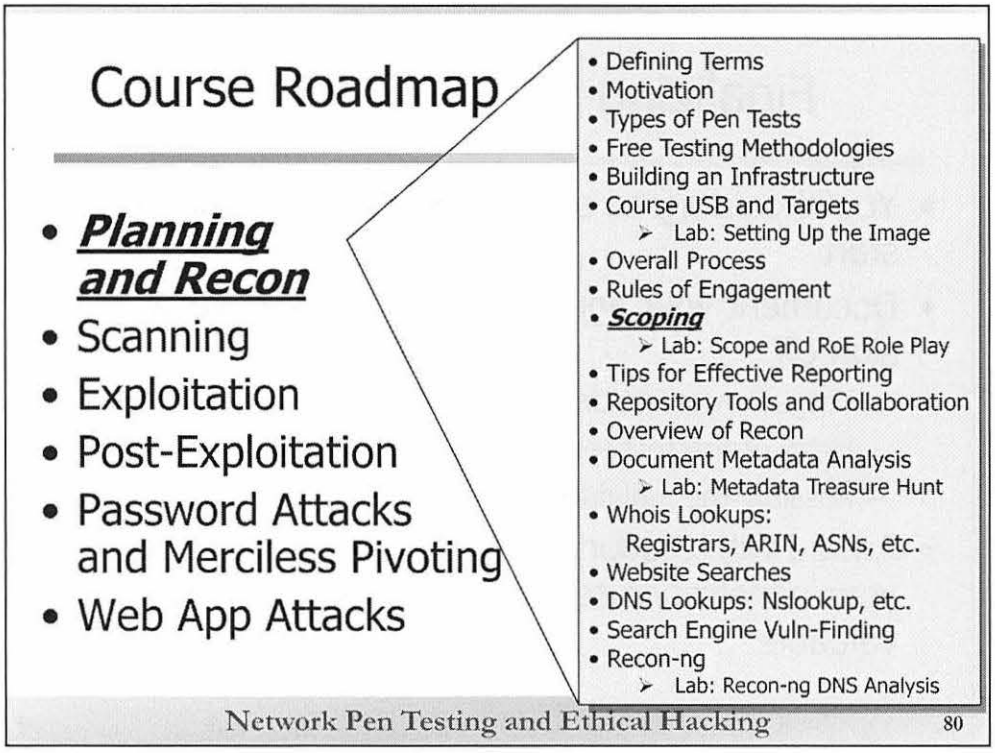
Here is a crucial issue to emphasize in your rules of engagement: If the team successfully penetrates a target, should it avoid viewing data on the target? Think about it: You might have sensitive employee or customer data on the target. A penetration tester has just gotten root on the box and might be able to view all that data. Only your Rules of Engagement prevent that from happening, so make sure they are clear on this topic. For some kinds of information, including healthcare and personal financial data, only duly authorized representatives with a need to know are allowed to see such information, under various regulatory initiatives such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm-Leach-Bliley Act (GLBA) in the United States and the European Data Privacy Directives.

We recommend that your default Rules of Engagement allow the testers to view configuration data from a machine, but to avoid looking at any customer and user data on the machine. Also, if sniffers are used on the compromised box, make sure the team explicitly documents that any personal data captured from the network will be ignored by the testing team.

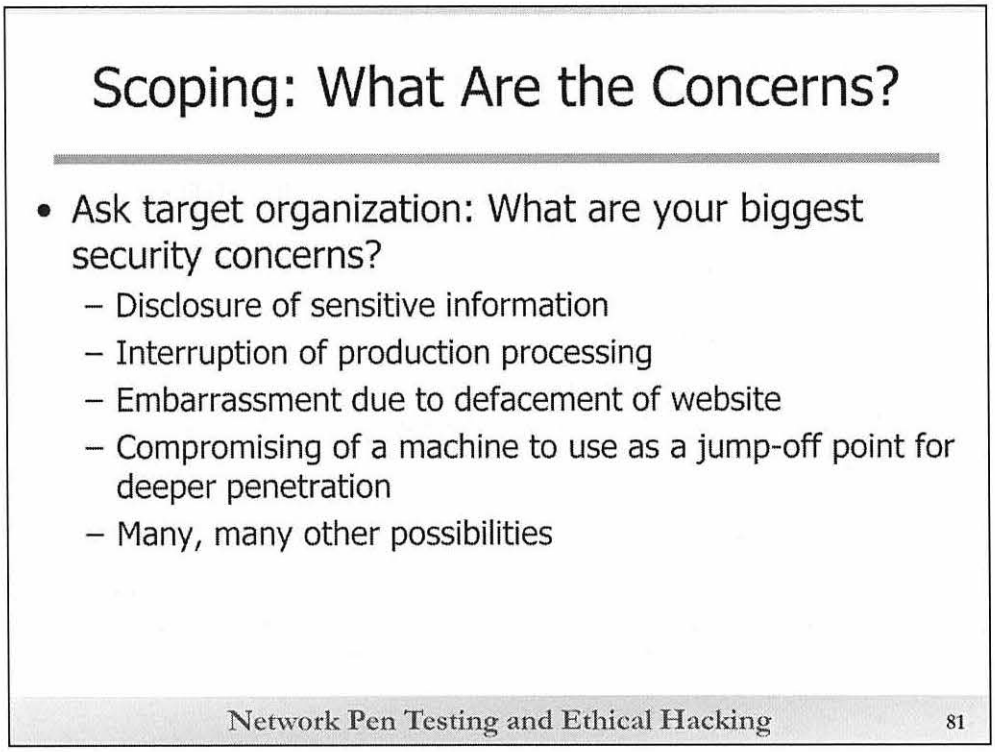
Although the default policy is usually to avoid accessing sensitive user data, in some tests, it may be appropriate to sample small amounts of sensitive user data to confirm access and assess business impact. Such access should be done sparingly and only with written permission from target system personnel. Also, try to focus more on counting the number of sensitive records you can access (based on file length, number of lines, number of files, number of rows in a database table, or other metric based on the type of data and the form of access you've gained) than on retrieving the sensitive content.



To conclude this segment of the course, make sure you have a solid, signed set of Rules of Engagement before you embark on any penetration test or ethical hacking project. You need to make decisions about these crucial issues in advance. If you do, you'll have a high-valued ethical hacking and penetration testing experience.

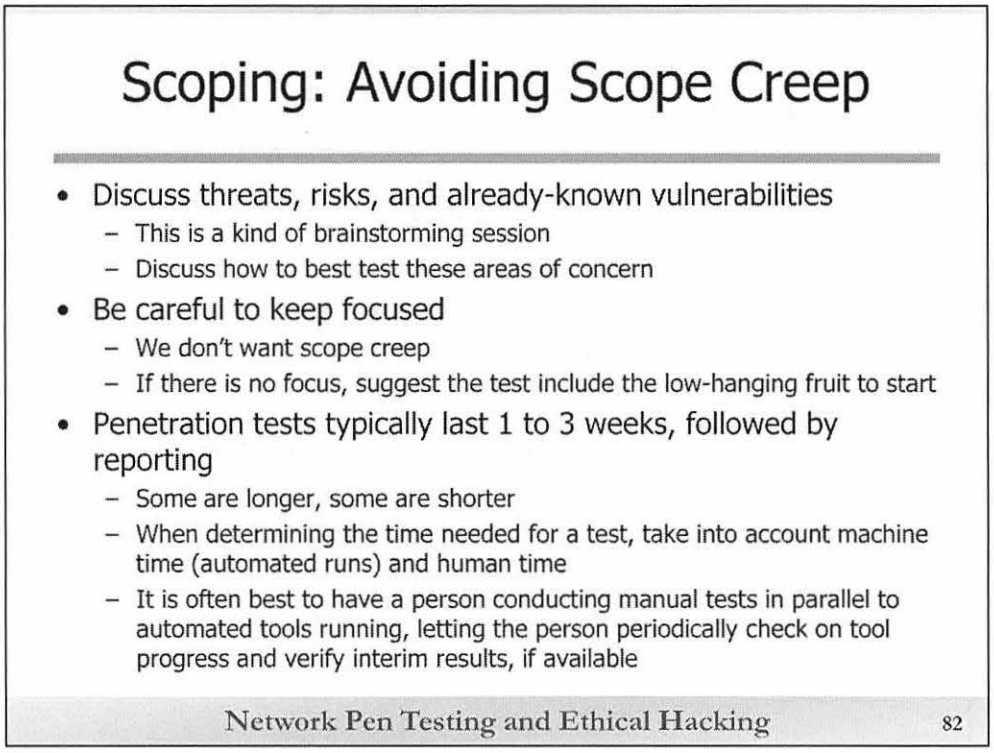


We'll now look at the scoping process, determining what should be tested and what should not be tested. In addition to determining individual target systems and networks, this scoping process will also look at some types of testing that may or may not be in scope.



To start out a scoping conversation, ask members of the target organization about their biggest security concerns. What worries them? Some example concerns could include the following:

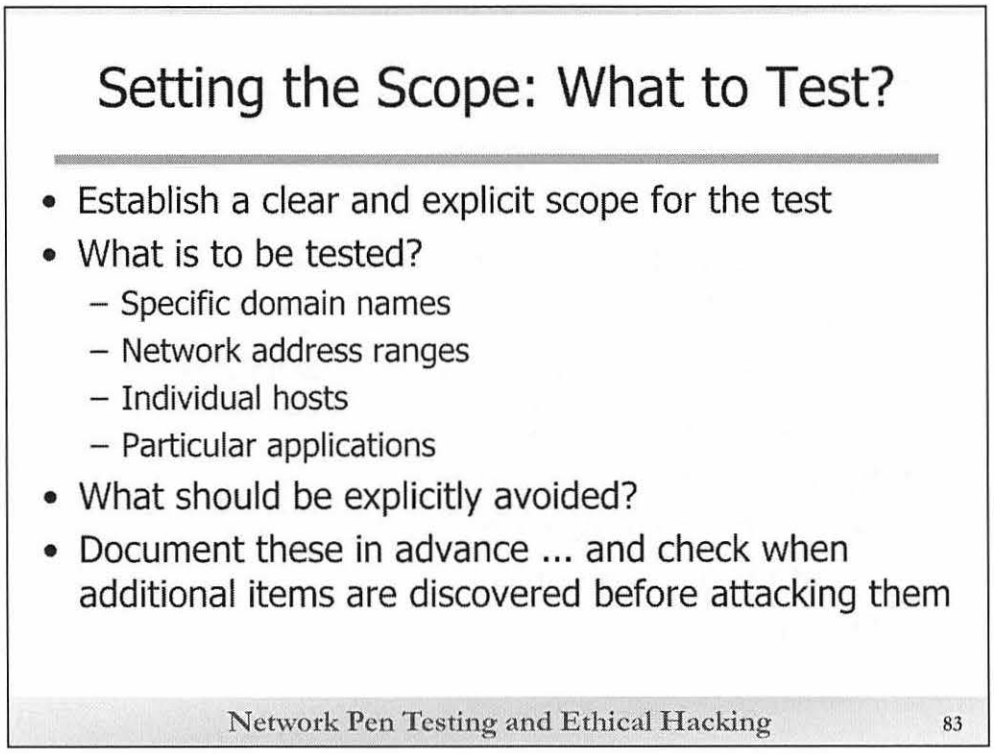
- **Disclosure of sensitive information**: Some organizations, such as financial services institutions, healthcare companies, educational organizations, and others store information that if stolen by an attacker would require public disclosure to comply with various laws. Furthermore, compromise of this information could result in the loss of customers, costing the victim organization dearly.
- **Interruption of production processing**: If certain critical systems crash, the target organization could be severely impacted, going out of business for the time it takes to restore the machines. Manufacturers, financial institutions, law firms, and many others are often concerned about this threat.
- **Embarrassment due to defacement of a website**: Some organizations would be badly impacted if their website is altered in an embarrassing way. Advertisers, government agencies, and others would be severely hurt if the public and their customers no longer trust them.
- **Compromise and pivot**: If an attacker takes over one machine on the target organization's perimeter and uses it for deeper access into the organization's network, the attacker could cause significant damage. Almost every type of organization faces this issue.



Note that the list of concerns on the previous slide is not comprehensive because many different organizations have widely different concerns, and some of them may be specialized based on their business. But this list should get the conversation started, allowing the person scoping the project to work with the client to brainstorm the most significant issues. Determining the primary concerns upfront can help narrow the focus of a test. Discuss threats, risks, and already-known vulnerabilities with the target organization's representatives.

It is vital to focus this conversation to determine exactly what needs to be tested. The last thing a tester wants is a blurry scope that could lead to scope creep. With *scope creep*, a misunderstanding of what should be tested leads the target organization to add more systems, target networks, target types, and types of testing to the test as it proceeds, a dangerous and costly proposition for a tester. If determining the focus is difficult because there are so many areas of concern, the scoper could suggest that the test focus on low-hanging fruit, those systems and networks that are of most concern and easiest to reach or compromise.

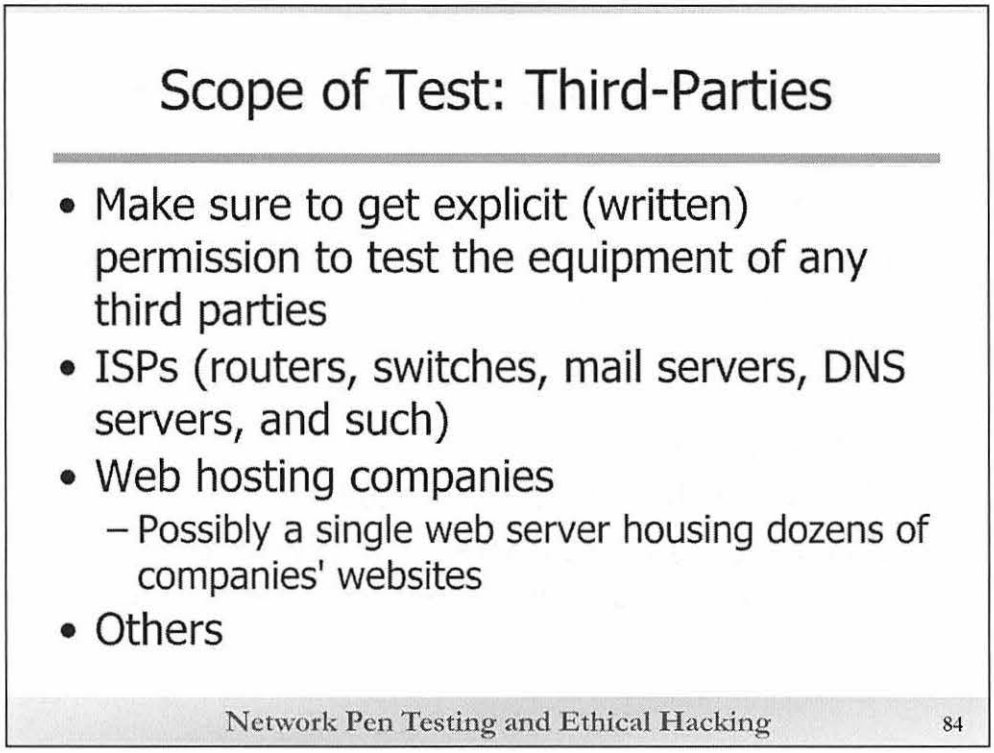
Most penetration tests occur over a 1–3 week span, followed by a reporting interval that could last 1 week or more. Some tests are longer than this typical time estimate, whereas others are shorter. When you scope the amount of time needed for a test, remember that the timing of a test must take into account automated tool running time and manual human hands-on time. For the most efficiency, we recommend that you have human testers working in parallel to automated tools running. The tools measure various issues in the target environment, whereas the humans conduct detailed analysis of other areas. The humans can then check to make sure that the automated tools function as wanted, and the humans can verify interim automated tool results as they are produced.



One of the most important elements to include in the project scope is a succinct statement of what is to be tested. Spell out explicitly those domain names, network address ranges, individual hosts, and particular applications that are included in the test.

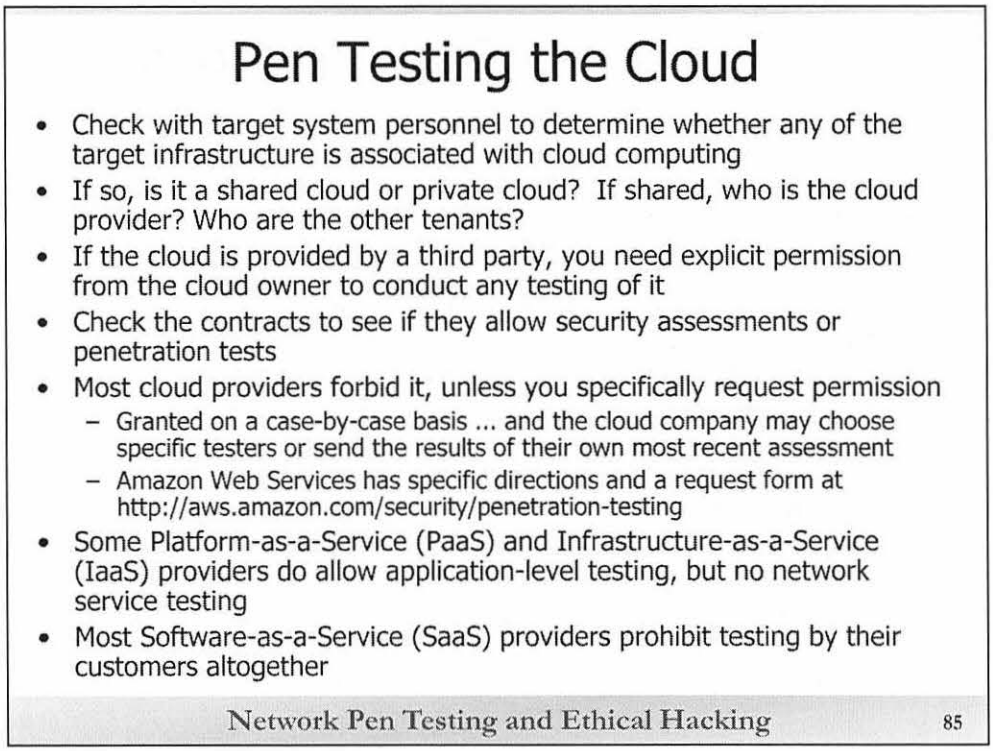
Also, if there are particularly sensitive elements of the organization that should not be tested, explicitly spell out that list of off-limits machines.

While the team is performing the test, they may discover additional systems within network address ranges or domain names that weren't considered in the original formulation of scope. Make sure the Rules of Engagement direct the testing team to check with the target organization before testing any other machines outside of the original scope that are discovered through reconnaissance or scanning after the test starts.



Many major organizations today use at least one third-party to manage at least part of their computing and network infrastructure. Some companies outsource this altogether, with their infrastructure actually being owned by the third-party itself. Examples include Internet Service Providers, whose routers, switches, mail servers, and DNS servers may fall into the scope of a test. A particularly significant concern is associated with third-party web hosting companies. Sometimes, a single web server may house the web presence for a dozen or more companies. If one of those companies contracts a penetration test without letting the web server owner know about it, there could be significant trouble!

If any third-party owned or managed systems are included in the scope, make sure to get written permission from these parties before the test begins. The target organization is responsible for getting this permission, and the testers are responsible for making sure the target organization does this.

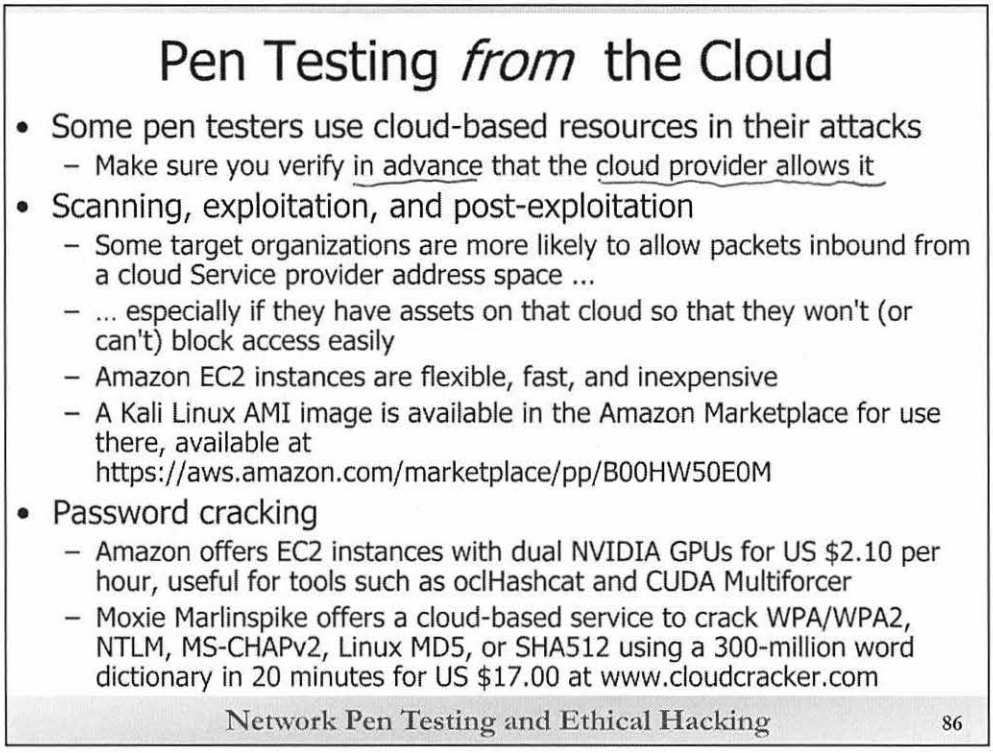


Cloud computing deployments are rapidly increasing, and penetration testers are sometimes called upon to test cloud environments. Be careful in getting permission to test such environments.

Ask target system personnel whether the cloud is shared among multiple enterprises acting as tenants on a third-party-provided infrastructure, or whether this is a private cloud run only by the given target organization. If this is a shared cloud operated by a third-party, you must get permission to test from the cloud provider.

For third-party provided clouds, check the contract to see if it allows or explicitly forbids security assessments or penetration tests. Most cloud providers explicitly forbid such tests, unless you specifically ask for and receive written permission, which is granted on a case-by-case basis. Some cloud providers will simply provide the results for their most recent assessment to their customers upon request, or choose a specific set of testers to honor a client's request for a needed test. Amazon Web Services has been quite forward-looking here, with directions for penetration testers and a specific AWS Vulnerability / Penetration Testing Request Form available online.

We've seen many cloud providers that offer underlying virtualized systems, in a Platform-as-a-Service (Paas) or Infrastructure-as-a-Service (IaaS) arrangement, allow customers to perform application-level testing (such as web app testing), but prohibit them from testing for underlying network and listening service flaws. Many Software-as-a-Service (SaaS) providers prohibit customers from conducting their own tests whether at the network/service layer or the application layer. Make sure you get appropriate written permission before launching any tests.



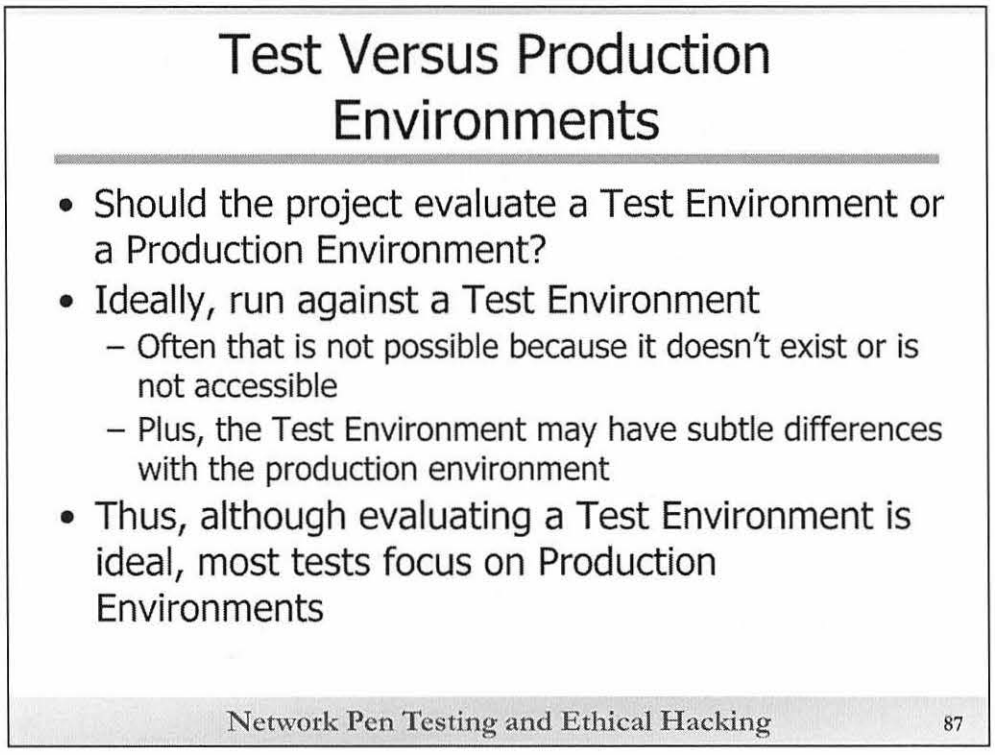
On a related note, some penetration testers rely on cloud-based resources as their attacking platforms. They often rely on such systems in two different ways: as sources of attack network-based attacks and as computing farms for password cracking attacks. Regardless of how you use cloud-based resources in your penetration tests, make sure that your use is allowed under the cloud service providers terms of use. Otherwise, you could be in breach of contract, suffer from a lawsuit, or even face a criminal investigation.

Using cloud-based systems as a source of attacks, penetration testers can launch scanning, exploitation, and post-exploitation activities from virtual systems located in the cloud. This could benefit the penetration testers from a cost perspective, running large-scale scans from essentially rented systems for less than US $ 1 per hour. In addition, it opens up other benefits. Many target organizations will be far more likely to allow inbound access from addresses owned by a cloud service provider. This is especially true if the given target organization hosts resources in that same cloud. Such organizations likely won't (or can't) then filter attacks coming from that cloud IP address space.

Amazon EC2 instances are flexible, allowing a penetration tester to run a variety of different operating system flavors to use in penetration testing. The Kali Linux penetration testing suite is available on Amazon EC2 as a prebuilt AMI image, ready to use at no extra charge beyond the computing costs from Amazon, available at the URL shown on the slide.

Alternatively, some penetration testers use cloud instances to crack passwords. Amazon's EC2 instances now offer an option of GPU support (with dual NVIDIA GPUs) for a premium price of US $2.10 per hour, still a reasonable price for access to such computing power. Tools such as oclHashcat and the CUDA Multiforcer, which we'll discuss in more detail in 560.5, could run on such EC2 instances.

Moxie Marlinspike runs a cloud-based cracking service that runs on a remote cloud with the equivalent of 400 CPUs clustered together. It can crack various forms of challenge/response authentications sniffed from the network, as well as some password hash types, including WPA/WPA2, NTLM, MS-CHAPv2, Linux MD5, and Linux SHA512. The service includes a 300-million word dictionary, which it runs through in 20 minutes for US $17.00. Using a single computer would take weeks to get through such a large dictionary.



One of the important issues to discuss before launching an ethical hacking or penetration testing project is whether to evaluate a Test/Quality Assurance Environment or a Production Environment. Some organizations have full-blown test environments in which changes and business functionality are evaluated before being applied in production. Some organizations have a third environment as well, often called Development, in which new code is created and scrubbed before being moved into the Test Environment.

Ideally, we'd like to conduct our penetration tests and ethical hacks against a Test Environment. After all, it was designed for testing. Any system or service crash conditions that are discovered in a Test Environment shouldn't impact production processing, which is certainly good news.

Unfortunately, we often cannot achieve this ideal. Most organizations do not have a fully functional Test Environment, forcing us to conduct penetration testing and ethical hacking against the Production Environment. Even for those organizations that do have a Test Environment, these systems often have both major and subtle differences with the Production Environment that might invalidate our findings.

Thus, while testing against the Test Environment is ideal, the vast majority of real-world penetration tests and ethical hacking projects occur against Production Environments. We should strive for Test Environments, but face this reality of most testing.

Internal and Pseudo-Internal Access

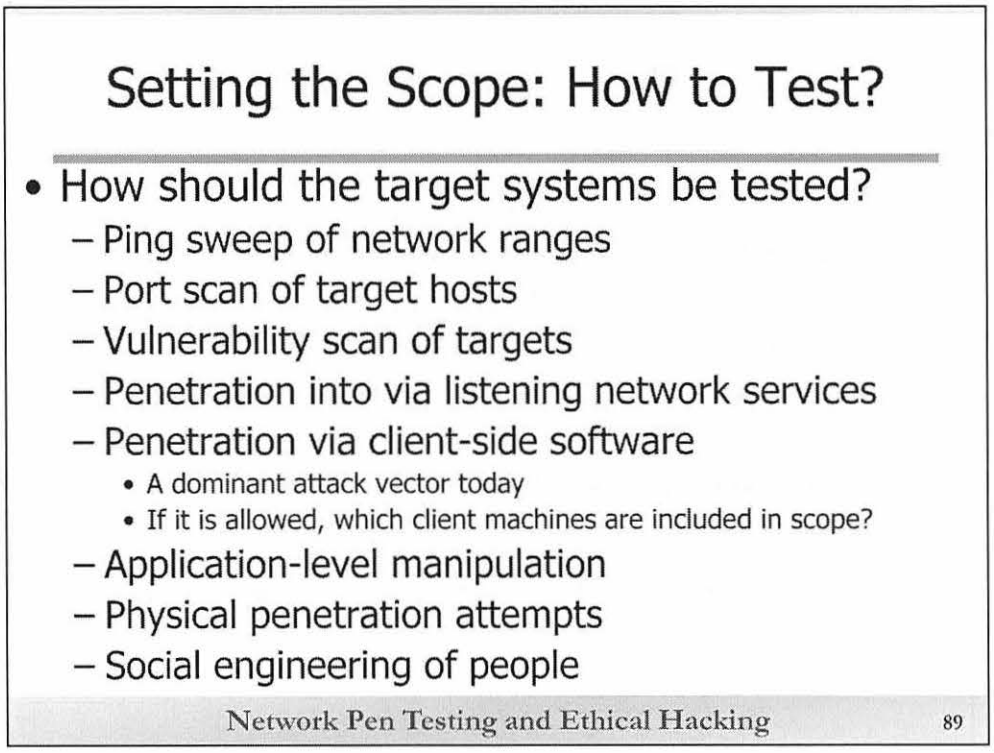
- Many penetration tests occur across the Internet
- But what about inside vulnerabilities?
- Methods for testing from the inside:
  - Team travels onsite and is granted access
  - Team travels onsite and tries to sneak in
  - Team travels onsite and looks for wireless
  - Team gets VPN or SSH access internally
    - Requires a lot of coordination and support
    - But you could ship target personnel a preconfigured laptop, ready to go, installed on their internal network or on their DMZ, which uses a cron job to establish a reverse SSH tunnel to you to implement inbound access
    - Some companies sell preconfigured systems designed to maintain a persistent reverse SSH tunnel back to a pen tester, preloaded with tools
    - Form factor of a large wall plug, running Linux on ARM processor, with Ethernet and/or wireless
    - Pwnie Express' PWN Plug R2 product starts at approximately US $1095 for Wi-Fi/4G/GSM version

Network Pen Testing and Ethical Hacking 88

The majority of penetration testing occurs across the Internet against publicly available servers and network equipment. However, what would an inside attacker see? To get this view, the penetration testing team would need inside access to scan and explore the internal environment. There are many ways to grant such access, such as traveling onsite, attempting physical break-in, looking for unsecured wireless LAN access, or using a VPN for remote internal access across the Internet. A particularly promising approach involves sending target personnel a preconfigured laptop set up with a cron job that periodically tries to make a reverse SSH connection across the Internet to one of your systems, setting up an SSH tunnel. Then, you can use that outbound SSH access to gain inbound access to your preconfigured machine and the rest of the network. This approach, of course, requires the organization to allow outbound SSH tunnels, which is typical in many organizations today.

Some companies sell penetration testing systems preconfigured to ship to target system personnel so that they can easily hook the machine up in their network, granting remote access to penetration testers. Many of these systems have the form factor of a small Linux machine. They tend to run Linux on an ARM processor, loaded with tools and the capability to create a reverse SSH tunnel back to the penetration tester who configures the system. Some have wireless interfaces as well, looking for Wi-Fi or even 4G/GSM cellular. A company called Pwnie Express offers its PWN Plug R2 product, which starts at approximately US $1,095 for the Wi-Fi/4G/GSM model, which also includes a USB interface to provide additional storage or to move tools to the system via a thumb drive.

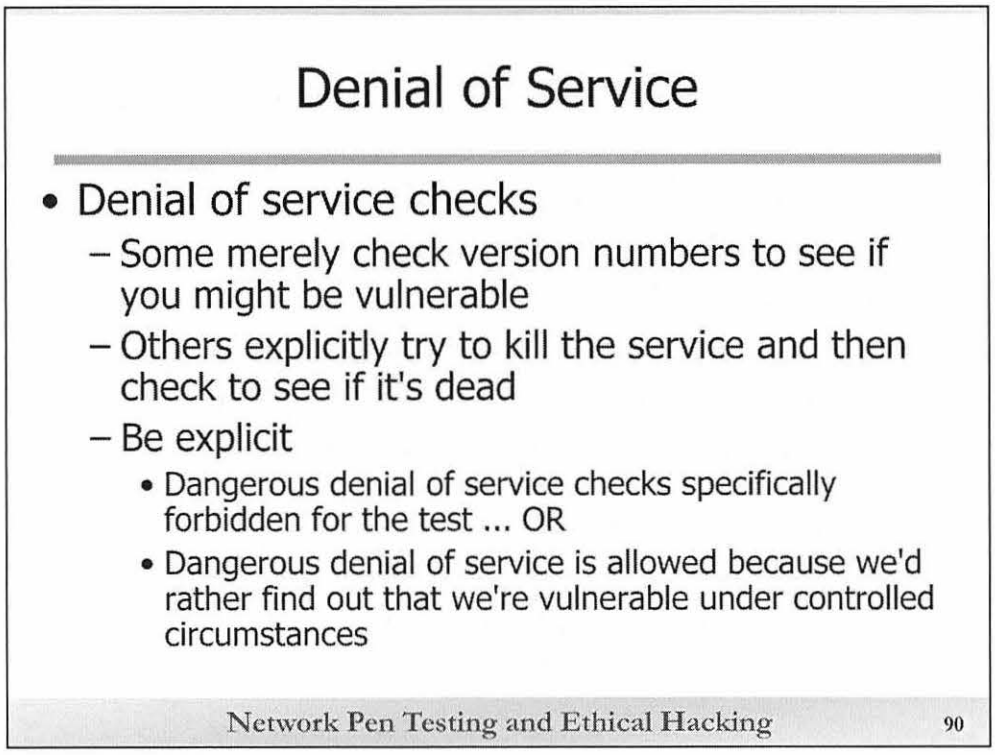
Make sure the scope clearly spells out how you will do internal testing, if at all.



Beyond what should be tested, the scope should specify the level of testing that should occur. Will the test merely be a network scan for targets and vulnerabilities, which would include a ping sweep, port scan, and vulnerability scan? Or should the testers go further and actually penetrate the target systems, getting access (such as a remote shell) on the targets if possible? If penetration is allowed, should it focus on listening network services, or will client-side software exploitation be allowed as well? Client-side software exploitation is a dominant attack vector today. If client-side exploitation is allowed, which client machines will be included in the scope?

Will the test include any application-level or client-side web component testing? Will it include physical penetration attempts, with the team trying to walk into an environment? Should social engineering, which involves duping human beings typically via the telephone, be attempted?

Your Rules of Engagement should specify each element on this list that is included in the scope.

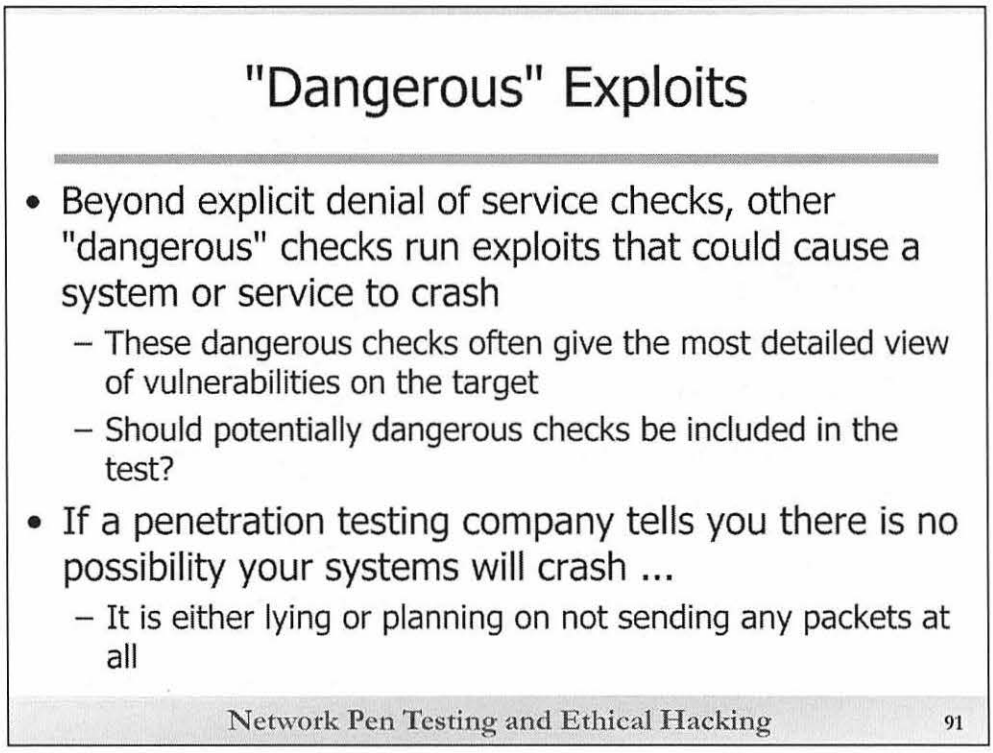


Another big decision to make during scoping a project involves denial of service. Some denial of service checks merely measure the version number of the target service and look it up in a list to determine if it is known to be subject to a denial of service attack. Such tests aren't very dangerous and should be included in the scope.

The other kind of denial of service checks are more dangerous. They first verify that a service is running on the target. Then, they launch the denial of service attack. Then, they measure whether the service has died. If it has, the target machine is indeed vulnerable. But the target service is also dead, a potentially devastating impact on a production environment.

Based on this concern, most penetration test scoping agreements explicitly rule out all dangerous denial of service checks. However, if a service can be crashed by a remote attacker, it may be better to learn about it during the controlled circumstances of a penetration test rather than waiting for a vile script kiddie to hit it in the middle of the night during a holiday weekend or some crucial crunch-time processing. Most organizations forbid dangerous exploits during a test, whereas others do allow them.

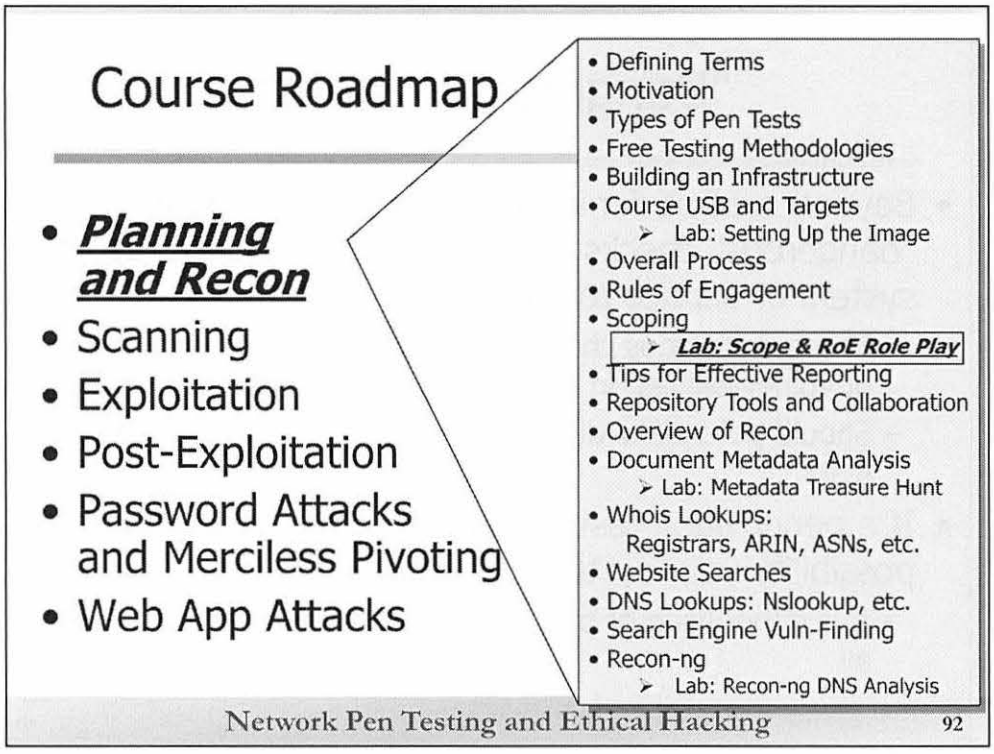
Therefore, you must be explicit about whether you want dangerous denial of service checks in your test's scope.



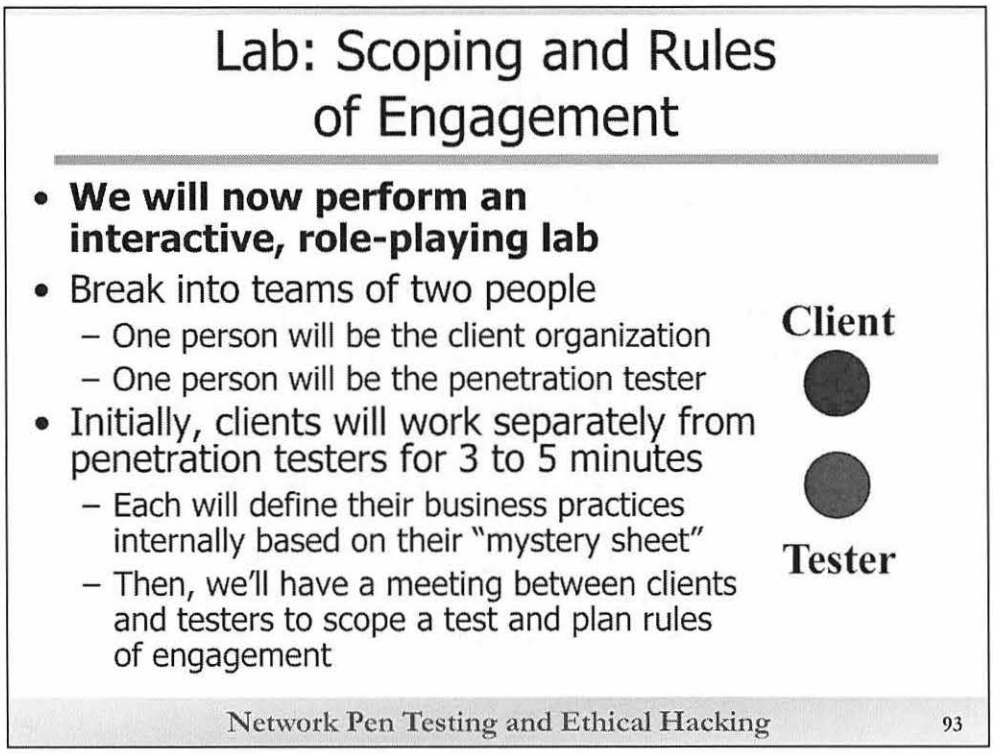
Beyond the checks that are explicitly designed for denial of service, other classes of checks could potentially crash a target service or system. These potentially dangerous exploits often try to spawn a shell for the attacker by clobbering a buffer overflow exploit on the target. However, such techniques manipulate memory and therefore sometimes render a service inaccessible. Unfortunately, running these potentially dangerous checks is often the best way to get detailed information about whether a target is vulnerable. Running a penetration test without dangerous checks gives less information about the target.

So, while scoping a project, target environment personnel and the testing team must decide should potentially dangerous checks be included in a test or avoided? Include a statement in your project scope documenting your decision.

Another important thing to keep in mind is that during a penetration test there is always some residual risk that the test could cause a system or service to crash, even if you explicitly disallow all dangerous checks. Even with no dangerous checks, the tester's tools will generate some strangely formatted packets that could cause a particularly feeble or unusual system to crash. Or the additional traffic load could push a highly loaded network's performance over a cliff. Therefore, if a penetration testing company ever tells you that there is no possibility of its tests crashing your systems, it is either lying to you or planning on not sending any packets at all. You should respond to such a company, "Ummm ... I know there is always a possibility of a crash in a target system. Tell me what you will do to minimize that chance, and your processes for detecting such a circumstance." In subsequent sections of this course, we'll look at scripts for monitoring services to make sure that they continue to run while a test is underway.



Because establishing firm Rules of Engagement and scoping a project properly are so important, we will now conduct a lab on the topic. In this lab, class attendees will formulate questions and responses associated with a penetration-testing Request for Proposal (RFP). The goal of this lab is to help testers get a feel for the kinds of questions and answers that should be asked to scope a project and to determine the Rules of Engagement, sharing information about risks and benefits between clients and testers.

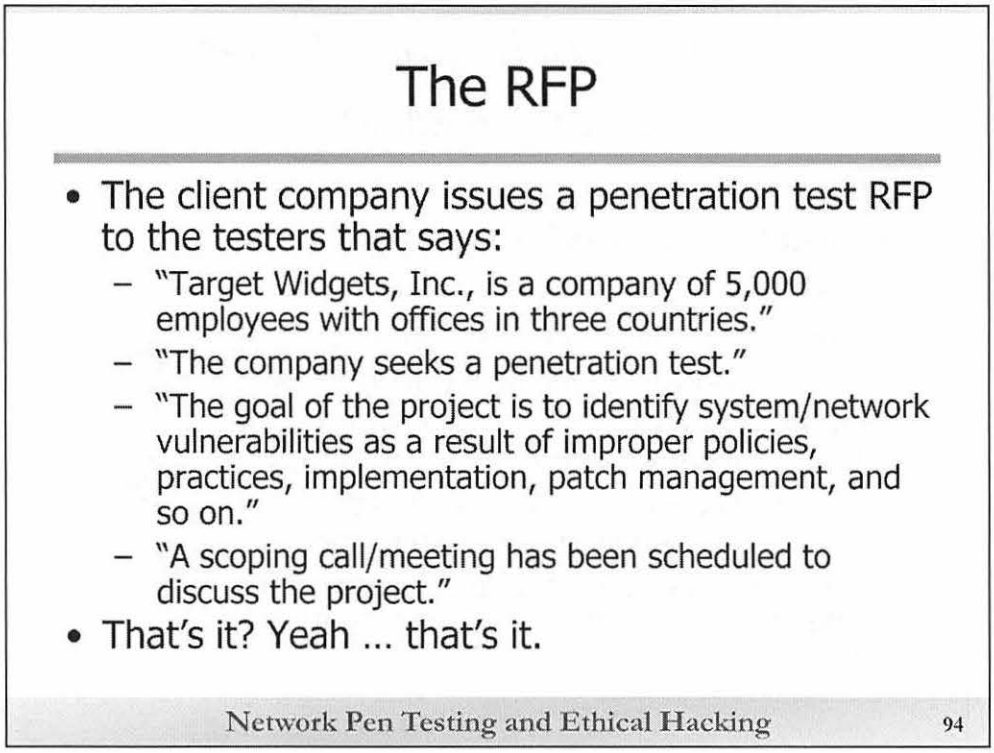


For this interactive, role-playing lab, we build Rules of Engagement and a scope for a sample penetration test by doing some interactive role playing. Your instructor will break you into teams of approximately two people. (If you are taking the class via SANS vLive, your instructor will play the part of the client during the lab; if you are taking the class via SANS OnDemand, you'll receive both the client and the pen tester mystery sheets for which you can formulate the appropriate questions and answers for the scope and rules of engagement.)

The pen test client group is an organization that has issued an ambiguous RFP for a penetration testing project. The pen tester will ask the client to get more information about the project for scoping. Furthermore, both sides need to agree on the Rules of Engagement. The testers should also describe the risks and recommended approach for the project to the client.

You could view this lab as an external penetration testing company getting information to prepare a bid for a client, or you could view it as a set of internal testers preparing for a test of their own organization by discussing the test with the target business unit. Either approach is acceptable.

The clients and testers will work separately for about 3 to 5 minutes, reviewing the details of a mystery sheet that each person will receive. The client sheet will describe the client's business with information that the testers should ask about. The tester sheet will describe the background of the testers to help them plan their approach.



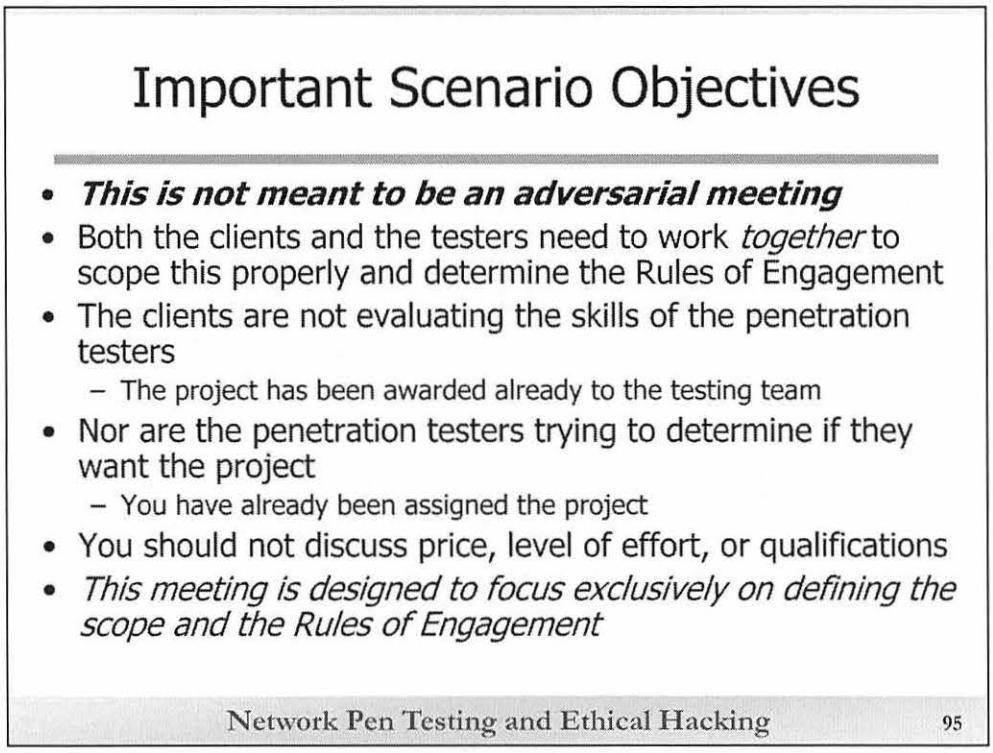
The RFP issued by the client company provides rather limited details about the test. Quite often, in the penetration testing business, testers are presented with limited information about a potential project upfront, making the scoping task vitally important so that both the client and the testers are on the same page for the test.

The RFP includes the following facts:

- The test will be performed for Target Widgets, a manufacturing company with 5,000 employees and offices in three countries.
- The company wants a penetration test (either from an outside penetration testing company or from a technical group within the company; either is a valid approach for our purposes here).
- The goal of the project is to find security flaws that may have resulted from improper policies, practices, implementation, patch management, and so on.
- That's it. The RFP includes no further information.

For the lab, the testers will receive a tester's sheet, whereas the clients will receive a client's sheet. The instructions on the sheet will provide more detail about the given organization, as well as certain items to cover during the scoping and Rules of Engagement meeting.

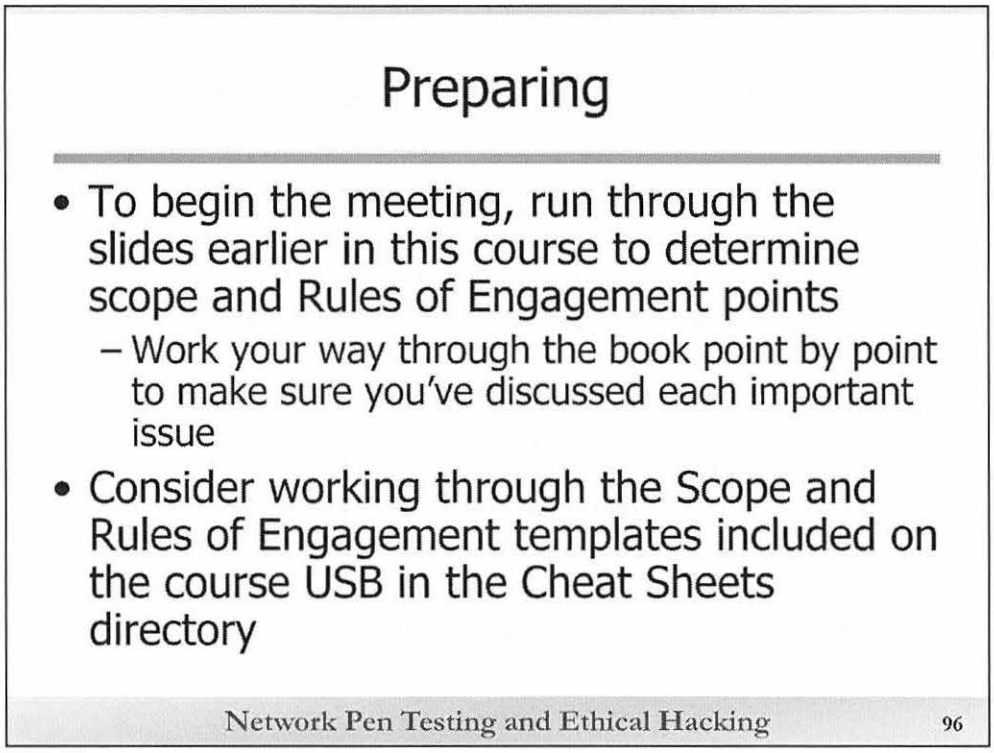
Make sure you take notes in your books and/or on the sheet because you must prepare for this meeting. If you have any further questions, please feel free to ask the instructor.



For this lab, keep in mind that we are focusing exclusively on scoping and setting the Rules of Engagement. ***The meeting is not meant to be adversarial.*** Engage in a positive discussion to determine the proper scope and Rules of Engagement, improvising where necessary.

The clients are NOT evaluating the skills or background of the penetration testers. Furthermore, the penetration testers are not trying to evaluate whether they want to engage in the project. The project has been awarded to the testing team, and both sides are delighted with the decision. The point here is to devise an appropriate scope and Rules of Engagement.

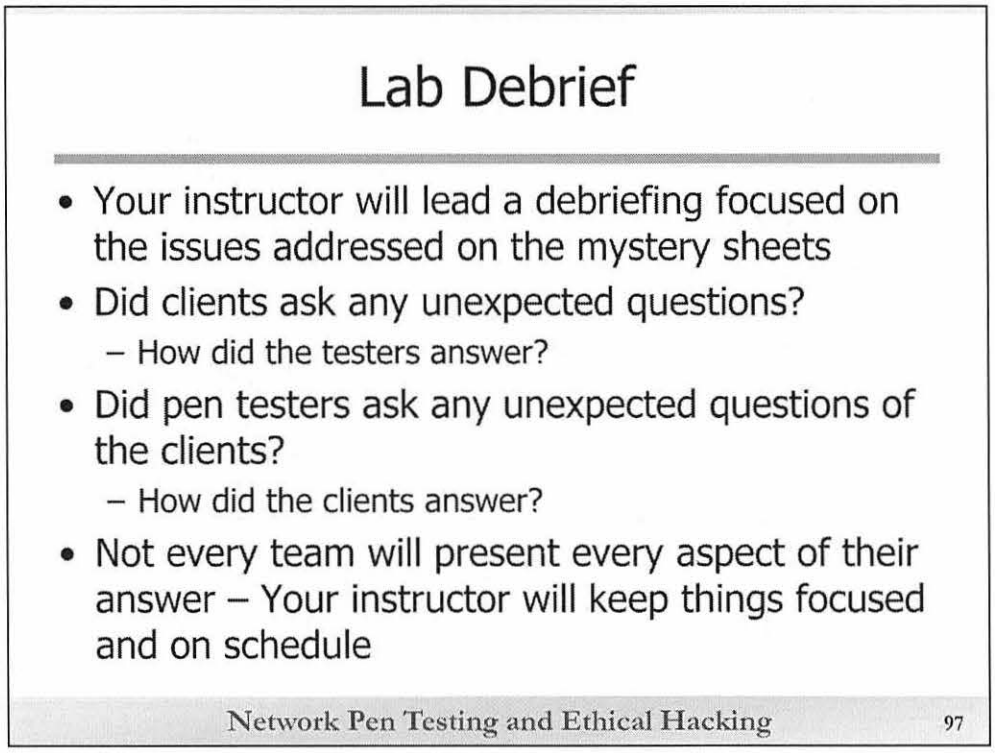
Do not discuss price, level of effort, or qualifications during this meeting because we need to focus on scope and Rules of Engagement.



To begin the meeting between clients and testers, run through the slides earlier in this session to devise a set of questions for scoping and setting Rules of Engagement. Work your way point by point through the book to make sure you've covered each issue.

In addition, you could work through this lab by filling out the templates for Scope and Rules of Engagement included on the course USB in the cheat sheets directory.

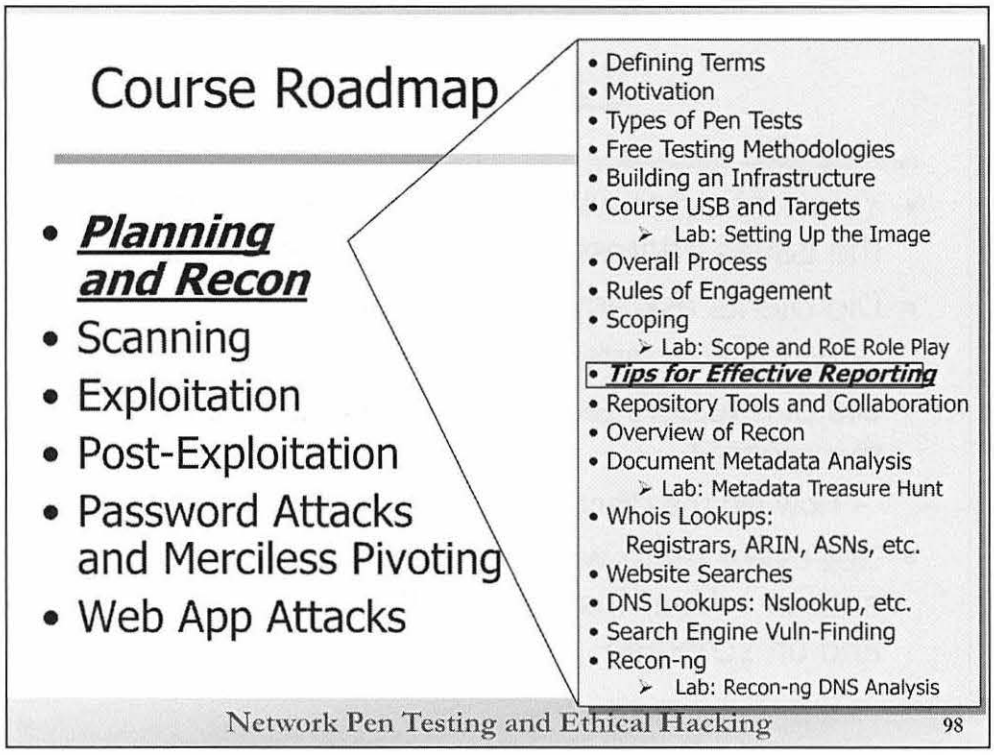
To get ready for the debrief, record your answers to the mystery sheet questions on the preceding page and on this page (or on those pages themselves).



Now, let's conduct a debrief session for the lab. The course instructor will lead a discussion, choosing people from each group to present on the results of their scoping discussion during the lab.

In particular, you will be asked whether the issues on the mystery sheet were properly addressed during the scoping discussion. Also, did the clients ask any unexpected questions? How did the testers answer? Did the testers ask anything out of the ordinary, and how did the clients respond?

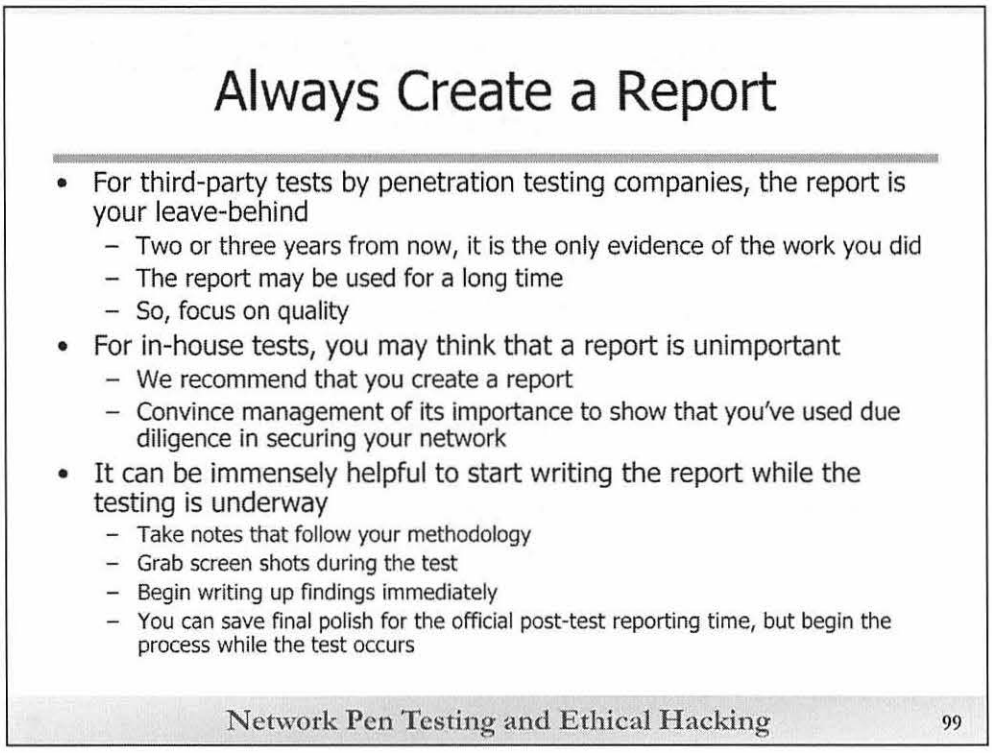
Unfortunately, there may not be time for every team to present every aspect of their results. Your instructor will help keep the conversation focused so that the most salient points will be addressed while keeping the class on schedule.



Next, we'll discuss reporting our results for the penetration test or ethical hacking project. Note that the base report structure that we'll cover is versatile, lending itself to network scans, vulnerability scans, penetration tests, physical security tests, shrink-wrapped product reviews, and more.

Pay careful attention to the reporting component of your project because it is where you'll usually provide the most value for the organization you are testing.

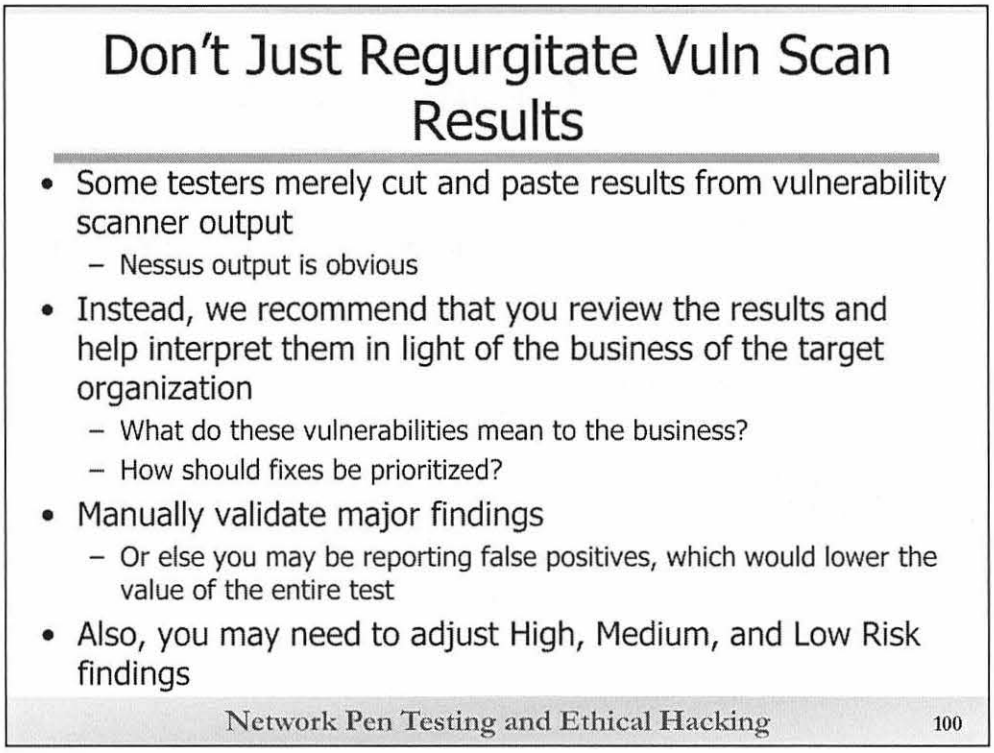
Note also that the reporting format we'll discuss is not only paper-based. The same overall structure can also apply to final presentations describing the project and its findings.



Testers should always create a final report describing their work and findings. If the testers work for a third-party penetration testing or ethical hacking company, the report is the only evidence they leave behind of the project they performed. A high-quality report may be used for several years after an engagement, as fixes are deployed, new architectures are considered, and new products and configurations are rolled out. Because of this potential long-term value of the report, make sure you focus on creating a high-quality deliverable. Reports that are technically incorrect, improperly formatted, or poorly written will linger and could cause significant problems in the future.

Sometimes, testers who perform vulnerability scans of their employers in-house think that they don't need to write a detailed report. After all, you're just scanning your own enterprise network to see how things look. Why waste the extra time documenting your results and findings? Even in such circumstances, we still recommend that you create a report because it helps codify the work you've done and provides an organizational memory of the value you've provided. If management won't provide enough time or staff resources for creating a report, work hard to convince them of its importance. A report is concrete proof that your organization is exercising its due diligence in conducting vulnerability scans on a regular (or at least sporadic) basis. If there are major security problems in the future (such as a major breach), and your organization is investigated by the government or share holders, the vulnerability scanning reports will be helpful in showing your attempts to measure your security stance in the past.

In addition, to increase the efficiency of your overall process, you should consider starting to write the report while the test is underway. That way, your reporting process will not take as long, and the overall quality of your results will better reflect the details of your work. As you conduct the test, take notes that follow along with the documented methodology that you use. Grab screen shots that illustrate your findings as you make those findings. When you have significant results, write them up immediately (or at least a bullet list of their most salient points), while they are still fresh in your mind. You usually don't have to *finish* the report while the testing is underway. Most penetration tests include a reporting period after the hands-on testing is complete. But, this post-testing report time will go much more smoothly and efficiently if you start writing during the test phase.



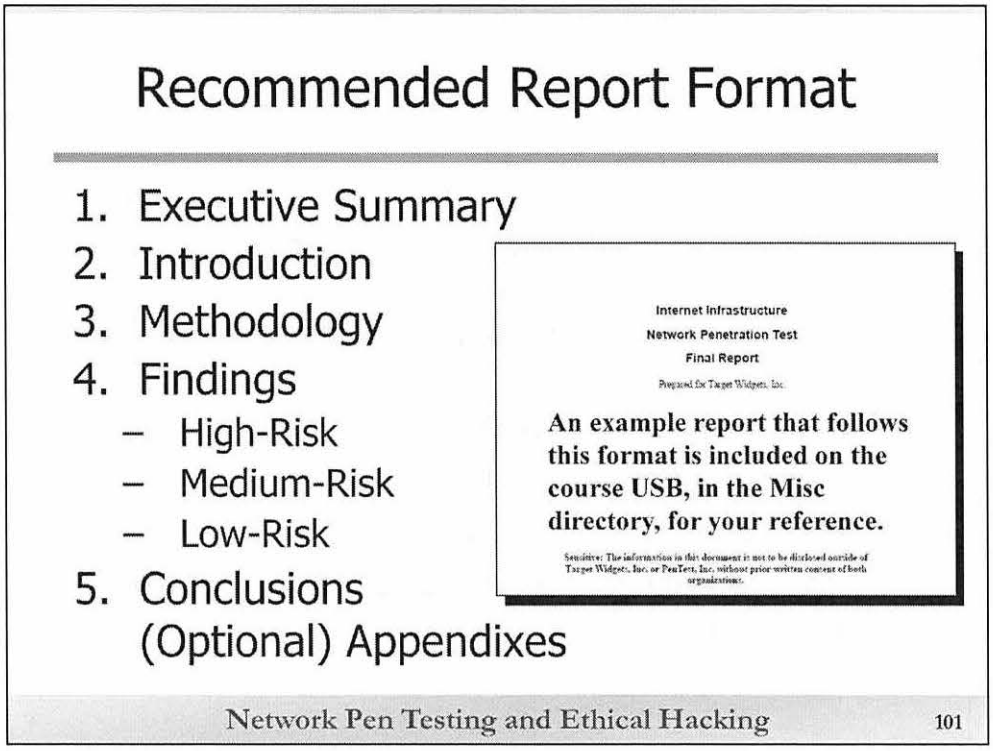
A common mistake of testers in writing reports is merely to cut and paste the output of their vulnerability scanning tools into a document and turn it in as the official report. Sometimes, they don't even format their results, simply throwing Nessus output over the wall hoping that someone will act on it.

Stock vulnerability scanner output is obvious. Its format is canned and often not easily interpreted by nontechnical personnel. Many of the issues that are identified are false positives, and the risks and recommendations do not tie in with the target organization's business objectives.

We strongly recommend that you review the results of your vulnerability scanning tools and other techniques, and then help interpret them in light of the business of the target organization. What do these vulnerabilities mean? How could they impact the business? Given a realistic threat that exploits the vulnerability, what risk does it pose? And, how should fixes be prioritized, whether they be patching, configuration changes, architecture alterations, and s on?

In addition, you should strive to verify all major findings manually, especially those that are high-risk. If you do not validate findings, your report may be full of false positives, which significantly lower the value of your work and may call into question your skills and approach.

Also, note that you may need to adjust the High/Medium/Low risk indications of your vulnerability scanning tools. What Nessus or another scanner calls a High risk might be a Low risk for the target organization. Of course, the opposite is also possible. All these issues have to be addressed in the business context of the target organization.



give comparison with other company

This slide contains a recommended report format. Of course, you may iterate on this structure, adding or removing components. Your organization may already have a report format for penetration tests and related work. If so, get it and use it. If not, feel free to adopt this structure, reviewing it and tweaking it to meet your needs.

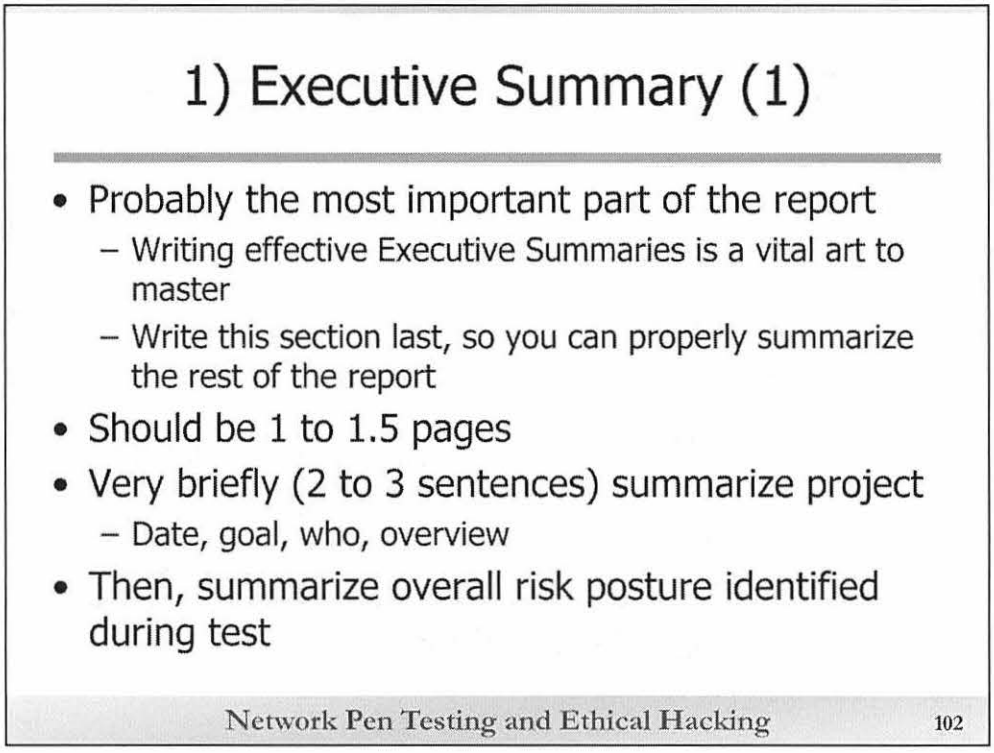
We recommend that the penetration testing, vulnerability assessment, or ethical hacking report includes these elements:

- **Executive Summary**: This brief upfront matter is meant for executives who may not read the full report, providing them the most important conclusions from the work.
- **Introduction**: This component describes the project at a high level, answering the who, where, when, and why aspects of the project.
- **Methodology**: This part of the report describes the "what" of the project: What did the team do? It covers the process of the penetration test or ethical hacking engagement.
- **Findings**: This section presents the actual findings, listed one by one, in the target environment with detailed technical descriptions. The findings are sorted so that the most significant risk issues are discussed first.
- **Conclusions**: This last section summarizes the project results and is reminiscent of the Executive Summary.

The report also may include optional appendixes of reinforcing data and results.

An example report that follows this format is included on the course USB in the Misc directory.

Let's look at each section in more detail.



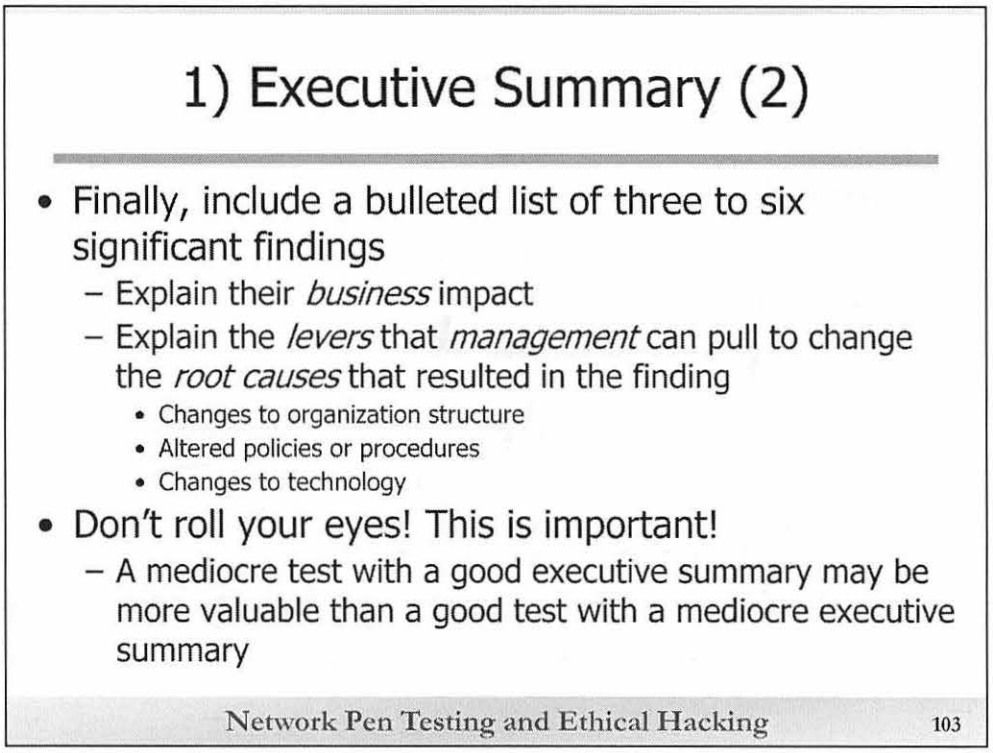
The Executive Summary is probably the most important section of the entire report. Composing an effective Executive Summary is quite an art form and an important one for testers to master. The purpose of this section of the report is to describe to decision makers what the results of the report mean, and, more important, recommendations for the actions they need to take based on its results.

Although it appears at the start of the report, we strongly recommend that you write the Executive Summary *last*, after you have completed the rest of the report. That way, you can properly summarize the findings and recommendations from elsewhere in the already-written body of the report.

The Executive Summary should be quite short. Ideally, it will be under 1 page in length, and should be no more than 1.5 pages. It should not get highly technical nor should it go over every finding. It should be composed so that it can be read by itself, offering firm conclusions and advice for someone who doesn't necessarily read the rest of the document.

The Executive Summary starts with a brief description of the project, in three or fewer sentences, describing the project's date and goals, as well as who participated in the project. It also provides a brief overview of what was done.

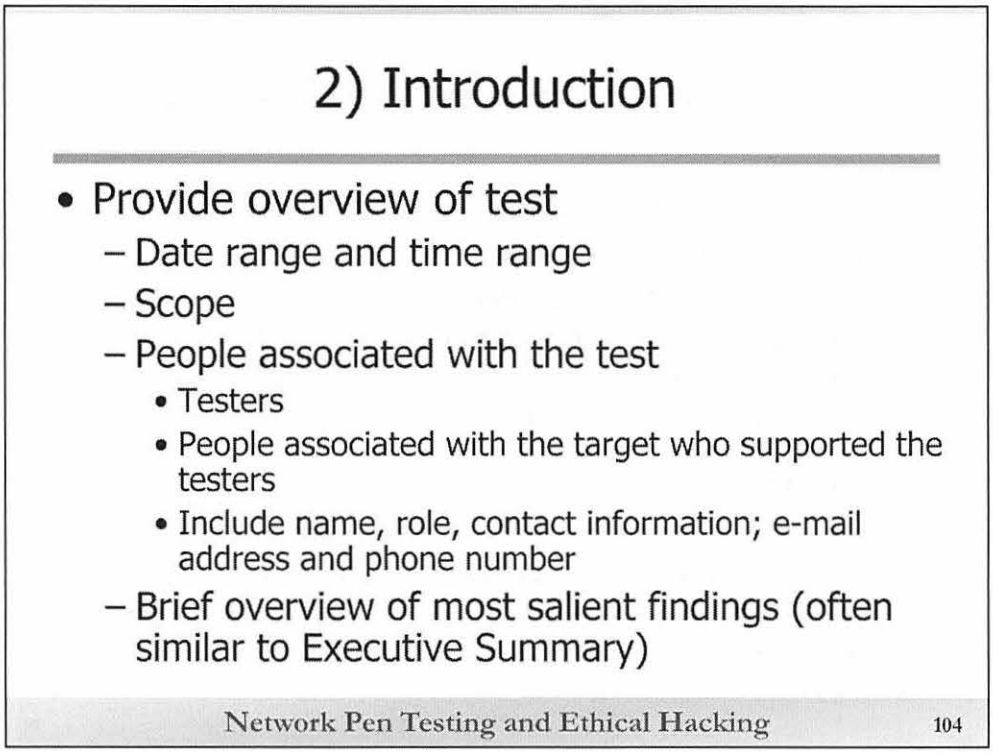
Next, the summary covers the overall risk posture of the target environment. Were major problems discovered? Is the environment worse than expected by testers and/or the people reading the Executive Summary? Or did the test results show that the security of the target environment appeared to be sound?



And now comes the vital part of the Executive Summary. You need to explain the business impact of your findings. You can use a bulleted list of the biggest three to six findings, followed by several sentences that explain the business impact in terms of risk for each finding. Focus on the institutional levers management can pull to change things to eliminate not just a given discovered vulnerability, but the underlying reason that the vulnerability exists. Such levers could include changing organization structure, altering policies and procedures, adding personnel, deploying new technology, or many other possibilities.

Don't roll your eyes about this component of the report and this mindset for composing it, thinking that it is merely management fluff. Effectively communicating your results to management is critical. A mediocre test with a good Executive Summary is likely much more valuable in improving an organization's security stance than a good test with a mediocre Executive Summary. The former provides impetus for change and improvement, whereas the latter just reinforces security vulnerabilities in the minds of technical people who, in all likelihood, know what those flaws are already.

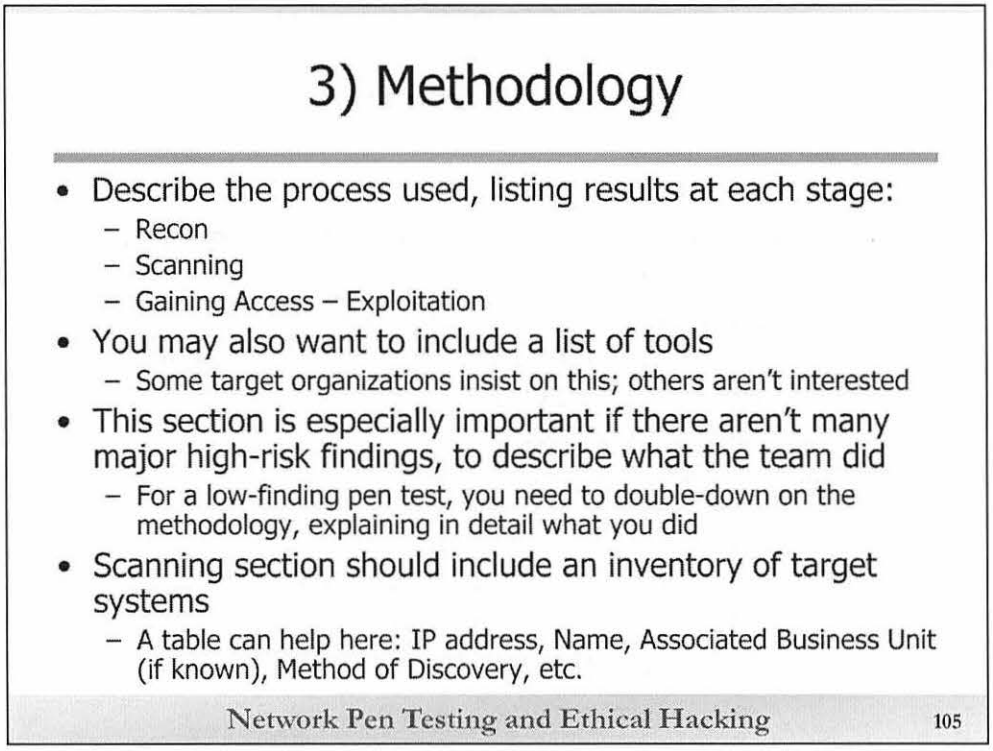
Remember that the Executive Summary should stand alone as a document without the rest of the final report. Often, these one or two pages are split from the rest of the report and sent up a management chain.



Next, we move to the Introduction component of the report. This one-to-three page section provides an overview of the project so that the reader understands when the project occurred, what was included in the scope (and possibly items purposely left out of the scope, if applicable), and who participated in the test.

Include a list of people, organized in a table, that identifies the testers and the individuals associated with the target environment that supported the testers (usually those who participated in the daily/weekly debriefings). The table should include each individual's name, role, and contact information such as a phone number and e-mail address. Then, provide a brief overview of the most salient findings, using language similar to that included in the Executive Summary, but possibly with more technical detail.

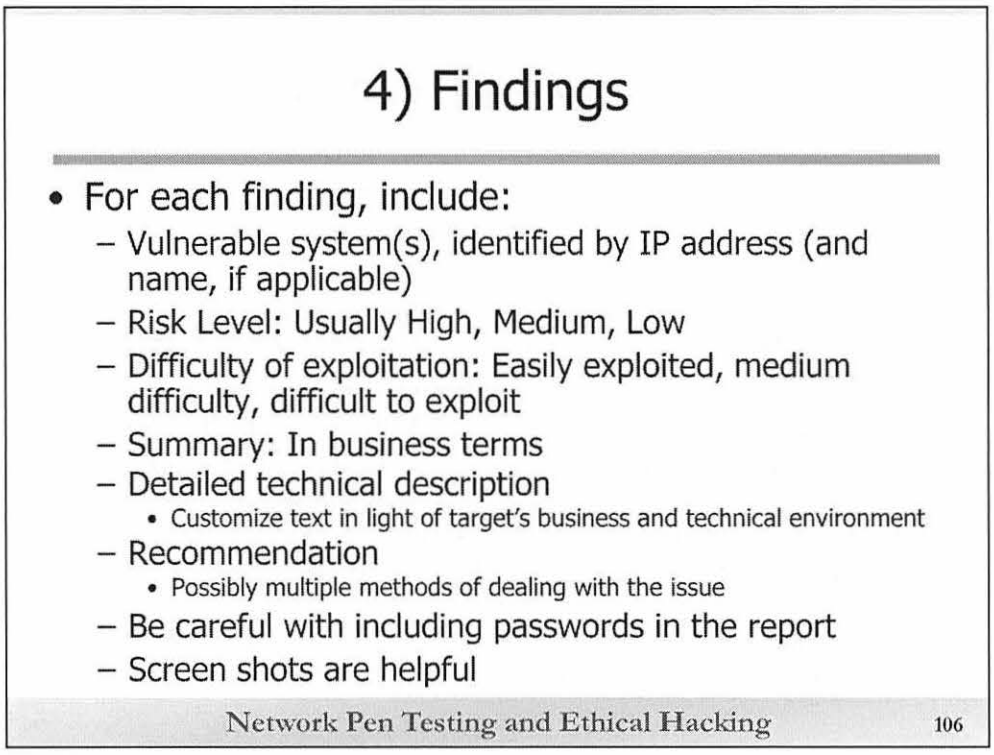
These Introduction components of reports are immensely helpful 6 months or later after the project in analyzing what was tested. Many organizations frequently refer to these Introductions to determine what else in their environment needs to be tested and what needs to be retested.



The next part of the final report includes a description of the methodology used in the testing. Provide a description of the step-by-step process used, and briefly list the findings discovered at each phase. Depending on how you conduct the test, this section may include details of the recon, scanning, and gaining access phases. You may want to include a list of tools you used in the project. Some target organizations want to see the tools to learn from your report, whereas other organizations are not interested in the list of tools. When in doubt, include the tools you used for the project in the narrative description of your methodology, and consider adding a table to briefly summarize the tools used.

This methodology section may run from one to ten pages, depending on how the test was conducted. If there are few high-risk findings associated with the project, this section actually grows in importance to demonstrate the value provided by the testing. Sometimes, a testing project finds little wrong with the target environment. This is, of course, good news. But, if the Findings section (which we'll discuss next) is skimpy, the target organization may question the quality, veracity, and completeness of the test. Thus, in situations in which there aren't many findings, we need to provide even more detail in this Methodology section to show at a fine-grained level what we did during the test. That demonstrates not only that we conducted a thorough test (again illustrating and recording our due diligence), but it also makes the test more reproducible in the future. A year later, the target organization may want to redo the test, and it can rely on this report to replicate it exactly, performing a gap analysis on changes in the interim or purposely tweak the process next time.

The scanning component of the Methodology section should include an inventory of all the machines that were included in the test. We recommend a table with one row per target machine, listing the IP address, machine name(s), the associated business unit (if it can be determined), and its method of discovery (DNS, ping sweep, Google searches, and so on). During the recon discussion later in this course, we will discuss maintaining this inventory while the test occurs. We can directly use this inventory, in a condensed form, in our final report. For some tests, this inventory can get long and should be moved into optional appendixes at the end of the report.



And now, we get to the Findings section, where the technical results are described. We recommend starting with high-risk findings, prioritized so that the highest of the high-risk issues comes first. Then, move down to medium and low risk, again prioritizing each within its subsection.

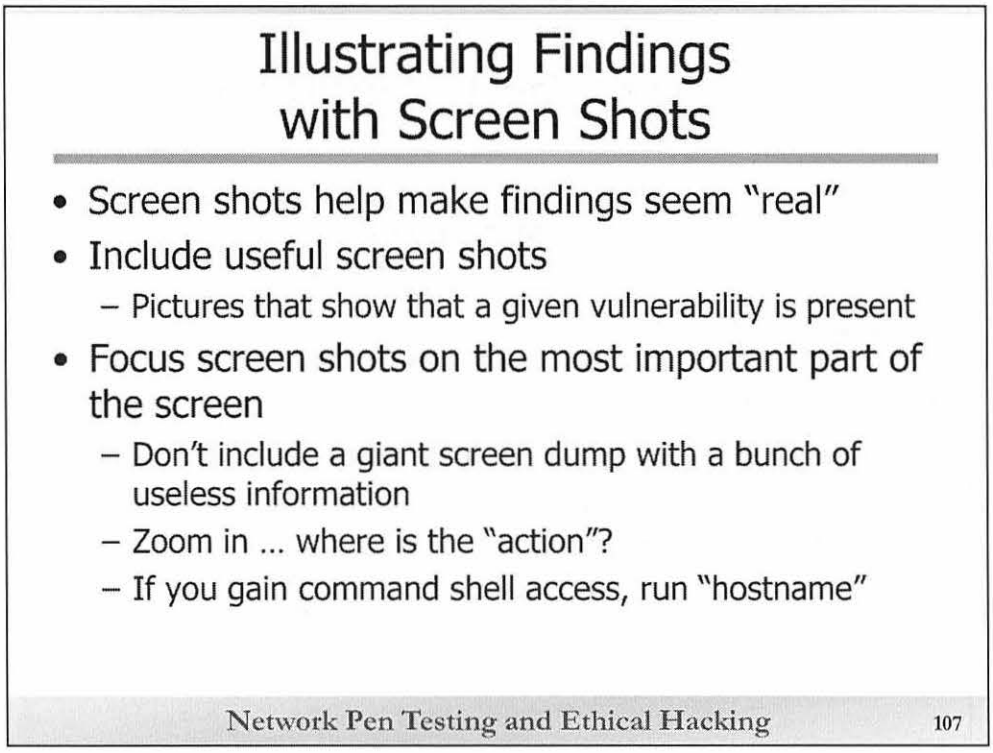
For each finding, list the system(s) that exhibit the given flaw, identifying the machine by IP address and name (if you know it). Include the risk level and estimate of how difficult it would be to exploit the given issue. The risk level is typically sorted into High, Medium, and Low, but your given organization may want to characterize risks in another fashion. Then, provide a two sentence summary of the finding, again focused on the business risk to the extent that you can. For some of the lower-risk findings, the business issue may be fairly small.

We then get into the technical details. Here, we provide a description of what the flaw is, and how an attacker could exploit it, causing an impact to the target organization's business. The discussion is described in technical terms, but again ties back into the business risks.

And, finally, we include a recommendation section, which should provide one or more means of remediating the discovered flaw. If there are multiple ways of addressing the issue, include them all, but point out the one most likely to match the business needs and technical environment of the target organization.

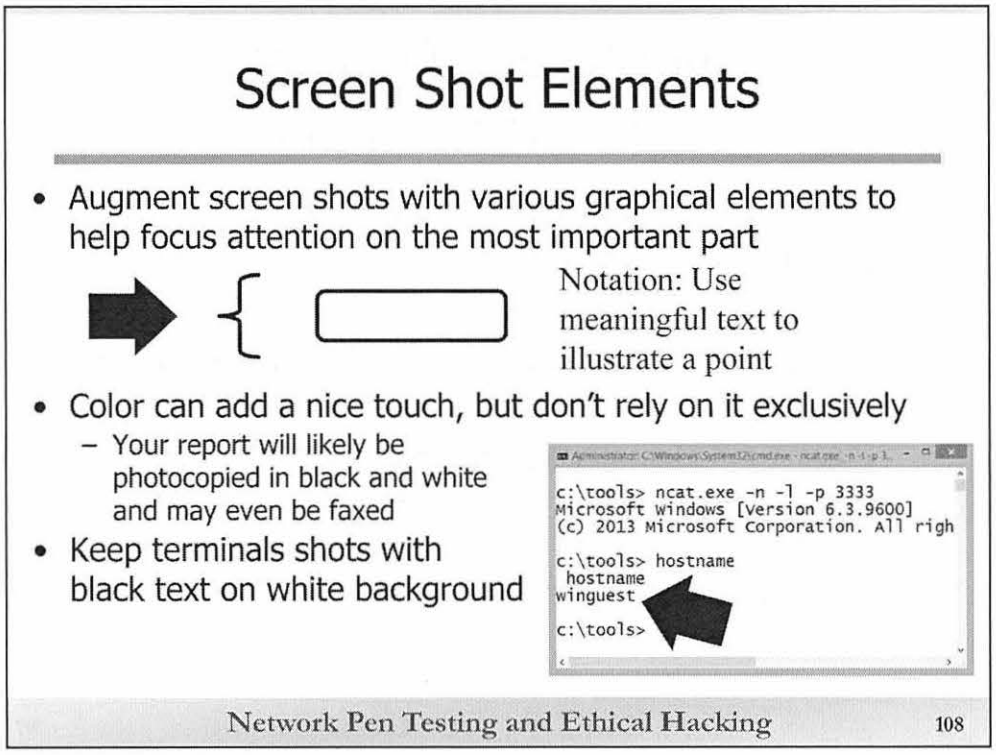
Be careful with including the successfully guessed or crack passwords in your report because these reports are often passed around to numerous people in a target organization. Including in a report passwords that might be reused elsewhere could be a security risk, so verify with the target organization whether such information should be included in a report. Many penetration testers have a policy of describing the characteristics of cracked passwords (such as length or character set) but not including the actual passwords in the report.

In the technical findings section, screen shots that illustrate the issue can be helpful in conveying a lot of information and making the results feel "real." Network diagrams can also go a long way in explaining some technical ideas.



Screen shots aren't just eye candy. They can help a report have its intended effect (motivating the organization to improve its security stance) because they show the success of a penetration tester or ethical hackers' work.

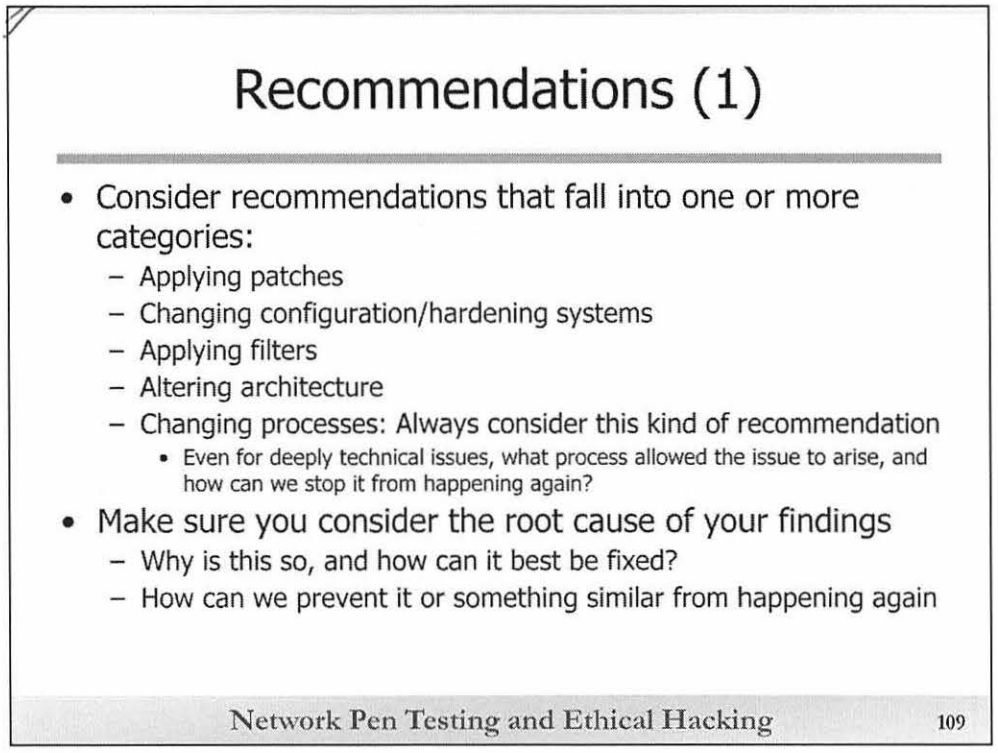
When including screen shots in your reports, make sure to focus the screen shot on the issue you are illustrating in the prose of your report. Some penetration testers include giant screen shots with a lot of detail distributed around the picture, but only a small portion of the figure is actually meaningful. The crux of your screen shot could be lost amid a sea of other unimportant information in the same figure. Focus your screen shots on the action you want to show, which might be a vulnerability discovered by a scanning tool, a command shell returned by a successful exploit (often running "hostname" to show the machine's name), or other useful items.



Screen shots should almost always be augmented with graphical clues as to where the most important information is located in the figure. Use arrows, brackets, or squares overlaid on the screen shot to focus the reader's attention on the most important information. Small textual notations can also help hammer home the point of a given element on the screen. Try not to make your screen shots too busy, however. Remember, you need to demonstrate that a given vulnerability is real, not bewilder the reader with so much obscure detail in your pictures that they will not understand.

Color can help to bring out salient points. For example, you could use red graphical elements to zoom attention on a big problem. However, keep in mind that your beautiful color report will likely be passed on to other people in the target organization via photocopying and faxes and will likely not retain the color elements you create. Thus, while color can help, don't rely on it exclusively to make your point. Make sure that there is contrast between your screen shots and your graphical elements.

Terminals shown in screen shots often work best with black characters on white background. They are easier to see that way (especially when faxed) and require less toner from printers.

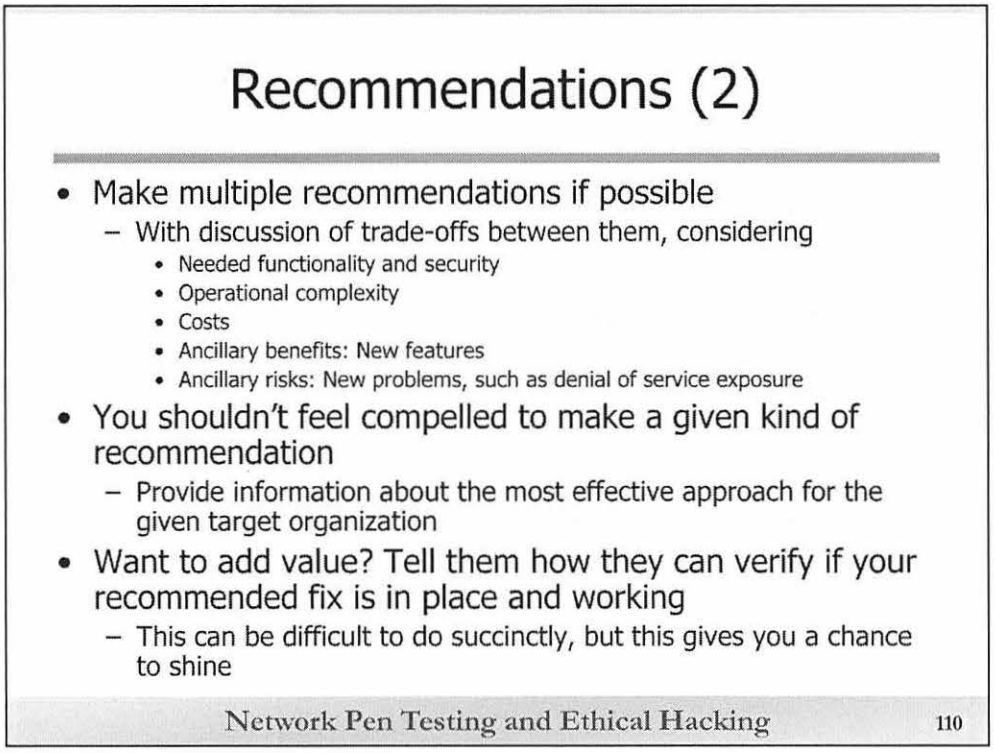


Because they are so valuable, let's look at the recommendations element of the Findings section of the report in more detail. Most of your penetration testing and ethical hacking project recommendations will fall into one of the following categories:

- **Applying patches**: Many times, a test discovers vulnerable software that can be fixed by upgrading to a more recent version or applying a patch. Such changes tend to be easy, provided that they are conducted in coordination with the change-management*/hardening systems:* Tests often have findings associated with shutting off an unneeded service, removing software that doesn't have a defined business need, or otherwise changing the configuration of a target machine to harden it from a security perspective.
- **Applying a filter**: Some issues identified during a test can be mitigated by applying a filter in front of the target service, through a network firewall, network-based Intrusion Prevention System, or even a host-based filtering technology, such as a personal firewall or host-based IPS.
- **Altering architecture**: The most effective way of dealing with some flaws is to alter the overall architecture of the target environment, moving systems around, deploying new kinds of technology, applying filters in a different order, or tweaking the flow of data in the environment. Such solutions tend to be among the most expensive when compared to other issues in this list.
- **Changing process**: For each one of your recommendations, always consider making a recommendation to improve processes. Even for deeply technical issues, some process associated with the given technical issue allowed the vulnerability to arise. Make recommendations for improving the process to avoid such an issue happening again in the future.

Most recommendations in a penetration testing or ethical hacking report fall into the first three categories here: patches, hardening, and filtering.

Whenever you make a recommendation, make sure you consider the root cause of the problem. Why is the issue present in the first place? How can target system personnel prevent it from happening again in the same instance or in other related areas? Answer those questions and you'll be providing additional business value for your work.



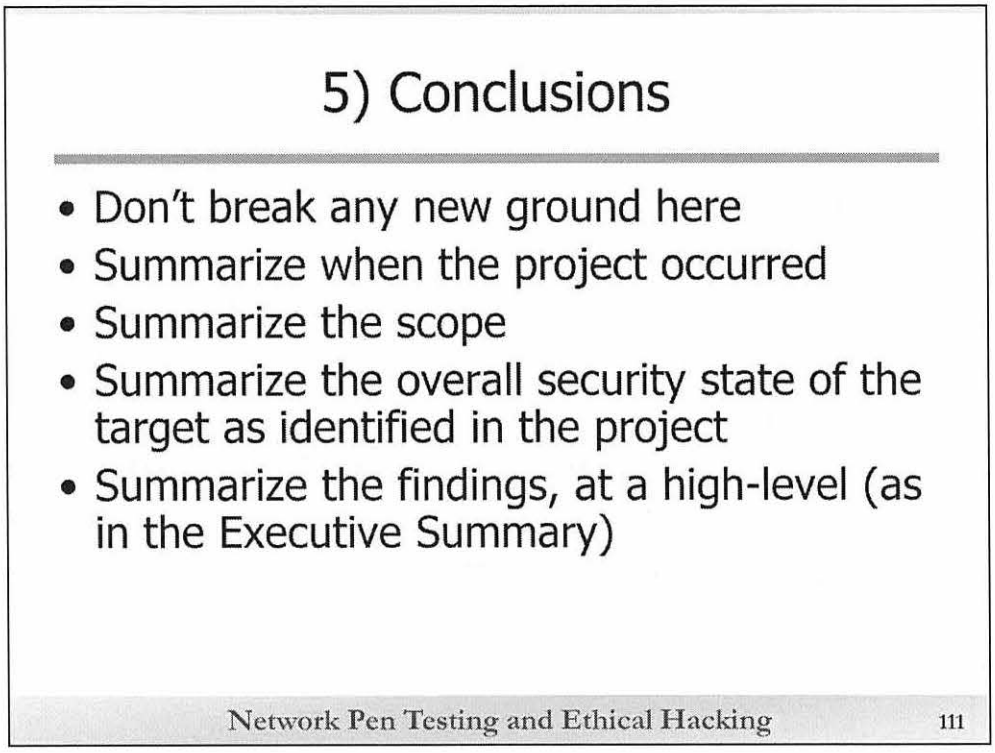
Where possible, provide multiple recommendations for addressing a given discovered vulnerability. Discuss the trade-offs between the recommended solutions in light of the target organization's environment, which could be associated with different impacts on operational complexity, varying costs, and ancillary benefits of some solutions over others (including the target organization's familiarity with a given type of operating system or software package, the enabling of future services or features from one solution over another, and more). Some solutions could have ancillary risks, as they open new vectors of attack, possibly including denial of service exposure.

Make sure that you base your recommendations on what you feel is truly the best way for the given organization to deal with a discovered issue, independent of pressures to raise or lower costs. If cost is a concern, you should list both higher- and lower-cost alternatives, and discuss which ones make the most sense in light of your understanding of the target environment.

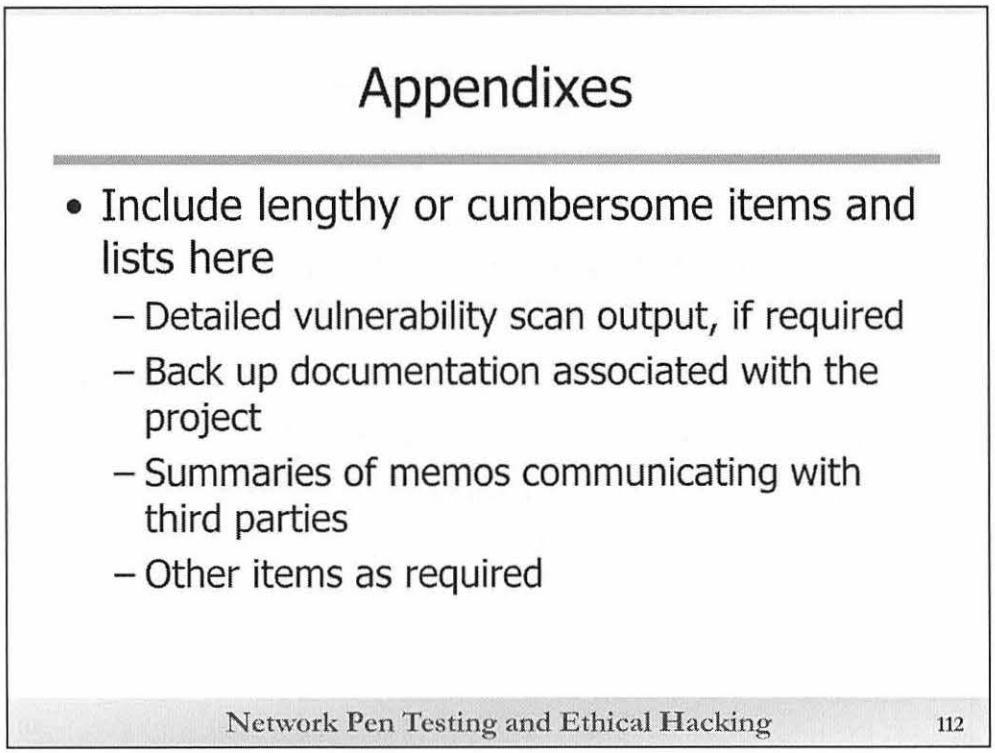
Don't be pressured to recommend only high-cost solutions, which some third-party penetration testing companies might do to raise the price of follow-on mitigation projects.

From the opposite perspective, recommend budget-conscious approaches to make sure you have a possibility of improving the security stance of the organization. But don't be pressured to use only low-cost solutions even if budgets are slim.

If you want to go above and beyond in providing value in your penetration tests, provide in your recommendation some steps an organization can take to verify that your recommended fix is in place and working effectively. For example, if you recommend applying a patch or a filter, provide target system personnel with some command-line activities it can run to verify that the patch is in place or that the filter is properly filtering. Such fix verification advice can be difficult to formulate succinctly, but it provides some extra verification for target system personnel that its defenses have improved because of your work. Might this prevent you from getting some extra follow-on testing work? Yes, that is possible, but it makes your initial test results and report much more valuable to the organization.



The final component of the report is its Conclusions section. This section is usually about a page in length and summarizes the various aspects of the project covered elsewhere. This part of the report shouldn't break any new ground. Instead, it briefly summarizes topics included already in the Executive Summary and Introduction, such as when the project occurred, its scope, and the overall security state of the target organization determined by the testing. The conclusions can also include the bulletized list of major findings presented in the Executive Summary.

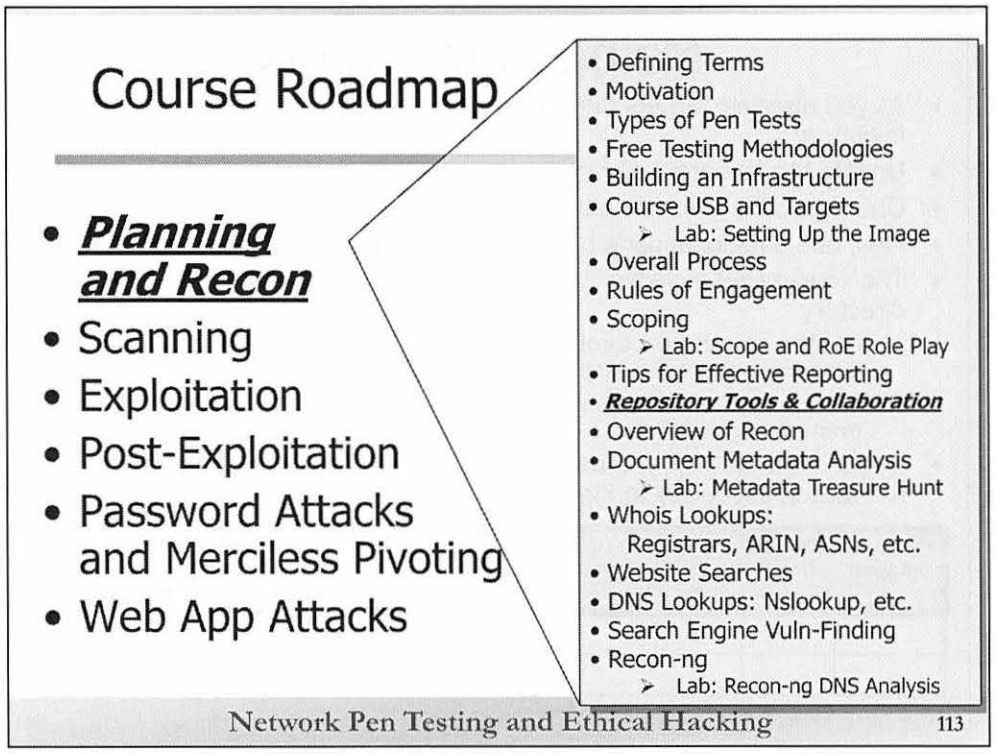


The report appendixes should include lengthy outputs that would be cumbersome to put inline with the rest of the report. You could include detailed vulnerability scan output in an appendix. However, make sure that the report recipients want such information. It is often counter-productive and may make it look like you are merely adding weight to the report without any useful business or technical reason.

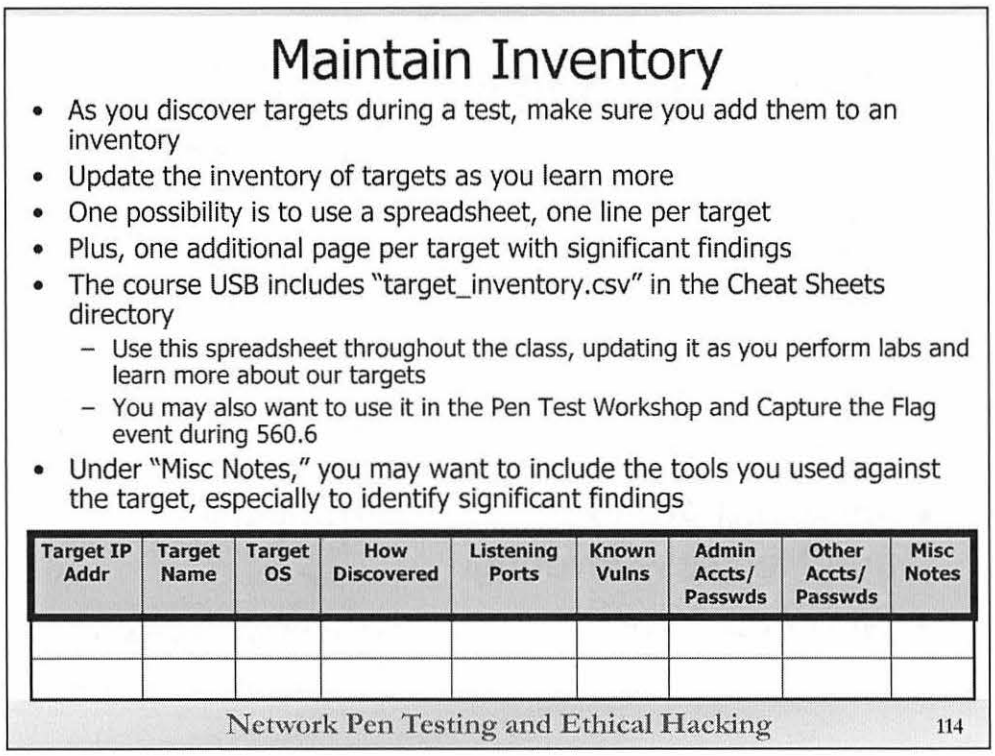
You could include back-up documentation associated with the project in an appendix, like the Rules of Engagement or scope description.

You also may want to include any memos that were sent to third parties getting their permission to scan their systems, such as a DNS server or web-hosted environment. Also, if any requests were sent to these organizations to change their configuration, you may want to include a copy of the request and response for the change in an appendix.

Include other items as they are required or helpful.



In our next section, we'll start with a brief overview of the Reconnaissance phase, looking at an overview of its goals and methods. We'll then look at some helpful recon tools, including Internet-accessible search tools and software you can run on your own systems for recon analysis.



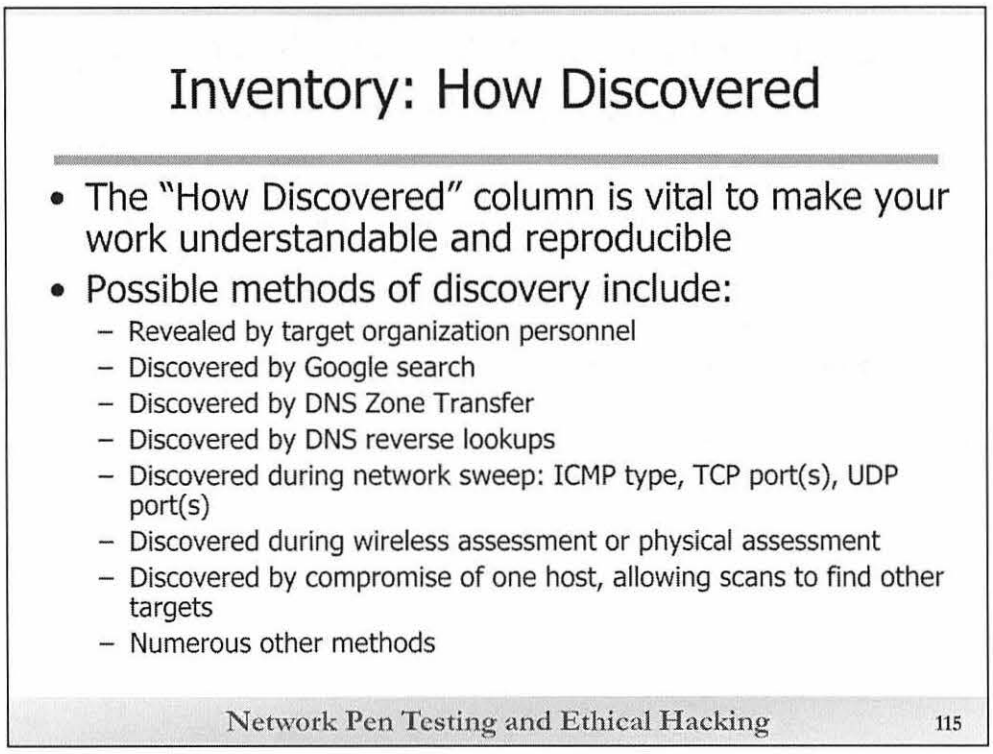
Throughout the test process (starting with Recon and Footprinting but going throughout the exploitation phase and through the end of the test), it is vital that you record your results in an organized fashion. Disorganized penetration testers and ethical hackers are often far less successful. The last thing you want to do is to miss out on a vital vulnerability in a target organization because it was lost in disorganized clutter. We're not trying to stifle your creativity or limit your style. But, we are trying to keep focus on performing the highest quality tests that we can.

One of the most helpful tools for recording results is an inventory of all discovered targets and their associated details. A convenient way to store this inventory is by using a spreadsheet, such as the one shown on this slide. Each discovered target system gets one line in the inventory, with the details populated as they are discovered throughout the remainder of the test. Under the Misc Notes column, you could include one or more tools used against the host, as well as a specific tool or technique that yielded particularly important findings.

The inventory includes numerous fields, such as the Target's IP address, name, and operating system type. Some of the most important fields to populate are How Discovered (which we'll cover in more detail shortly), known vulnerabilities, and the accounts and passwords that are determined. Note that you might not populate every field for every discovered target. For example, it is possible that a given discovered target will have an IP address but you cannot discover its name. Perhaps the target does not have a DNS record, and you cannot get access to the box. Thus, you will likely not know its name. Instead of leaving the field blank, we recommend entering Not Found or Not Applicable to at least show that the given field was not overlooked.

For hosts with more significant findings beyond what is listed in this table, you may want to have a document with one page per significant host. For example, if you can compromise a host, getting remote command shell access to it, you might use that shell to take an inventory of that machine's information assets, configuration weaknesses, and the other hosts that it can reach. This information will likely not fit into your spreadsheet but must be recorded in all its exciting detail. Store that information on a series of separate pages, with approximately one page per host.

The course USB includes a file called target_inventory.csv in the Cheat Sheets directory to use as a sample throughout this class.



The How Discovered field on the inventory spreadsheet is one of the most important items because it contains information about how you first identified that the given host was present in the target environment. Recording this information makes your test more understandable (to answer, "Where did all this information come from?") and repeatable so that others can verify the integrity of your results.

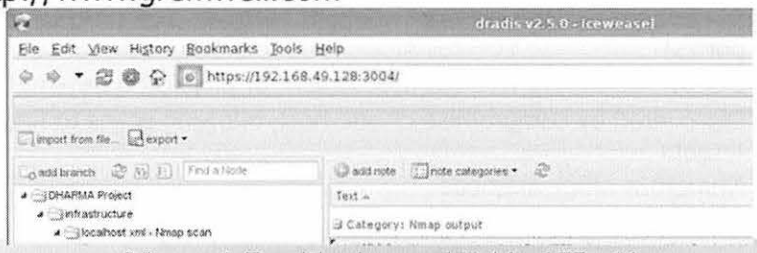
This field may include numerous potential values for how a given host was discovered. In fact, if you want to be thorough, it might include multiple methods as you verified the presence of a given host with alternative means. Some of the methods that will likely be recorded in this field include

- **Revealed by target organization personnel**: At the outset of the test, the target organization's representatives may provide a list of targets as a starting point. We need to indicated those which were given to us.
- **Discovered by Google search**: Google is a treasure trove of information.
- **Discovered by DNS Zone Transfer**: DNS provides a great deal of information, if zone transfers are allowed.
- **Discovered by DNS reverse lookups**: We can find hosts by doing reverse DNS lookups.
- **Discovered during network sweep, ICMP type, TCP port(s), UDP port(s)**: There are numerous methods for sweeping through a network range to find hosts.
- **Discovered during wireless assessment or physical assessment**: If the test includes wireless, we may find some hosts via this method.
- **Discovered by compromise of one host, allowing scans to find other targets**: This is one of the most exciting methods for discovering hosts, pivoting through one and finding additional targets.
- **Numerous other methods**: This list cannot be exhaustive. There are a huge variety of other methods you may use in finding target machines.

Don't worry if you are unfamiliar with some of the items in this list right now. We'll spend the next several course sections going through how to perform each of these techniques.

Other Pen Test Inventory, Recording, and Collaboration Mechanisms

- Some penetration testers store their results in a wiki, such as MediaWiki, to support collaboration among testers
- Alternatively, the Dradis tool is designed for collaborative recording and analysis among a group of pen testers, with auto report generation:
  - Server runs on Windows, Linux, or Mac OS X
  - Command-line client, several thick clients, and web-based interfaces
  - Supports importing Nmap, Nessus, Qualys, Nikto, and Burp scan results
- A similar tool is MagicTree, available at http://www.gremwell.com



Instead of a spreadsheet and individual documents to store results, some penetration testers prefer an online collaborative environment for documenting and analyzing their findings. Such online systems are better-suited for multiperson penetration tests than individual spreadsheet or word processing documents.

For online result storage and collaboration, some penetration testers set up a wiki (a website that allows easy updates of textual information and sharing of files among multiple people). MediaWiki is a common choice, freely available at www.mediawiki.org.

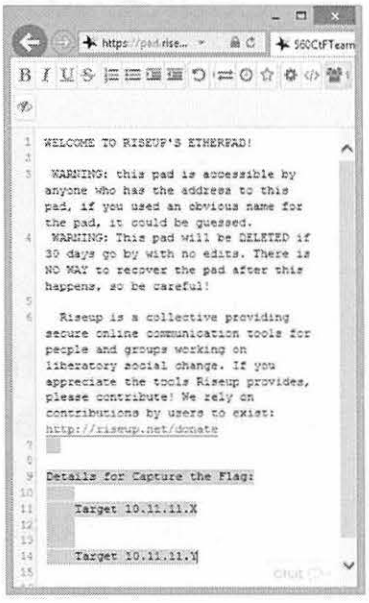
Alternatively, the Dradis tool is a Ruby-on-Rails project designed for recording information among multiple penetration testers working on one or more projects together. The Dradis server runs on Windows, Linux, or Mac OS X and features multiple client options: a command-line client, several different thick client applications, or a web-based interface.

All results are organized as a hierarchical tree, typically organized starting by overall project, then split according to functional areas of the target infrastructure (for example,, DMZ/intranet/extranet, servers/network devices/clients, or other applicable divisions of the test's scope), then separated by individual devices, down to individual ports on those devices, and then through findings and notes associated with each port.

With Dradis, a tester can import results from the Nmap port scanning tool, the Nessus or Qualys network vulnerability scanning tools, the Nikto web server scanning tool, or the Burp web application attack tool. In addition, the tester can manually enter findings and notes, or add analytical notes to results already imported.

Additional Collaboration Tools

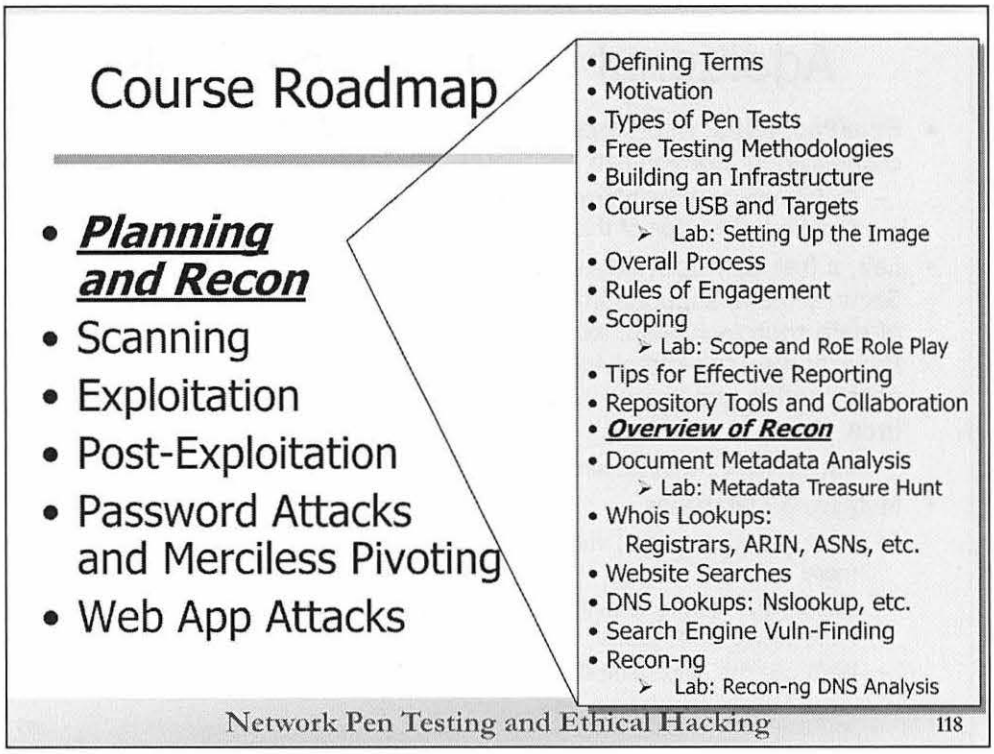
- EtherPad, a free web-based open source collaboration tool you can host yourself
  - Or for nonsensitive information, there is a free hosted version of this at pad.riseup.net
- Lair, a free tool from Accuvant's FishNet Security team, supports importing a variety of data sources (Nmap, Nessus, Nexpose, Burp, and so on) and giving access to collaborating penetration testers via their browsers
  - https://www.fishnetsecurity.com/lair
- Metasploit Databases
  - Stores hosts, services, vulns, creds, loot, and more
  - Import info from scanning tools or based on Metasploit's own results
  - We'll use the Metasploit Database in 560.3



Another excellent browser-based collaboration tool is EtherPad, which provides a free server you can download and host, access via a browser, and share a text-based workspace and interactive chat. Alternatively, if you would like to share nonsensitive information (such as for a CtF, not a penetration test), you could use the free hosted version of EtherPad at http://pad.riseup.net.

The FishNet Security team at Accuvant has released its Lair tool, which it uses in-house to collaborate on penetration tests. It can import data from a variety of scanning tools, including Nmap, Nessus, Nexpose, Burp, and more. Multiple penetration testers can then access it via their web browsers to analyze findings and collaborate on a report.

And, finally, the Metasploit exploitation framework supports a database into which you can import results from various scanning tools. In addition, Metasploit can store its own results in the database, which includes a hosts table (for targets), a vulns table (for discovered vulnerabilities), a creds table (with usernames, passwords, and hashes), a loot table (for pilfered information, including crypto keys), and much more. We'll use the Metasploit database hands-on in 560.3.



The next portion of the course deals with the first phase of the test: Reconnaissance. During this phase, the tester learns as much as possible about the target organization. One important aspect of this recon phase is building an inventory of potential target machines that are likely associated with the target organization. This inventory must be vetted carefully to ensure that it is indeed in scope before any scanning activities (the next phase of the process after reconnaissance) can begin.

When planning a test, we recommend that you budget and schedule at least 8 to 10 hours of detailed reconnaissance work or more if resources allow. Don't skip recon. It can provide crucial insights for the remainder of the entire test.

Reconnaissance

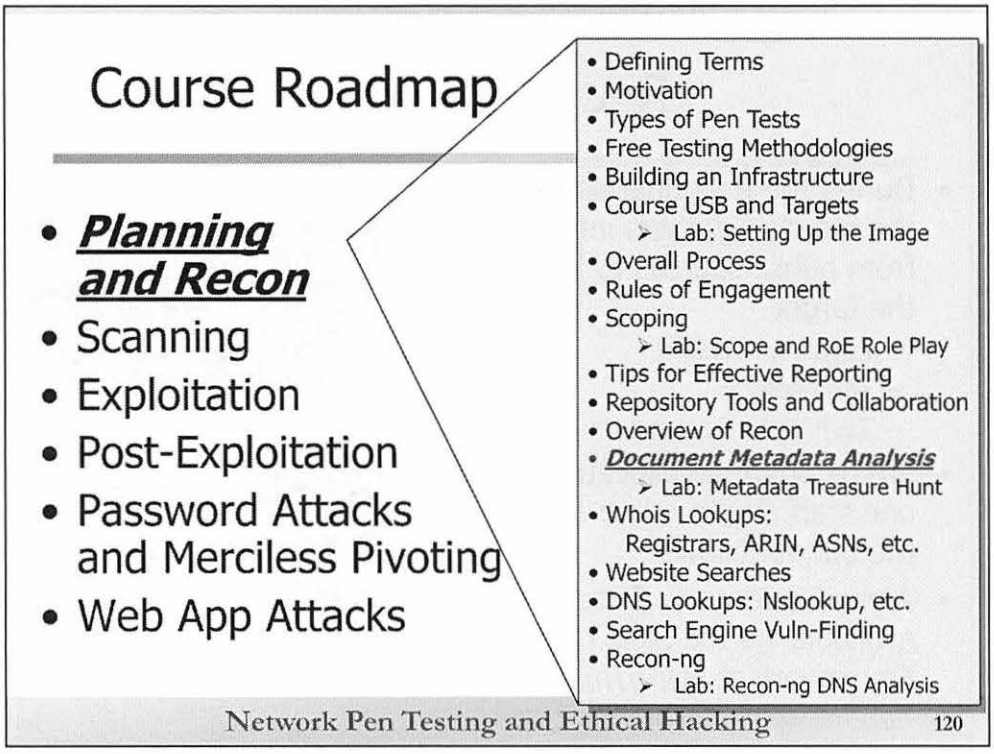
- During the Reconnaissance phase, the attacker gathers information from public sources to learn about the target
  - People and culture
  - Terminology
  - Technical infrastructure
- We recommend allocating at least one staff day to recon and more if the budget allows
- A giant inventory of recon tools is available via the Skip Tracing Framework at http://makensi.es/stf/

Network Pen Testing and Ethical Hacking 119

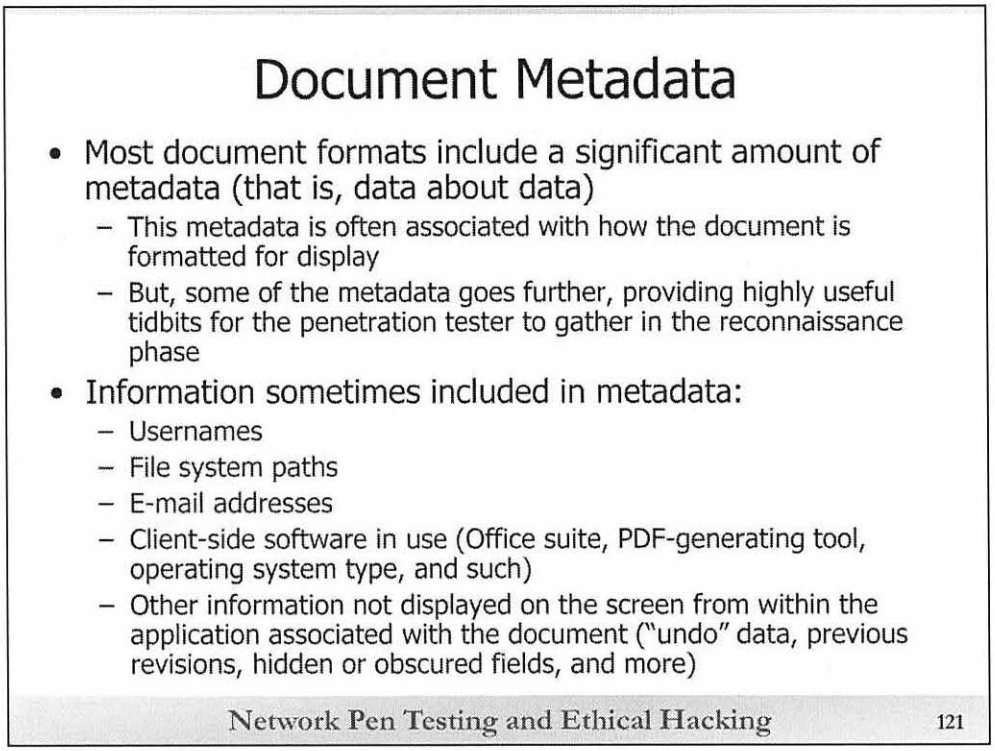
After the test has been thoroughly scoped and any required agreements are signed, the test begins with the reconnaissance phase. In this phase, the tester gathers information about the target organization from various public sources. The tester needs to become familiar with the target's people and culture, learning the specific business terminology used by people in the target organization. We try to find out what is important to the target, and what they are telling the public about how they do business. We also seek tidbits about the technical infrastructure of the target organization, looking for clues about its architecture, products, and configuration in public sources.

This recon phase is extremely important in conducting thorough penetration tests. Don't dismiss it because it doesn't get deep into technology. The information gathered during the reconnaissance phase will be helpful throughout all the other testing phases and will be instrumental in the development of the final report.

Given its importance, we recommend that the scope of the test include at least one staff day (8 to 10 hours) for reconnaissance and more if the budget allows.

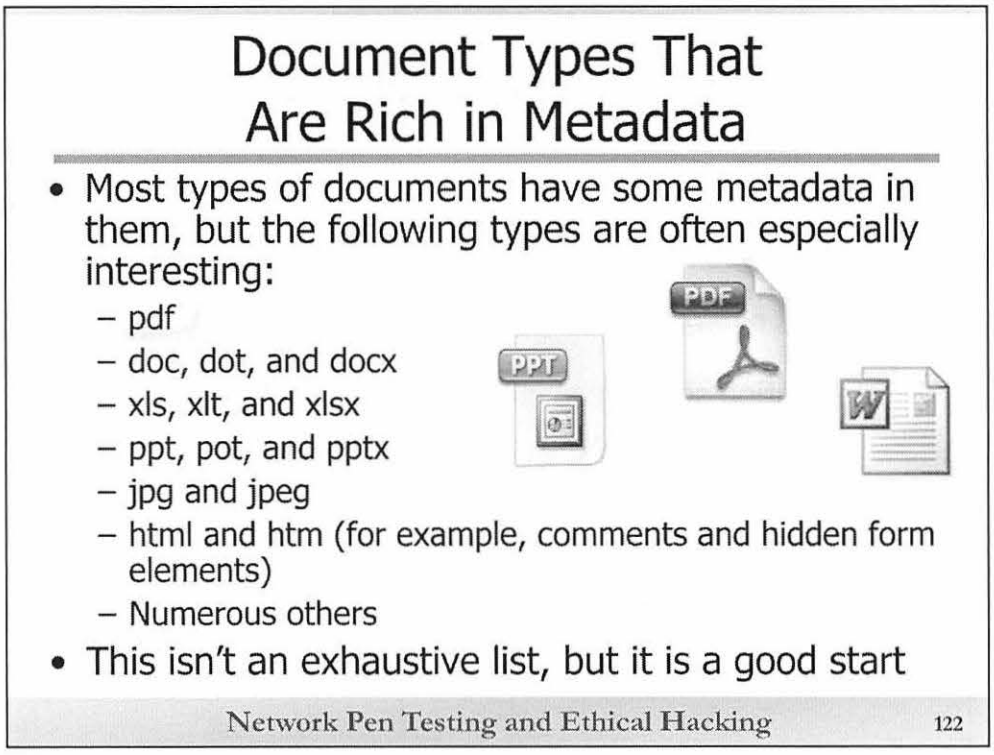


A useful source of information during reconnaissance is the metadata stored inside of documents that penetration testers can gather from the target's website and target organization personnel. The next section of the class analyzes the topic of document metadata and how penetration testers can use various tools to gather and analyze it during the reconnaissance phase of a test.



As organizations create documents, the software that they use to create these documents embeds an enormous amount of information in the document files. Of course, much of this information is the contents of the file. But, a good deal of metadata (that is, data about data or data describing other data) is also included in the file. Much of this metadata is associated with formatting and display of the other data in the file. Besides this formatting metadata, a lot of file creation and editing tools include additional metadata entries that can be useful for penetration testers during our reconnaissance phase, such as

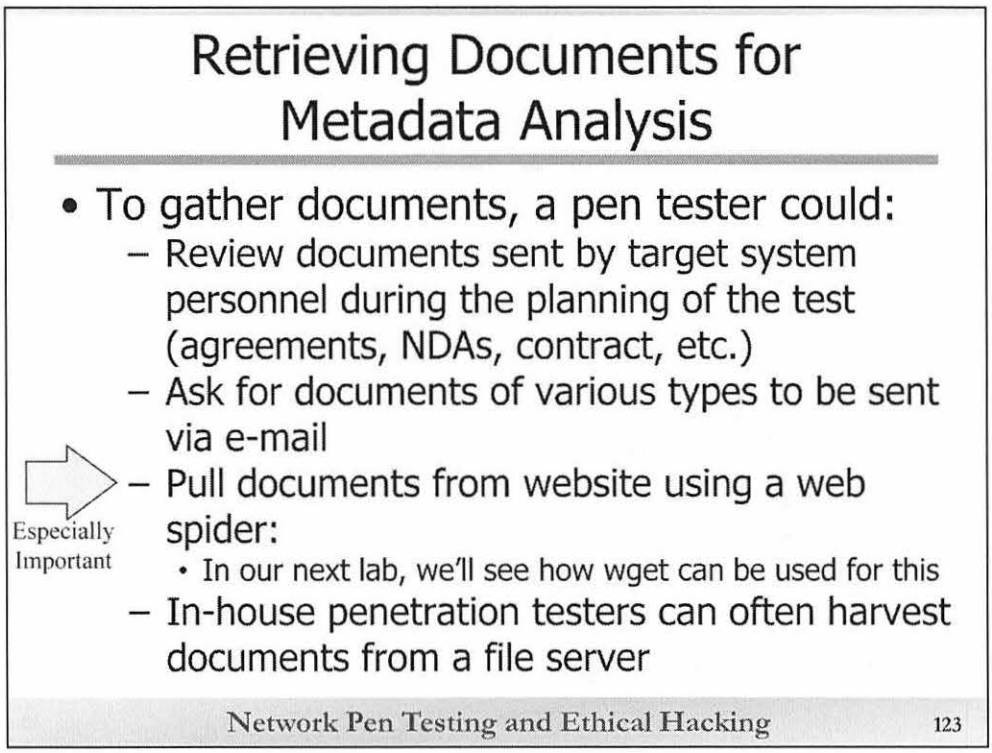
- **Usernames**: Penetration testers often need usernames for exploitation and password-guessing attacks.
- **File system paths**: Knowing the full path of the original file when it was created can reveal useful tidbits about the target organization, including hints about important commonly mounted file servers, critical directories, and common practices of the given user.
- **E-mail addresses**: This data can be useful if the penetration test scope includes spear phishing tests (sending e-mail to target personnel to see if they will click links or open attachments). However, such tests should be performed only if they are explicitly allowed for the given target personnel explicitly included in the scope of the test.
- **Client-side software in use**: Given that client-side exploitation is such a common attack vector, it can be helpful to penetration testers to know which client-side programs are in use, including the office suite, PDF-generating tool, and even operating system type. Metadata often also reveals version numbers of this software, but those versions were in effect when the document was created or last edited and are not necessarily the current version.
- **Other information**: Other useful information is often associated with content of the document that isn't displayed on the screen within the application, such as undo information, previous revisions, and hidden or obscured information (such as a collapsed column obscured in a spreadsheet, or critical text in the document hidden under a picture).



Almost every document type has some form of metadata, but some are richer in metadata than others. The following types of documents, generated and used by most enterprises, are of particular interest to penetration testers:

- **pdf files**: These files are associated with Acrobat Reader and a variety of other pdf creation and editing tools.
- **doc, dot, and docx files**: These files are associated with Microsoft Word but are also used by several other related tools. Although doc and docx are content files, a dot file is a template often used as a source to create other documents.
- **xls, xlt, and xlsx**: These are common spreadsheet files, often associated with Microsoft Excel.
- **ppt, pot, and pptx**: These files are associated with Microsoft PowerPoint and other slide-generating programs.
- **jpg and jpeg**: These image files often contain a significant amount of metadata, including data about the camera used to take a picture, the file system of the machine where the image was edited, and details about the image-editing software.
- **html and htm**: These file types contain web pages, and may at first seem uninteresting. However, their comments and hidden form elements could contain metadata that is useful to a penetration testers. In addition, scripts embedded in the HTML may reveal sensitive information or undocumented features of a web application.

In addition to these types of documents, there are hundreds of others that may be interesting. This list is not intended to be exhaustive but is instead designed to get the reader thinking about interesting and useful document types to analyze during a penetration test.



To perform metadata analysis, a penetration tester must first retrieve files to analyze. Numerous methods could be applied to gather these documents.

First, the penetration tester may have already received some documents generated or edited by target system personnel during the planning of the testing project. For example, the tester may have received Rules of Engagement agreements, scope information, diagrams, non-disclosure agreements, contracts, policies and procedures, and other information.

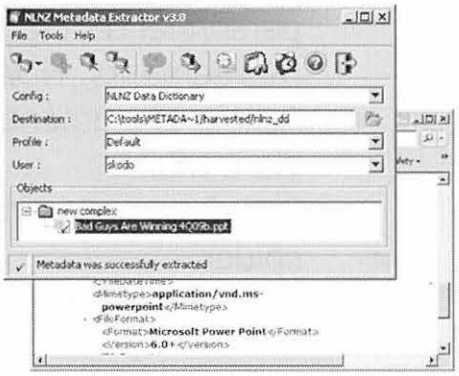
In addition, a penetration tester can simply ask target system personnel for documents. During the recon phase of the test, you could ask them to e-mail you a document created by their word processor, spreadsheet, and slide editing tools. You could also request a PDF document from them.

One of the most common and especially important methods for harvesting documents for metadata analysis is to use a web spider tool against the target organization's website, pulling all potentially interesting documents onto the penetration tester's machine for analysis. In our next lab, we discuss how the wget tool can be used for this.

And finally, if the penetration test is conducted by in-house testers (employees of the same target organization), they typically can get an ample supply of documents for analysis from file servers in the organization.

Tools for Analyzing Document Metadata

- There are dozens of different tools for analyzing document metadata, such as:
  - ExifTool
  - FOCA
  - Metadata Extraction Tool by the National Library of New Zealand (NLNZ)
  - Strings
  - Numerous others
- We'll look at two of the most useful and popular: ExifTool and Strings

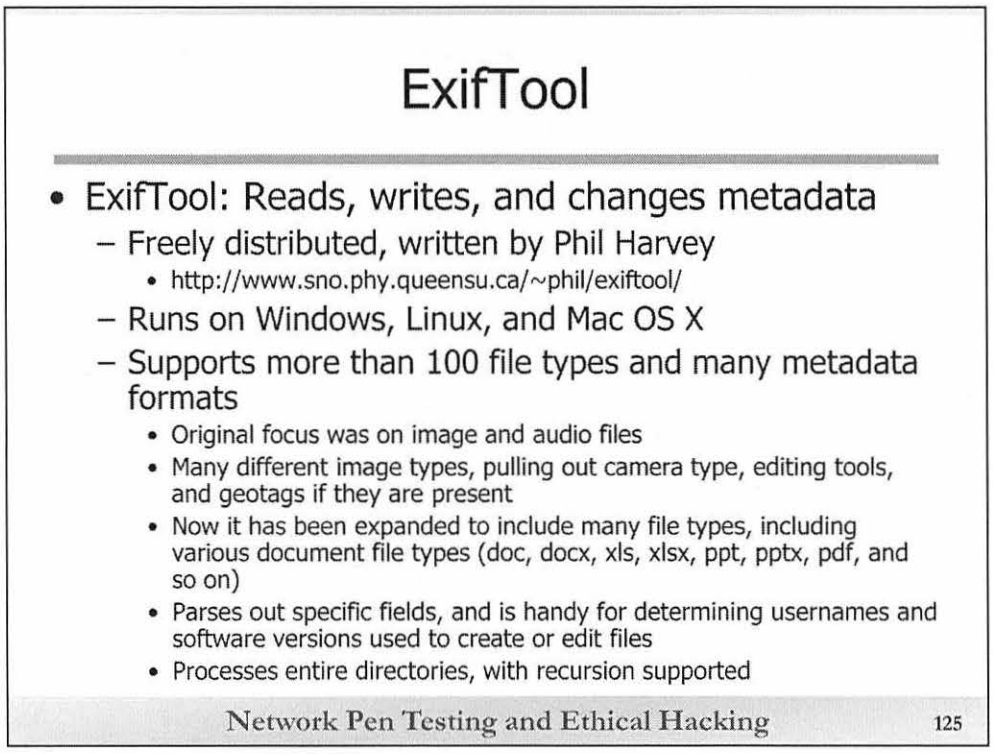


Network Pen Testing and Ethical Hacking 124

There are a variety of tools to extract metadata from documents. Some of the most powerful include ExifTool, FOCA, and the Metadata Extraction Tool by the National Library of New Zealand. All these tools are free and are focused on extracting specific, enumerated types of metadata for a specific set of files. That is, these tools pull structured metadata from documents that are organized in a specific format with specific locations and/or specific tags for the metadata.

However, the strings command pulls all strings from a given file, regardless of the structure. Strings isn't just focused on metadata but can pull various types of information with an unknown structure from any kind of file. The strings command will often find metadata from files that other tools (which do not recognize the structure of the file) cannot find but will bury it in a barrage of output with other uninteresting strings. Searching through the output of the strings command can be difficult, but it often provides some useful tidbits.

We will zoom into two tools commonly used for metadata analysis: one focused on finding structured data in various known file formats (ExifTool) and the other focused on unstructured data (the strings command). We'll cover each tool and then perform a lab using them.



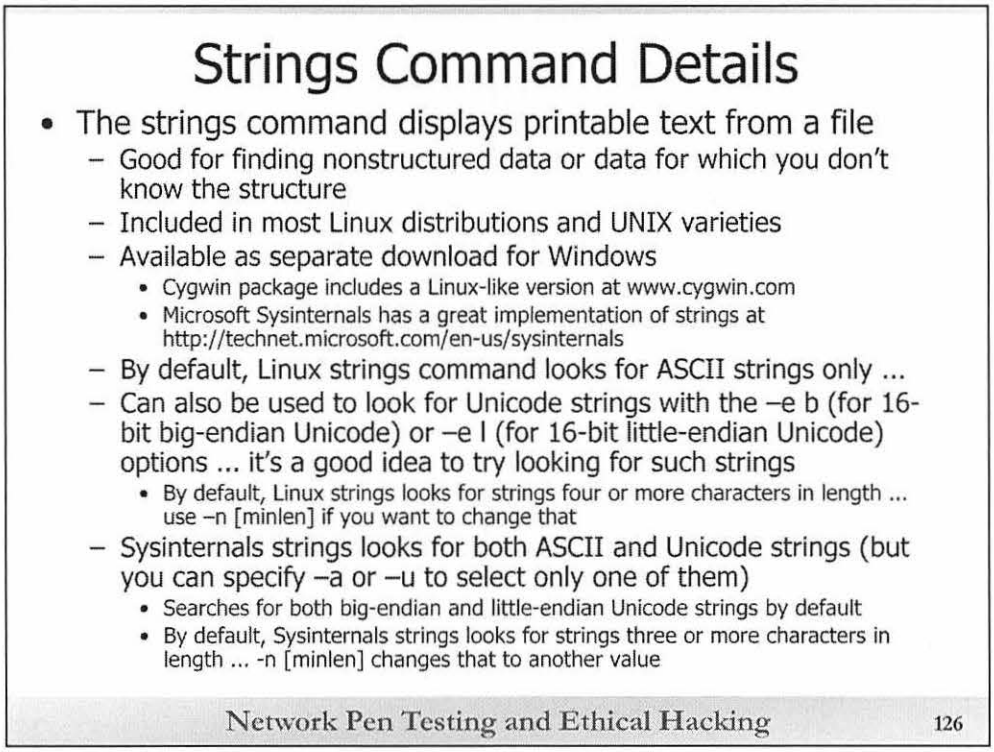
The ExifTool program focuses on reading, writing, or editing the metadata in more than 100 different file types, including images, audio files, videos, office documents (doc, dot, xls, ppt, and more), pdfs, and a multitude of other formats.

Written and freely distributed by Phil Harvey, ExifTool runs on Windows, Mac OS X, and Linux.

When it was first released, the original focus of ExifTool was on image and audio files. For images, it focused on pulling out the camera type and details about the format of the image. It also pulls information about any tools that were used to edit the image or audio. If the image includes geotags indicating the latitude and longitude of where the photo was created, ExifTool retrieves that information.

ExifTool has been significantly extended beyond its original roots in image and audio metadata, now pulling data from the vast majority of file formats a penetration tester is likely to encounter. Of particular interest to pen testers is ExifTool's capability to discern usernames, e-mail addresses, and document editing tools from the files it analyzes.

By default, ExifTool handles one or more files provided to it on its command-line invocation. Alternatively, the tool processes entire directories on the local machine where it runs, handling every file in the directory and can even be set to recurse through a directory structure, analyzing all files it finds.

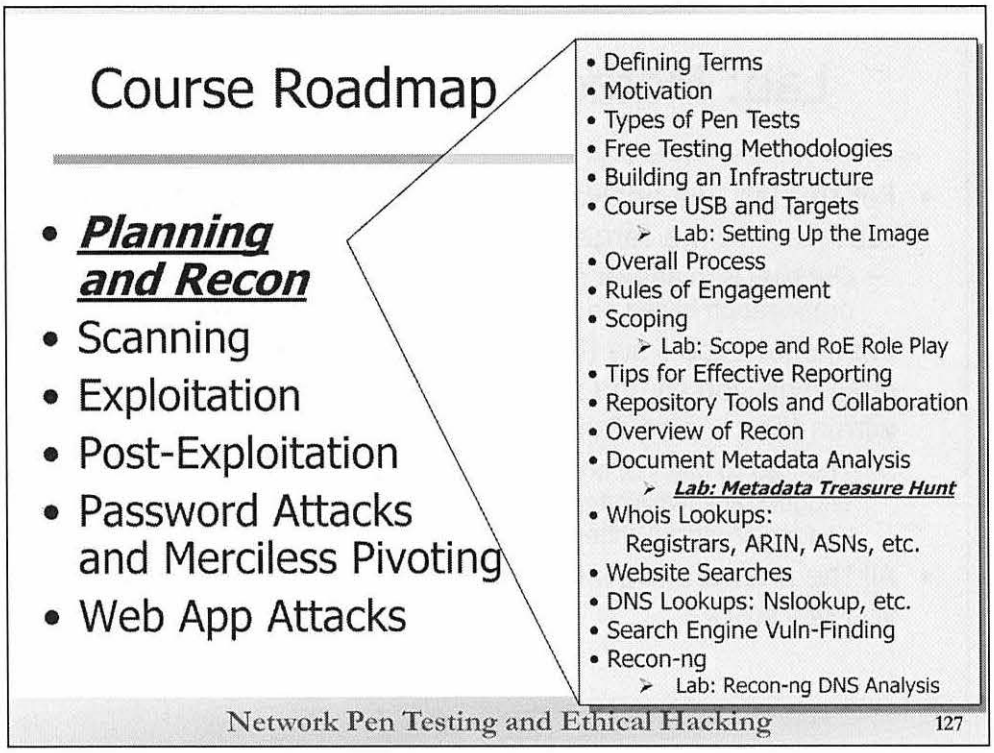


Unlike many metadata analysis tools that focus on structured data, the strings command is useful for finding unstructured data or data for which the structure is unknown. The strings command simply displays printable text from files. It is included in most Linux distributions and UNIX varieties. The strings command is available as a separate download for Windows in a variety of different packages. For example, it is available as a component of Cygwin, the free POSIX environment for Windows available at www.cygwin.com. Or strings is available as a free stand-alone download from Microsoft Sysinternals.

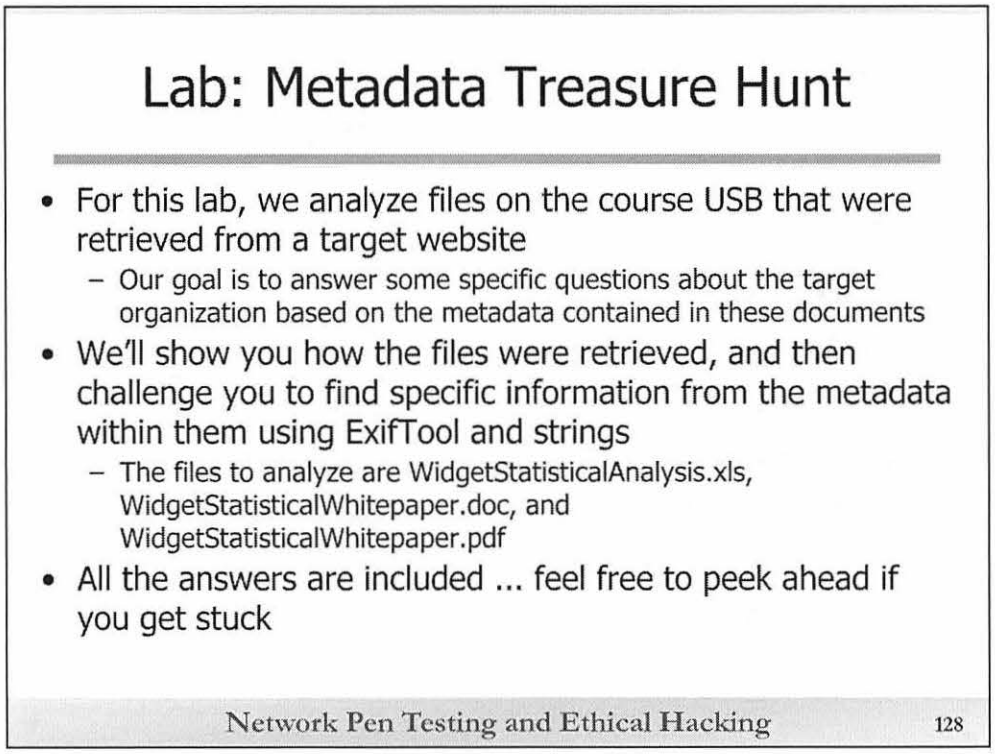
By default, the Linux version of strings looks for printable ASCII strings only. It searches through the file for four or more consecutive ASCII characters and then prints them to Standard Output. To change the default minimum string length, strings can be invoked with the –n [N] option to specify whatever string length the user wants. The default of four characters is reasonable for most uses.

Many document types, especially those associated with Microsoft Office programs (doc, docx, xls, xlsx, and such) store some strings not as ASCII (an 8-bit character representation) but instead as Unicode (a 16-bit character representation). If you run Linux strings with its defaults, it will show you only ASCII strings, and you may miss out on some highly useful information. It's a good idea to run strings multiple times: once with its ASCII default, once with the –e b option (for "encoding" type of big endian 16-bit characters), and once with –e l (a lowercase "L" for 16-bit little endian encoding). Big endian and little endian refer to the way the bytes are ordered for the given string in the file. Most Microsoft document editing tools use little Endian encoding, but sometimes (even in the same file) will store some strings in big Endian format.

The Sysinternals version of strings looks for ASCII, big endian Unicode, and little endian Unicode strings by default (pulling each of those different formats in a single invocation), focusing on strings three or more characters in length. To focus only on ASCII or Unicode, the tool can be invoked with –a or –u, respectively. And to change the minimum character length, we can invoke it with –n [N].



Let's perform a lab now, in which we use ExifTool and strings to harvest metadata from three sample files included on the course USB in a metadata treasure hunt. When conducting the lab, think about how a penetration tester can use the information retrieved from the metadata during the remainder of the penetration test.



In this lab, we analyze three files included on the course USB, searching for truly useful gems of information in metadata by running ExifTool and the strings command on the course Slingshot Linux image.

The goal is to answer some questions about the target organization, specifically searching for information that would be useful in a penetration test.

We show you how the files were originally retrieved; although, you will not have to retrieve the files for the lab, given that they are already included on the course USB in the Slingshot Linux image.

The files to analyze are called

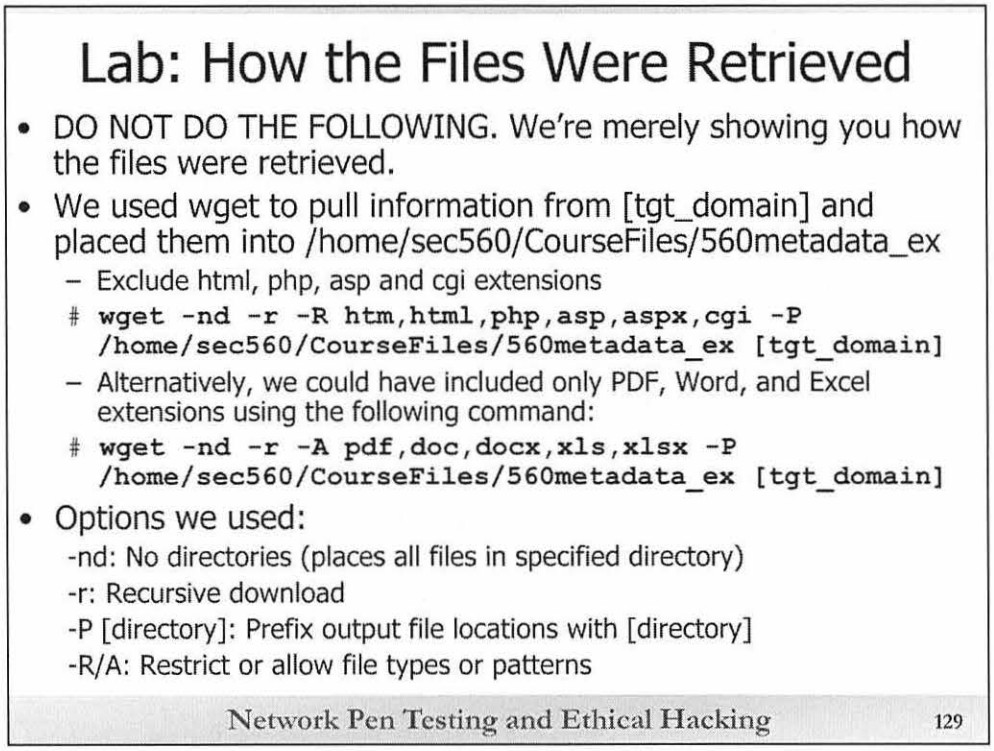
WidgetStatisticalAnalysis.xls

WidgetStatisticalWhitepaper.doc

WidgetStatisticalWhitepaper.pdf

The lab is organized around a challenge-response format. We challenge you to find specific information treasure in the metadata of these files.

You can peek ahead at the answers and the techniques if you get stuck.



YOU DO NOT NEED TO PERFORM THIS STEP. WE ARE MERELY SHOWING YOU HOW THE FILES WERE RETRIEVED TO SET UP THIS LAB.

The files for this lab were retrieved from the target website using the wget command. This command pulls web pages from target web servers and supports a myriad of options. When used with the –r option, wget acts as a web spider, downloading pages from a target site, analyzing them for links, and pulling the pages they link to from the same target site.

When pulling files for document metadata analysis, we could use wget to simply get all files from the target site. However, sometimes we want to focus on only specific types of files. We can tell wget either to omit files of a certain type, getting everything else, with the –R option followed by a list of file types that we don't want to get. Or we could specify a set of file types that we do want to retrieve, with the –A option to allow only certain types.

When pulling these files, we wanted to put them all in a single directory, without mimicking the directory structure of the target website, so we used the –nd option to make wget omit directories from the pulled files. Also, we wanted the files to be placed into the /home/sec560/CourseFiles/560metadata_ex directory, so we used the –P option to add a directory prefix.

The resulting wget command that we used to pull these files for this lab was

```
# wget -nd -r -R htm,html,php,asp,aspx,cgi -P
/home/sec560/CourseFiles/560metadata_ex [target_domain]
```

DO NOT RUN THAT COMMAND HERE. It is just an example of how we could retrieve the files.

Alternatively, we could have specified we wanted only pdf, doc, docx, xls, and xlsx files using the syntax on the slide. In document metadata analysis for real-world pen tests, it is typically better to create a blacklist of file types you do NOT want (the first wget command) rather than creating a white list of types you do want (the second wget command on the slide). With the first command, we may get a file type that we did not expect but which includes particularly juicy information that the second command would omit.

Lab: ExifTool and Strings Metadata Treasure Hunt

- Using ExifTool and strings in the VMware Linux image for the course, analyze these files from /home/sec560/CourseFiles/560metadata_ex/
  - WidgetStatisticalAnalysis.xls
  - WidgetStatisticalWhitepaper.doc
  - WidgetStatisticalWhitepaper.pdf
- First, copy these files into the /tmp directory of Linux:
  `# cp /home/sec560/CourseFiles/560metadata_ex/Widget* /tmp`
- A copy of each file is also included in the Windows directory of the USB, if you want to look at them there
  - But, perform the lab in the VMware Linux image
- Try to answer the questions on the next page by searching through the Metadata
  - Also, remember that you can peek ahead if you get stuck

Network Pen Testing and Ethical Hacking 130

To start the lab, first create a copy of the files in the /tmp directory, so we can perform our analysis:

```
# cp /home/sec560/CourseFiles/560metadata_ex/Widget* /tmp
```

That should copy over the three files you want to analyze:

WidgetStatisticalAnalysis.xls
WidgetStatisticalWhitepaper.doc
WidgetStatisticalWhitepaper.pdf

The goal of this lab is to run ExifTool and strings on each of these files, trying to answer the specific questions on the next slide.

A copy of each of these files is also included on the course USB in the Windows directory. You can open them in Windows and look at them if you'd like, but the lab should be performed in Linux, which has ExifTool and strings installed.

ExifTool can be invoked on the VMware Linux image to analyze a file by running:

```
# exiftool [filename]
```

To run strings against a file, you could simply use

```
# strings [filename]
```

Try this for each of the files, and enter the data you discover that answers the questions on the next page. ALSO REMEMBER THAT YOU CAN PEEK AHEAD AT THE ANSWERS AND THE APPROACH USED TO DETERMINE THEM.

Metadata Treasure Hunt

- Please answer the following questions:
  - What is the full name of user Bob? What is Bob's nickname? Bob Boberson bob the awesome
  - What is Bob's e-mail address? bob.boberson@560gc.tgt
  - What Personally Identifiable Information is located in the spreadsheet (.xls) file? customer information
    - Hint: # `strings -n 8 [filename]` shows strings only eight characters long or longer so you don't get inundated with short strings
  - What information is associated with the organization's firewall rules?
    - Hint: # `strings [filename] | grep -i firewall` shows lines of output with the word "firewall" in a case-insensitive fashion.
    open port 8000 on the window web server · request sandra before lunch.
  - Bonus: If you have extra time, scour the files for all file system paths and URLs
    - Hint 1: To look for a forward slash: # `strings [filename] | grep /`
    - Hint 2: To look for a backslash: # `strings [filename] | grep '\\'`

Network Pen Testing and Ethical Hacking 131

Try to answer the questions on the slide, starting by running exiftool against each of the files as follows:

```
# exiftool /tmp/WidgetStatisticalWhitepaper.doc
# exiftool /tmp/WidgetStatisticalAnalysis.xls
# exiftool /tmp/WidgetStatisticalWhitepaper.pdf
```

What is the full name of user Bob? What is Bob's nickname?
bob boberson / bob the awesome

What is Bob's e-mail address?
bob.boberson@560gc.tgt

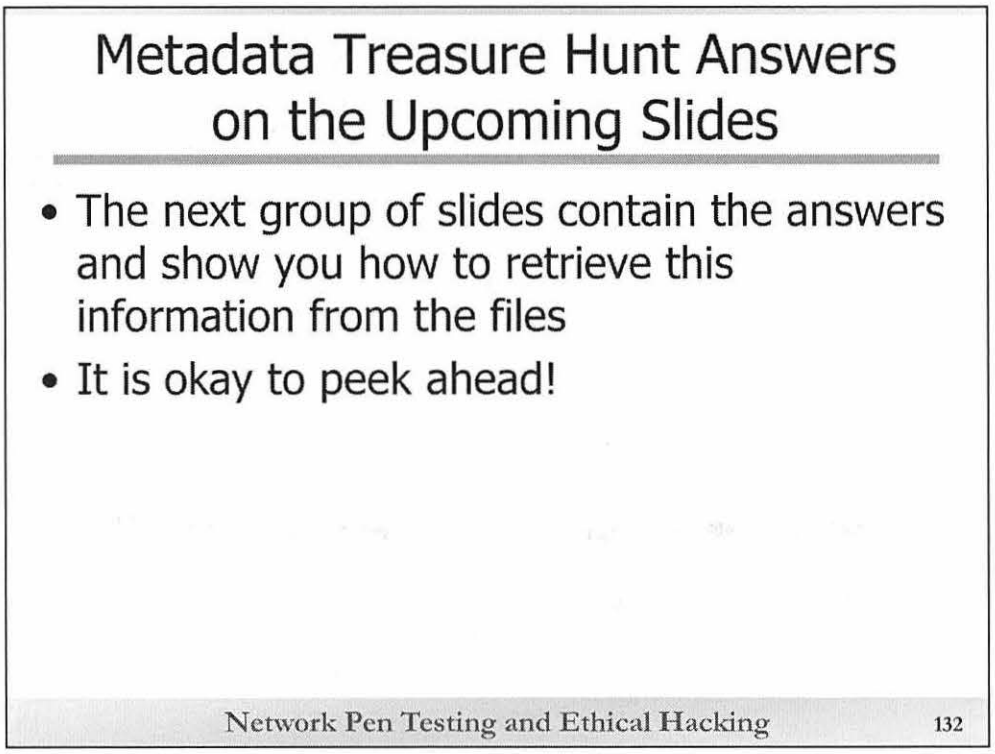
Next, use the strings command to try to find even more data embedded in the associated files.
What Personally Identifiable Information is located in the spreadsheet (xls) file?
Hint: # `strings -n 8 [filename]` shows strings only eight characters long or longer.
customer information

What information is associated with the organization's firewall ruleset?
Hint: # `strings [filename] | grep -i firewall` shows lines of output with the word "firewall" in a case-insensitive fashion.
open port 8000 on web server before lunch.

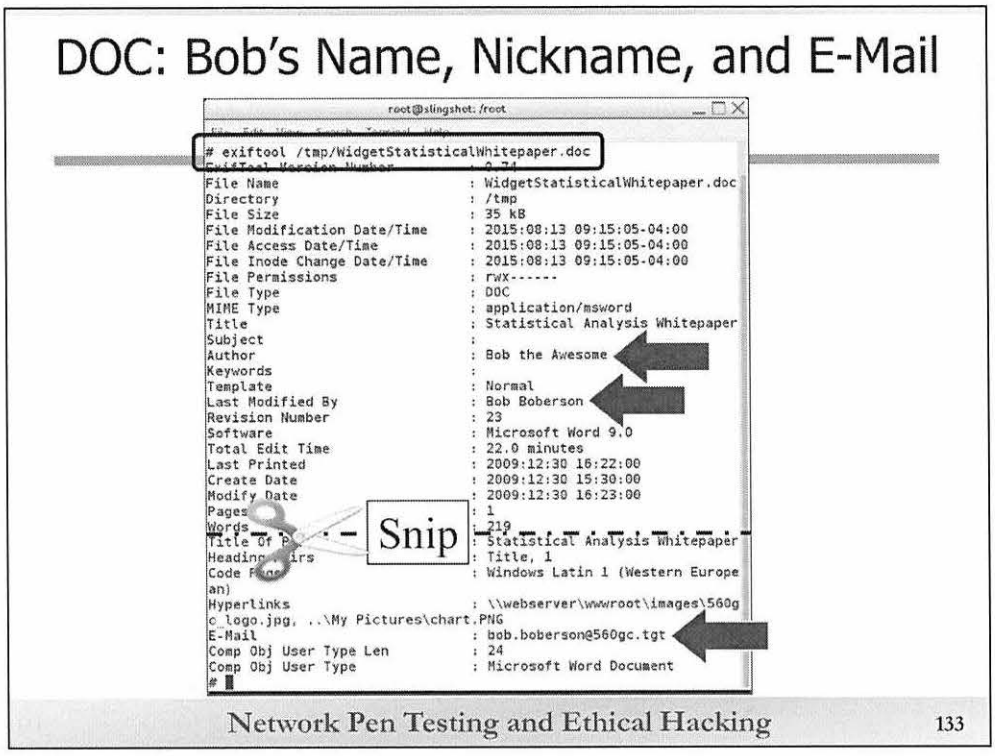
If you have some extra time, also look through the files to find all file system paths and URLs. As a hint, you should consider looking for forward slashes by piping your output through grep to search for a / character. As a further hint, to find lines with a single backslash in them, you could pipe your data through grep '\\'. That syntax will make your shell send a single \ into the grep command.

URLs & File Paths: XLS = #C:\WIND only have 4 sheet
PDF: HTTP://purl.org/dc/elements/1.1
http://ns.adobe.com/pdf/1.3/

ALSO REMEMBER THAT YOU CAN PEEK AHEAD AT THE ANSWERS.



The analysis and answers to this lab start to unfold on the next slide. You can peek ahead if you haven't finished the previous part of the lab and need a hint or inspiration. Or even if you have finished with the lab, you can look ahead to check your work.



Bob created the .doc and .xls files in Microsoft Word and Microsoft Excel, respectively, so we can analyze the metadata of either file to determine Bob's full name and nickname.

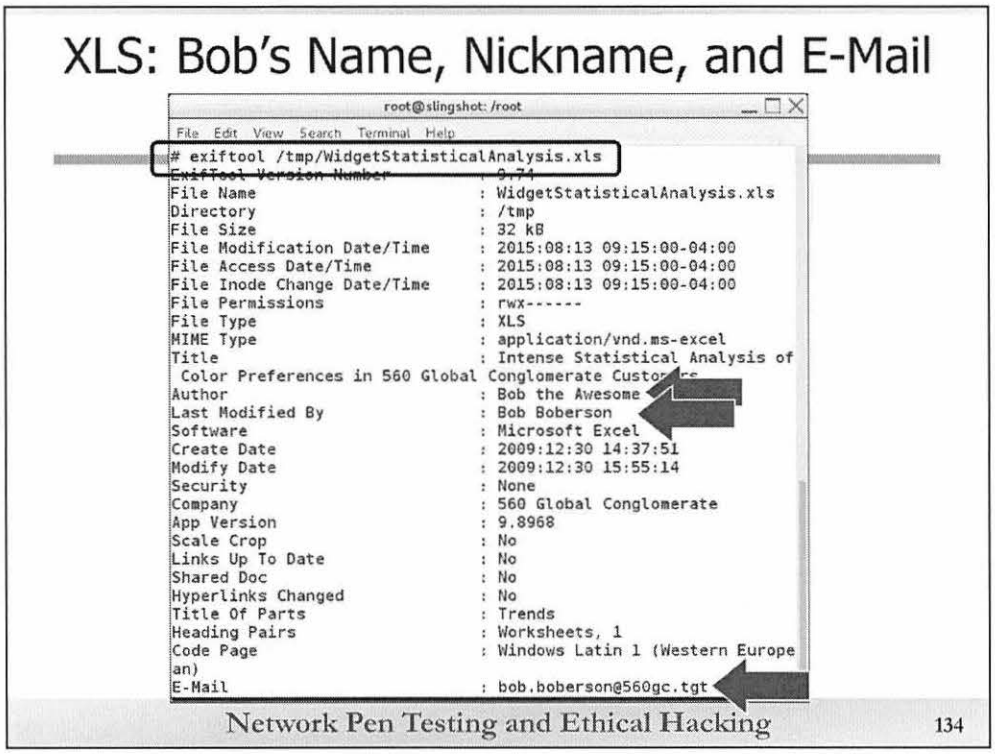
Microsoft Office inserts usernames and author information in specific fields of the files it generates, so we can look for this structured metadata with Exiftool. You can run Exiftool against either the .doc or the .xls file.

```
# exiftool /tmp/WidgetStatisticalWhitepaper.doc
# exiftool /tmp/WidgetStatisticalAnalysis.xls
```

Bob's full name is **Bob Boberson** (from the Last Saved By field).

Bob's nickname appears to be **Bob the Awesome** as indicated in the Author field.

Bob's e-mail address appears to be **bob.boberson@560gc.tgt** as indicated in the E-mail field.



Bob created the .doc and .xls files in Microsoft Word and Microsoft Excel, respectively, so we can analyze the metadata of either file to determine Bob's full name and nickname.

Microsoft Office inserts usernames and author information in specific fields of the files it generates, so we can look for this structured metadata with Exiftool. You can run Exiftool against either the .doc or the .xls file.

```
# exiftool /tmp/WidgetStatisticalWhitepaper.doc
# exiftool /tmp/WidgetStatisticalAnalysis.xls
```

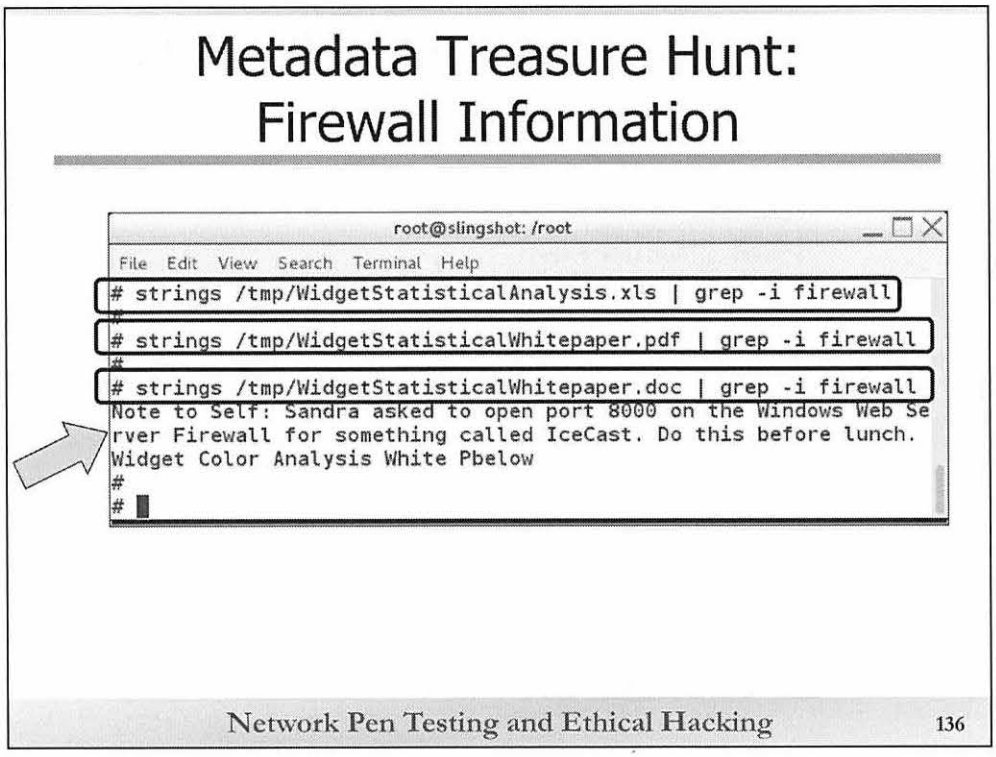
Bob's full name is **Bob Boberson** (from the Last Saved By field).

Bob's nickname appears to be **Bob the Awesome** as indicated in the Author field.

To find PII in the .xls file, we can look for strings of consecutive characters. However, many files are littered with meaningless small strings, so we'll focus our search on longer strings, such as eight characters or more in length. When we do this using the strings command with the –n 8 option, we find some interesting strings in the .xls file, as shown in the slide.

```
# strings -n 8 /tmp/WidgetStatisticalAnalysis.xls
```

In the output, you'll see strings with full names of various people (**Mrs. Boberson, Sally Southers**, and more) along with data that appears to be **Social Security numbers** or some **related government identification number**. This is likely PII that has leaked out of the target organization.



Next, we'll look for information about the firewall of the target organization by running strings and grepping its output for the string "firewall" in a case-insensitive fashion.

You'll note that there are no references to firewall in the output of the command run against the .xls or .pdf files. But, there is a reference in the .doc file.

```
# strings /tmp/WidgetStatisticalAnalysis.xls | grep -i firewall
```

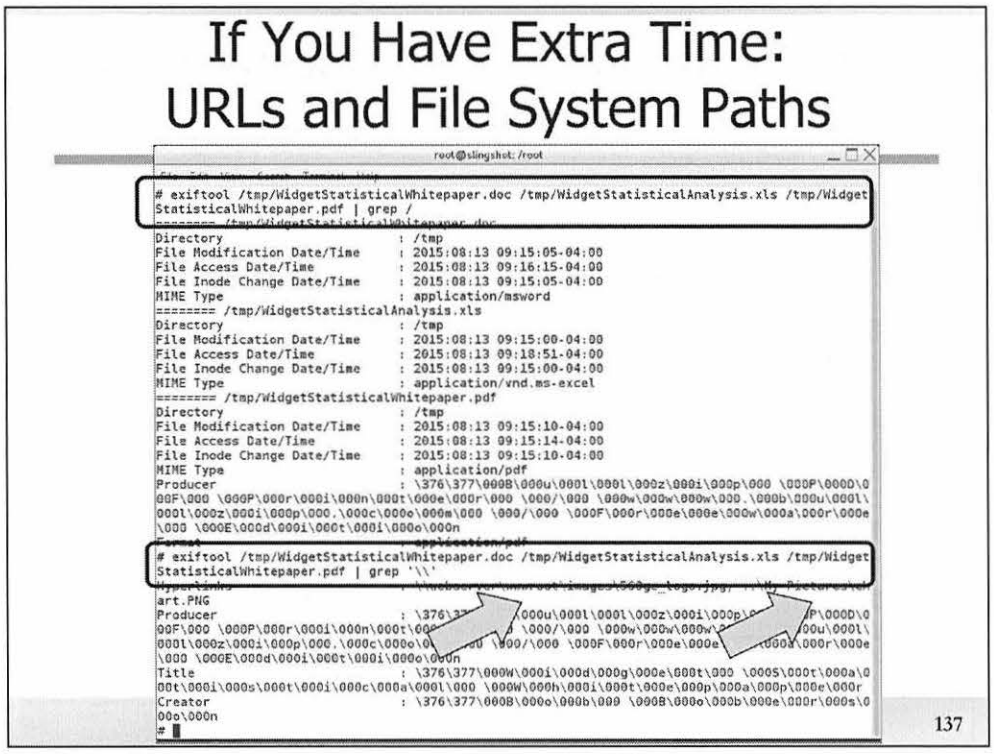
There's no output, which implies that there are no such ASCII strings in this document.

```
# strings /tmp/WidgetStatisticalWhitepaper.pdf | grep -i firewall
```

Again, no output.

```
# strings /tmp/WidgetStatisticalWhitepaper.doc | grep -i firewall
```

Here, we see output that mentions opening up **port 8000 on the Windows Web Server Firewall for a IceCast**, which is a streaming audio service. Bob apparently made this comment to remind himself to take this action before lunch.



If you have extra time, you can look for additional information in the files, specifically URLs and file system paths. These might be useful to a penetration tester who is looking to target specific valuable information assets in a target organization.

File system paths may be structured or unstructured metadata, so we'll look for them using both Exiftool and strings.

We'll start with Exiftool. To make our analysis more efficient, we'll rely on a feature of Exiftool that lets us specify multiple files on the command line, one after another, and the tool will retrieve metadata from all files we specify.

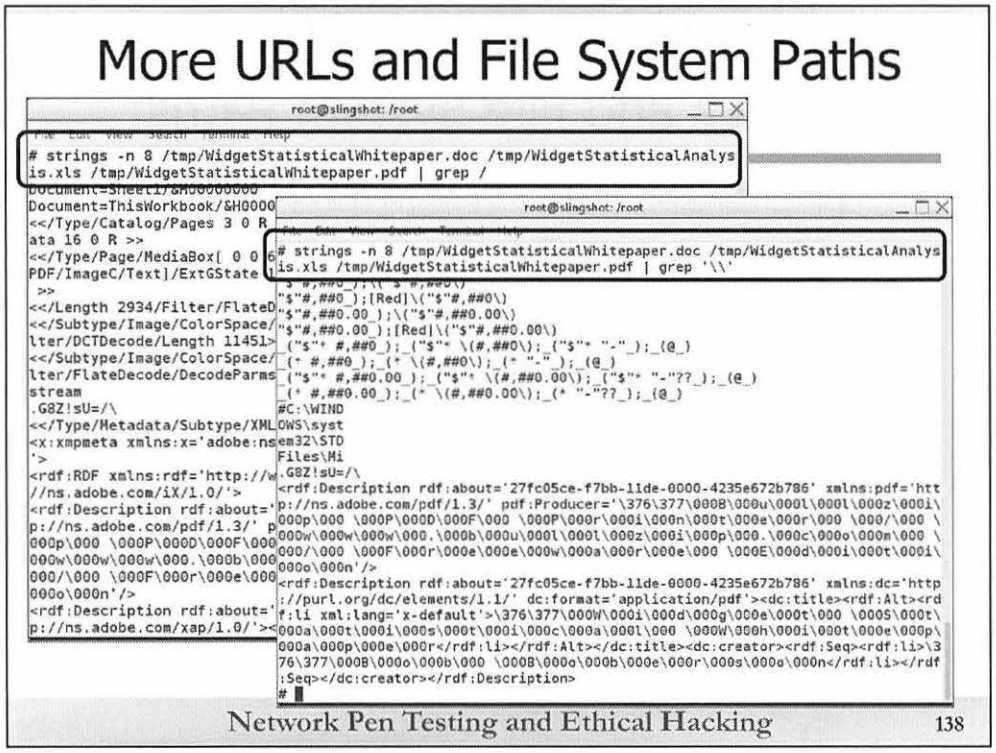
First, let's run Exiftool to look through each of our three files, grepping our output to find slashes (/):

```
# exiftool /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf |
grep /
```

Next, let's look for backslashes. (Sending grep '\\' makes grep look for a single backslash only.)

```
# exiftool /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf |
grep '\\'
```

Here we see file system paths of **\\webserver\wwwroot\images\560gc_logo.jpg** and **..\My Pictures\chart.PNG.**



Next, we'll look for ASCII strings in our files using the strings command, also taking advantage of the fact that strings supports multiple files on the command line. We'll start by searching for strings greater than eight characters (-n 8), looking through the output for the / character:

```
# strings -n 8 /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf |
grep /
```

Here, we see a lot of strings on the output, which includes several URLs: **http://www.w3.org/199/02/22-rdf-syntax-ns#**, **http://purl.org/dc/elements/1.1/**, and **http://ns.adobe.com/xap/1.0/mm/**. These URLs are likely just part of the PDF file and point to items outside of our target scope.

Let's now try our analysis looking for ASCII strings with a backslash in them:

```
# strings -n 8 /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf |
grep '\\'
```

Here, we see some similar strings on our output, again noting http://purl.org/dc/elements/1.1/, which is likely outside of our project scope.

So, our analysis looking for standard ASCII strings didn't prove too useful. Let's look for big endian and little endian Unicode strings to see if we get any more useful information that way.

We'll start by looking for big endian strings eight characters or more in length, which include a slash (/):

```
# strings -n 8 -e b /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf | grep /
```

Our output is empty. Let's look for little endian Unicode strings with forward slashes:

```
# strings -n 8 -e l /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf | grep /
```

Again, nothing. Let's look for big endian Unicode strings with backslashes:

```
# strings -n 8 -e b /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf | grep
'\\'
```

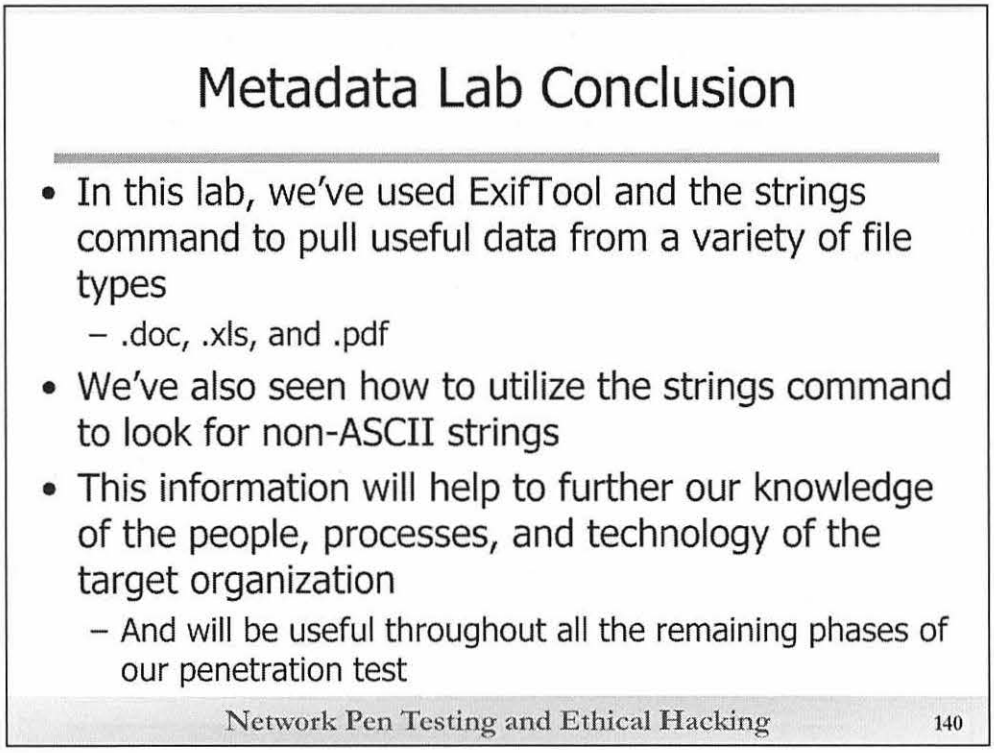
This gives us some useful information. Here, we found a potentially interesting piece of information: a file system path to the original file on Bob's machine:

C:\Users\Bob Boberson\My Documents\WidgetStatisticalWhitepaper.doc.

Finally, let's look for strings with little endian Unicode backslashes:

```
# strings -n 8 -e l /tmp/WidgetStatisticalWhitepaper.doc
/tmp/WidgetStatisticalAnalysis.xls /tmp/WidgetStatisticalWhitepaper.pdf | grep
'\\'
```

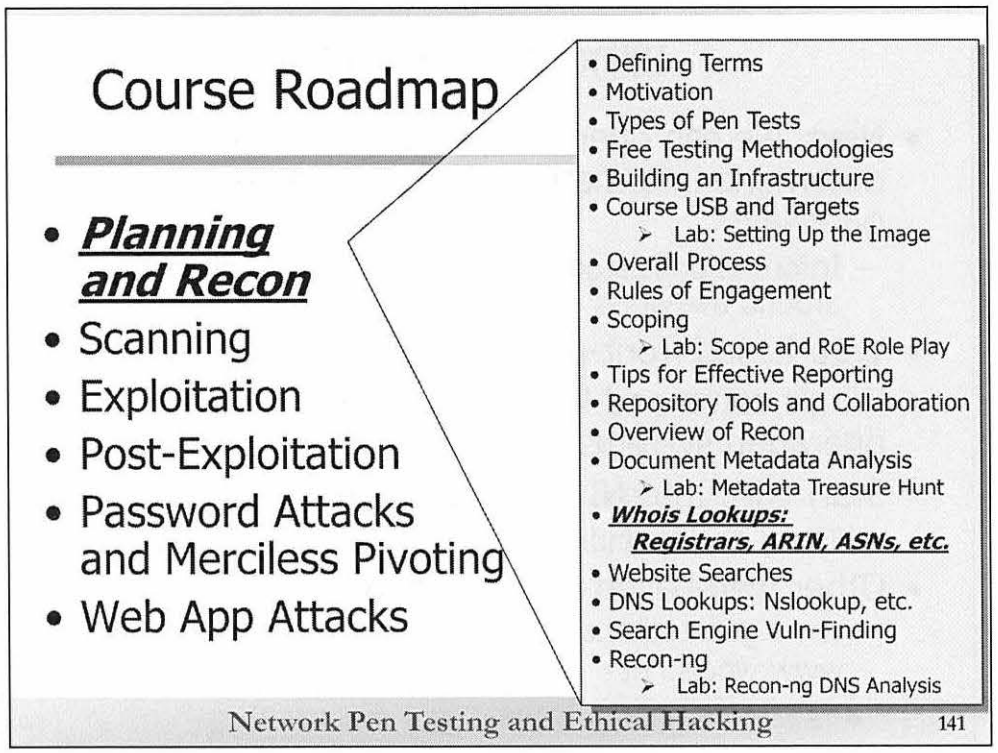
With this one, we've found numerous file system paths, including paths to a webserver file, a Visual Basics for Applications DLL, the Path to Office on the machine, and much more.



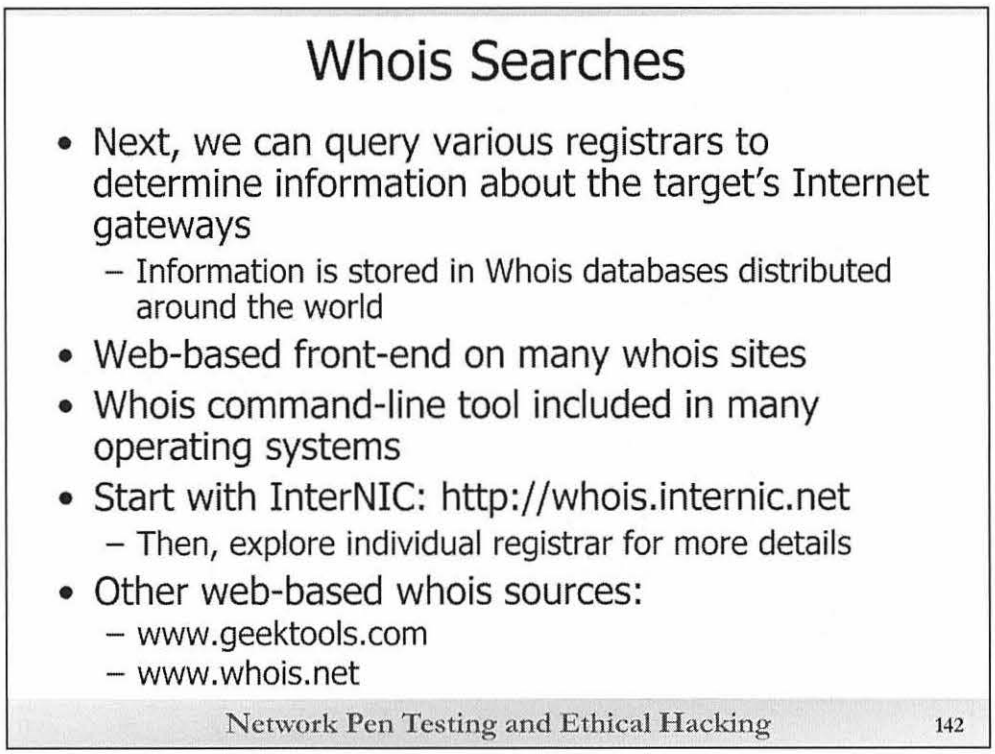
In this lab, we've seen how we can use ExifTool and the strings command to pull data from files that may be useful to us in our penetration test. We've seen the advantages of structured data and ExifTool in pinpointing useful information.

We've also seen the advantages of looking for unstructured data with the strings command to find something that ExifTool isn't designed to show: obscured fields and comments.

We've also seen how to transcend the default limitation of ASCII strings on Linux with the –e option to look for Unicode strings, both big and little endian.



Our first recon step will be to determine more information about the target domain names that we've gathered from our initial scoping information. We can look up domain names in Whois databases maintained by various domain name registrars and related organizations around the world. Our goal with Whois lookups is to find out about the people associated with these domains, to learn about related domains, to identify the Domain Name Servers used to resolve target names, and to determine the target's IP address assignments. All this information and more is in Whois.



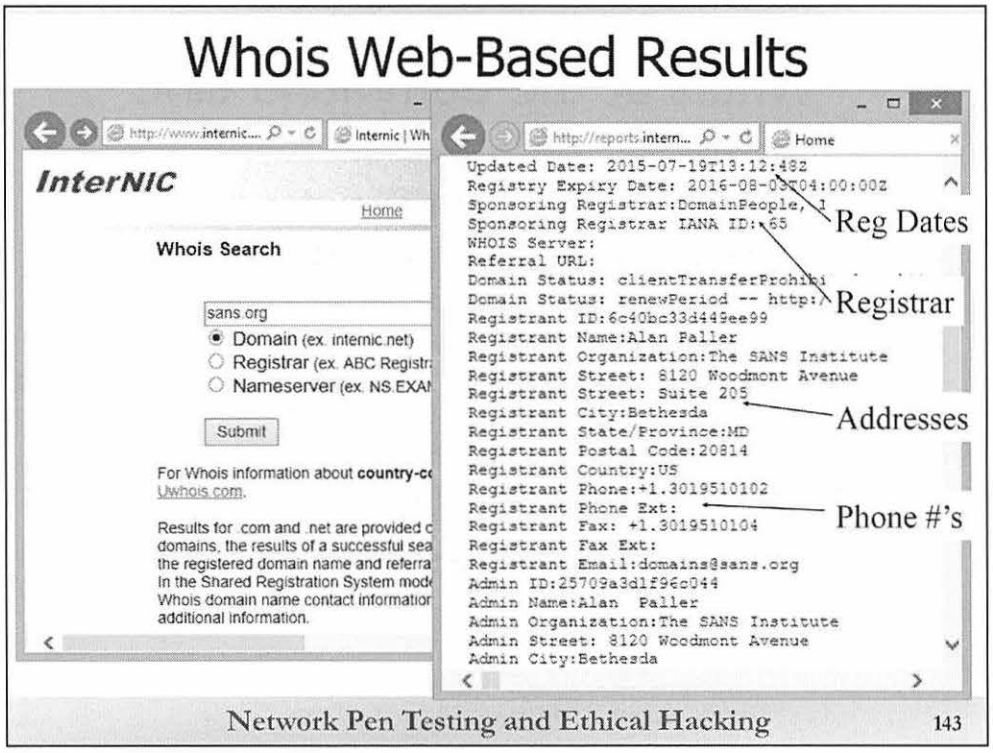
To determine more detailed information about a given target domain, we can look it up using various Whois databases distributed around the world. When a domain is registered, the registrar gathers a significant amount of information about the Internet gateway and people associated with the domain. Most registrars put this information in publicly accessible whois databases. Many of these databases, which are organized in a hierarchical fashion, have a web-based front-end so that they can be accessed via a browser. Alternatively, the whois command built in to some operating systems can be used to formulate a whois query.

To start a whois search using web-based tools, we usually do not know the particular registrar in advance used to register the target domain. Thus, we go to the Internet Network Information Center (InterNIC) Whois front-end and perform a query on the domain. InterNIC will either forward the request to the appropriate registrar and return the information to us or tell us who the registrar is so that we can query that registrar's whois database directly.

Several websites are devoted to getting whois information, each providing a portal that queries various whois servers on the Internet. Some of the most popular include:

www.geektools.com

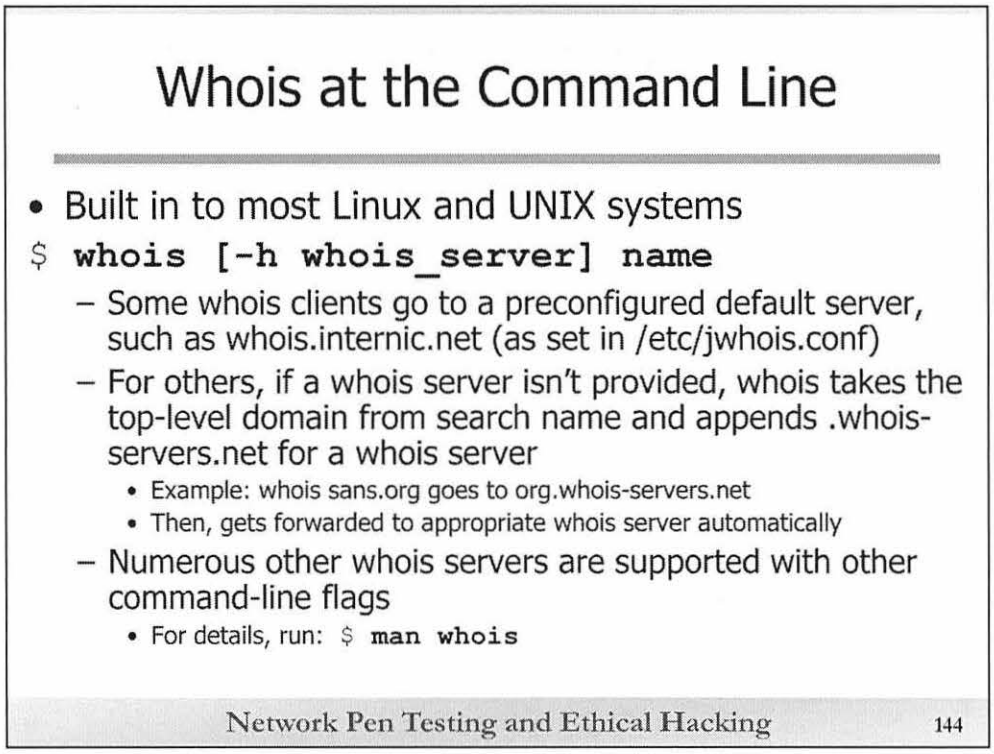
www.whois.net



Here, we have surfed to the InterNIC whois portal and performed a search on sans.org.

The results show us some useful information, including:

- The dates when the domain was registered and updated, as well as when it will expire
- The registrar that was used for the domain
- The name, phone number, postal address, and e-mail address of the Registrant, the Admin, and the Tech contact for the domain
- The DNS servers of the domain, listed in primary, secondary, and tertiary (if applicable) order



Most Linux and UNIX variants have a whois command. In many Linuxes, the whois command actually invokes the jwhois program.

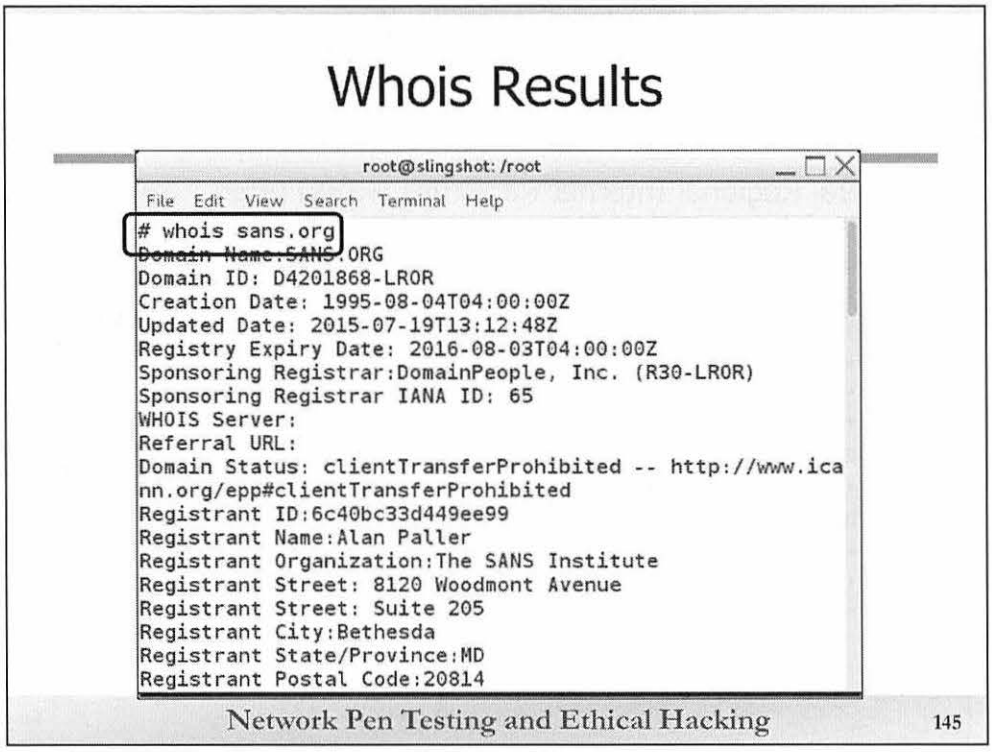
There are many command-line options for whois, but it is most commonly used this way:

```
$ whois [-h whois_server] name
```

The –h option tells the whois client to use a specific whois server to fetch information. If no –h and server are provided, the whois client defaults to a whois server. The default server depends on the specific whois client used. In many Linuxes, the /etc/jwhois.conf file is consulted, which tells the client which server to use based on the top-level domain (.com, .org, .uk, .jp, and such) that is searched for. In many BSD-derived UNIXs, the whois client takes the top-level domain suffix and appends .whois-servers.net to it as a default server for that domain.

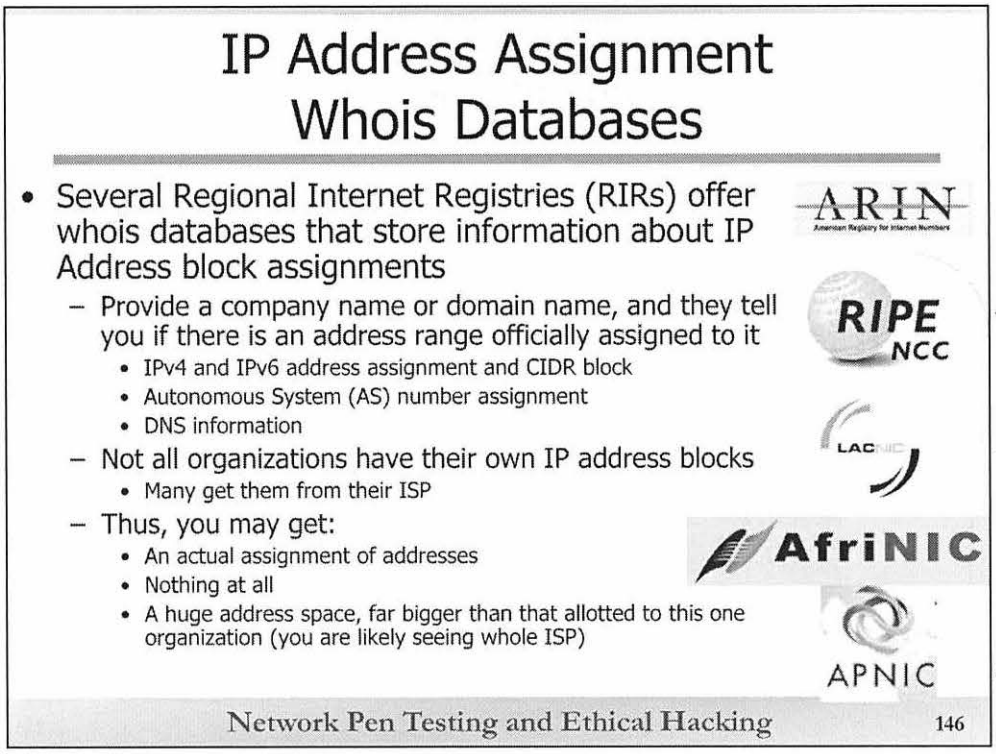
The whois database may then return the information back directly or redirect the request to another whois server.

There are several other command-line flags associated with directing whois clients to different servers, which often vary from system to system. These can be viewed in the whois man page.



In this slide, we've done a whois search from Linux for the domain sans.org.

In the response, we can see the registrars used for the domain (DomainPeople, Inc.). We can also see the name and address of the registrant, along with dates associated with the registration.



Another important element of reconnaissance involves determining the IP address blocks that are assigned to the target organization. There are several Regional Internet Registries (RIRs) that store this information in whois databases. By surfing to the appropriate website, a user can provide a company name or domain name and retrieve official address assignments, including IPv4 and IPv6 addresses. Most records also include the CIDR (Classless Inter-Domain Routing) block, telling us the size of the target network. The American Registry for Internet Numbers (ARIN) covers North America, including the United States, Canada, and certain Caribbean islands. The Réseaux IP Européens Network Coordination Centre (RIPE NCC) is the RIR for Europe, the Middle East, and parts of Central Asia. The Asia Pacific Network Information Centre (APNIC) covers the Asia-Pacific region. The Latin American and Caribbean Internet Address Registry (LACNIC) encompasses Latin America and most of the Caribbean. AfriNIC covers the continent of Africa.

Also, these databases provide autonomous system (AS) numbers, sometimes known as ASNs. An AS is a collection of IP networks and their associated routers under the control of a single technical administrator, such as an ISP or enterprise, that has a common routing policy with respect to the Internet. The AS will have its own internal routing policy but presents a separate routing policy to the Internet, which moves packets between various autonomous systems using a routing protocol, like the Border Gateway Protocol (BGP). Each AS is assigned a unique ASN, which is stored in the Regional Internet Registries. These databases also store DNS information.

Not all organizations have an IP address block assigned to them. Some get IP addresses from their ISP. When searching a Regional Internet Registry for information, we may therefore not get exactly what we are looking for. When searching for an IP address block for a company, for example, we may get the actual results for that company, certainly a good thing. Alternatively, we may get nothing at all, implying that the given enterprise gets all its IP address space from its ISP. Thirdly, we may get a giant block of addresses back that do not apply to only the organization we searched for, but instead apply to its entire ISP. Thus, we have to be careful when targeting the results we receive from a Regional Internet Registry, verifying that they are actually within the scope of our test.

ARIN Lookup

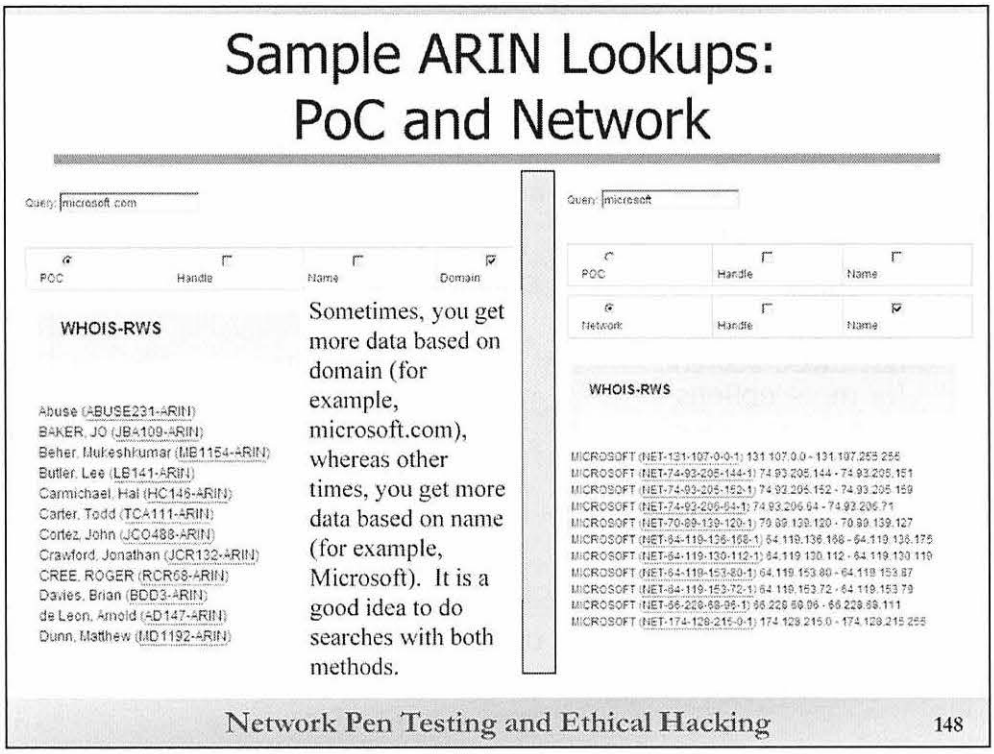
- Specify company name or IP address
- Will return single detailed record if single match
  - Summary records if many matches
  - Click individual summary record for details
- Alternatively, click advanced search for more options:
  - Point of contact
  - Network address space
  - Autonomous system numbers (ASNs)
  - Organization
- Check the Handle, Name, or Domain box based on the kind of data you enter for your search

Let's focus on ARIN for some search examples. By surfing to www.arin.net, we are presented with a field to enter data for a whois search. By providing a company name or IP address, we can get information about the associated organization.

ARIN will respond with summary records if there are many matches, with each summary taking one line and providing a link for more detailed information. If there is only one record that matches our query, we will see the detailed data returned without a summary.

We can focus our searches to look for specific record types within ARIN by clicking Advanced Search. This option brings us to a page which lets us search for points of contact for a provided company or domain, network address space allocated to the organization, Autonomous System Numbers (ASNs) of their groups of routers used in the Border Gateway Protocol (BGP), or organization details. Enter the data you want to search on (such as microsoft.com), select the radio button for the type of data you want to query (PIC, Network, ASN, and such), and then check the box based on the type of data you've entered for your search, such as a Handle (a reference to various objects in ARIN), a Name (such as Microsoft), or a Domain (such as microsoft.com). Click Submit to conduct the search.

whois
└ nslookup
└ whois (ip addr)



The screen shots on this slide show a Point of Contact (POC) and Network search associated with Microsoft. Note that ARIN searching is case-insensitive. For the POC search on the left, we focused on entering in a Domain (microsoft.com) because it gave us more results than searching based on the name microsoft. For the Network search on the right, we received more results by searching for the name microsoft than we did searching for the domain microsoft.com. It is a good idea to do searches for both name and domain to help ensure you get the data you need for your reconnaissance phase.

We can see that Microsoft has a huge number of points of contact and network ranges.

Our next step in the reconnaissance phase will be to learn about the given target organization through searches in publicly available information sources. The World Wide Web is a treasure trove of information, highly usable by us as testers. By searching appropriate websites directly affiliated with the target, as well as third-party search engines, job sites, blog postings, and so forth, we can build a complete dossier of information about the target organization.

As a start of the recon phase, the tester can use a search engine such as Google to learn more about the target organization. In particular, we recommend conducting searches on the target organization's name to gather the following information, which should be recorded in the tester's results:

- **Major businesses**: What is the industry or industries associated with the target? Financial services? Government agency? Manufacturing?
- **Major products or services**: What does the target organization produce? What are the brand names of its products or services?
- **Corporate officers and other VIPs**: Who is most important in the target organization? Who are its leaders? Who is associated with its technical infrastructure?
- **Major competitors**: Who competes with the target organization? What is the target organization's relative performance vis-à-vis its competitors? Is it the market leader?
- **Physical locations**: Where are the major facilities of the target organization?
- **Recent press releases**: What has the target enterprise told the public lately about itself? What does it consider important from an image and marketing perspective?

Look for Open Job Requisitions

- Job requisitions can help us get information about the information technology products used in a target organization, such as:
  - Web server type
  - Web application dev environment
  - Firewall type
  - Routers
- Google searches to find job reqs
  - site:[companydomain] careers
  - site:[companydomain] jobs
  - site:[companydomain] openings
- Also, searches of job-related sites
  - www.monster.com - Search on Info Tech and Internet/E-commerce

Network Pen Testing and Ethical Hacking 151

Most organizations have job requisition information available on the Internet, as they look to hire new staff. These job requests often contain detailed information about the technical environment of the enterprise. For example, if the target organization is looking for IIS administrators, we now know something about the web servers it uses. If it seeks skilled Checkpoint firewall admins, we have information about at least some of its firewalls. If it is looking for developers with Cold Fusion experience, we now know a little more about some of its web applications. What's more, if the job req is still active, we know that the target organization does not have enough experienced staff members to handle that part of its infrastructure. After all, if it did have the expertise already in-house, why would it be seeking to hire people with those skills?

To search for job requisitions, you could use Google with the "site:" directive focused on the target's domain, followed by common terms used on pages for hiring. We recommend searches like the following:

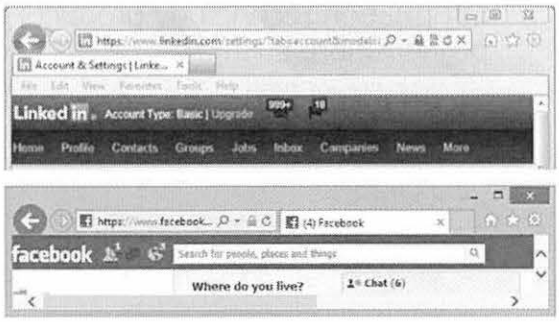
```
site:[companydomain] careers
site:[companydomain] jobs
site:[companydomain] openings
```

You can narrow down your results further by inserting terms such as Information Technology, Internet, e-commerce, firewall, and so forth.

In addition, a thorough tester should look for job reqs on various job-hunting sites, such as Yahoo's Hotjobs.com and Monster.com. Both of these sites let you search based on categories of jobs. For Hotjobs, both the Technology and the Telecomm categories are useful searches for our purposes. On Monster.com, the Information Technology, Internet/E-commerce, and Telecomm categories are helpful. You can even narrow down your searches based on geographic areas.

Searching for Relevant People

- Social Networking Sites
  - Look up target organization
    - LinkedIn
    - Facebook
    - Twitter
    - Google+
    - Pinterest
    - Myspace
    - Orkut

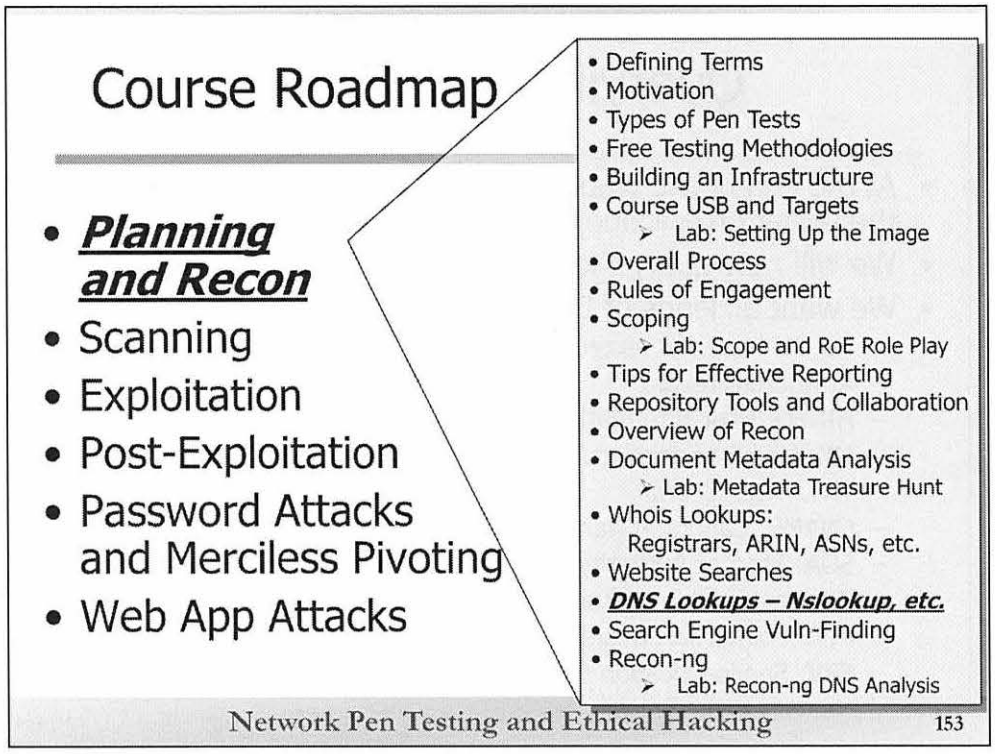


  - Great insights on technology in use, people's relationships and potential source of password lists from profiles

Network Pen Testing and Ethical Hacking 152

Other helpful areas to search are social networking sites. People put a significant amount of information about themselves on these sites, often including where they work. That employer information is exactly what we are looking for. By searching within the social networking site for people who work for the target enterprise, we can then focus in on their background and skill set. We may find, for example, that John Doe used to work in the organization ABC Widgets. Looking at John's profile, we may find out that he has developed applications that involve Microsoft's SharePoint Server product. Those are useful nuggets of information.

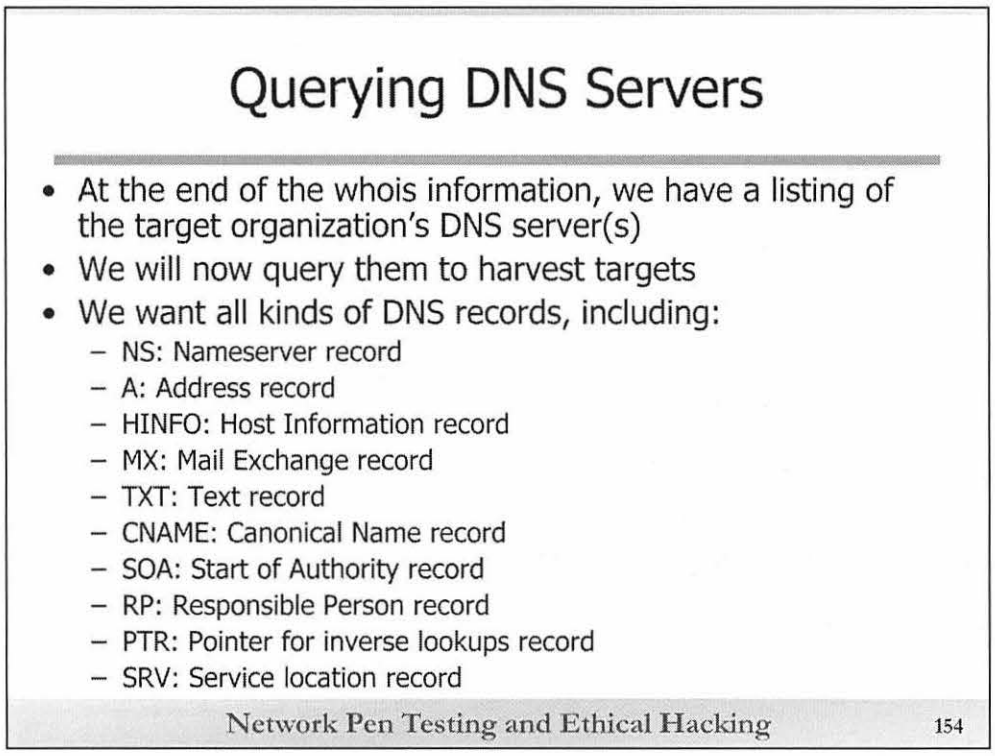
LinkedIn and Orkut, with their greater appeal to professionals, often contain the best information for our purposes, but a quick search on MySpace and Facebook may also reveal useful data. Twitter is also useful in finding people associated with a given target and learning about their interests and work habits.



Now that we've retrieved a list of DNS servers associated with the target from our Whois lookups, we want to query those servers to gain an inventory of potential target machines associated with the given target domain(s).

The techniques we'll cover next will likely identify numerous systems that are directly and indirectly associated with the target. Be careful before moving beyond the recon phase with respect to each of these servers. Don't just start scanning them! It is highly worthwhile to conduct a status call with target organization personnel to make sure that all machines identified throughout the recon phase, and especially those discovered during the DNS interrogation phase, are within the testing scope.

You may find that some systems identified in this phase are outside of the scope but are unknown to personnel in the target organization before the test begins. That's why it is crucial to verify their inclusion in or exclusion from the test's scope.



The last elements of the whois record include the Domain Name System (DNS) servers associated with the target organization, listed in the order of primary, secondary, and tertiary (if it exists) DNS servers. We will next try to harvest records from those name servers.

Name servers are focused on resolving domain names into IP addresses, but that isn't their sole function. They also indicate which machines are mail servers for a given domain, among other useful information. DNS servers house a variety of different records, including

- **NS**: Nameserver record, which indicates the name servers associated with a given domain.
- **A**: Address record, which maps a domain name into an IP address.
- **HINFO**: Host Information record, which associates an arbitrary set of information with a domain name, formerly used to indicate system types.
- **MX**: Mail Exchange record, which identifies the mail servers for the given domain.
- **TXT**: Text record, which includes an arbitrary text string for the domain.
- **CNAME**: Canonical Name record, which indicates aliases and alternative names for a given host.
- **SOA**: Start of Authority record, which indicates that a server is authoritative for that DNS zone (set of records).
- **RP**: Responsible Person records, which are informational, not functional (that is, they have no impact on DNS functionality) and indicate the human responsible for a given domain (seldom used).
- **PTR**: Pointer for inverse lookups records, also called a reverse record, indicating an IP address to domain name mapping.
- **SRV**: Service location records, which provides information about available services, including port and hostname (seldom used).

SPF → stop spam.
MS → ?
↓
TXT record.

The nslookup Command

- The nslookup command is included in modern Windows, Linux, and UNIX systems
- Can type nslookup followed by domain name
- Or type nslookup to invoke nslookup in interactive mode
- We will query primary, secondary, and tertiary servers found with whois search

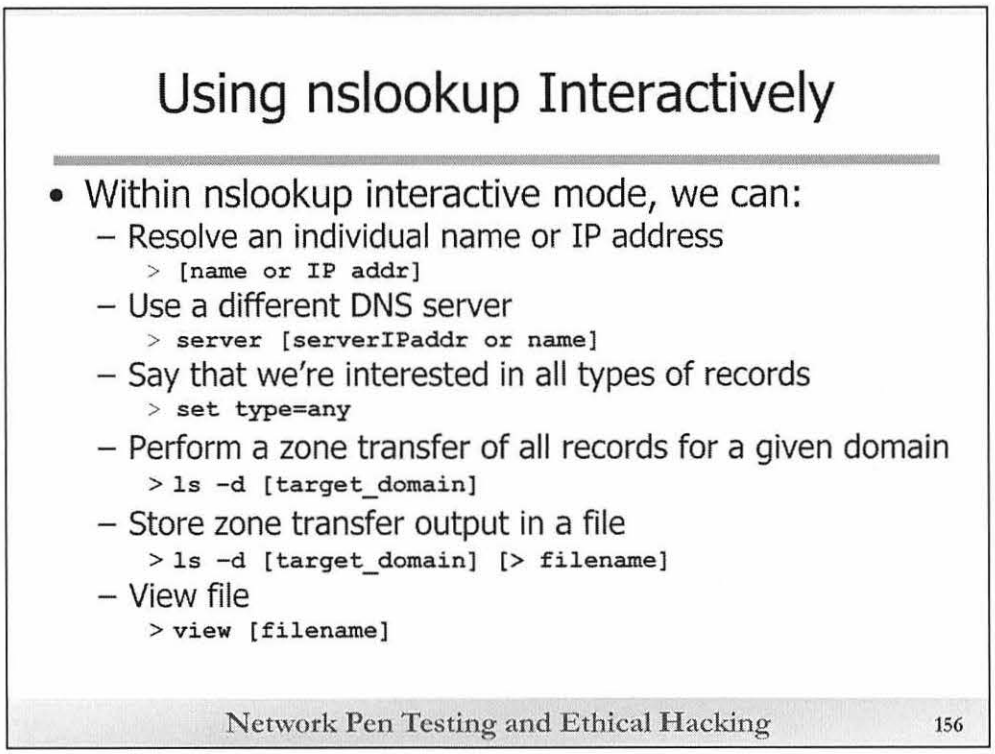
To query DNS servers, we can use the nslookup command built in to modern Windows systems. It is also included in most Linux and UNIX variants.

Nslookup can be used in two ways. First, we could simply type nslookup followed by the domain name that we want to query. The nslookup command will use the local operating system settings to determine a name server, to which it will submit the request, displaying the results.

Alternatively, we could use nslookup in interactive mode, by running nslookup by itself and pressing Enter. Then, we are given an nslookup prompt >, into which we can type names for resolution or directives to control nslookup's configuration. We can redirect nslookup from within this prompt to use other DNS servers.

In the screen shot on this slide, we show both types of invocation for nslookup, used in both ways to resolve www.sans.org.

Next, we'll try to harvest records from DNS, querying the primary, secondary, and tertiary DNS servers that we learned from our whois lookup.



In nslookup's interactive mode, we can simply resolve a name by typing it at the > prompt.

To tell nslookup to use a different DNS server, we could use this syntax:

```
➢ server [serverIPaddr or name]
```

By default, nslookup tries to pull Address records. We are often interested in other record types, such as MX. We can use the "set type=" directive to look for other kinds of records. If we want all kinds of records for a given domain, we can type:

```
> set type=any
```

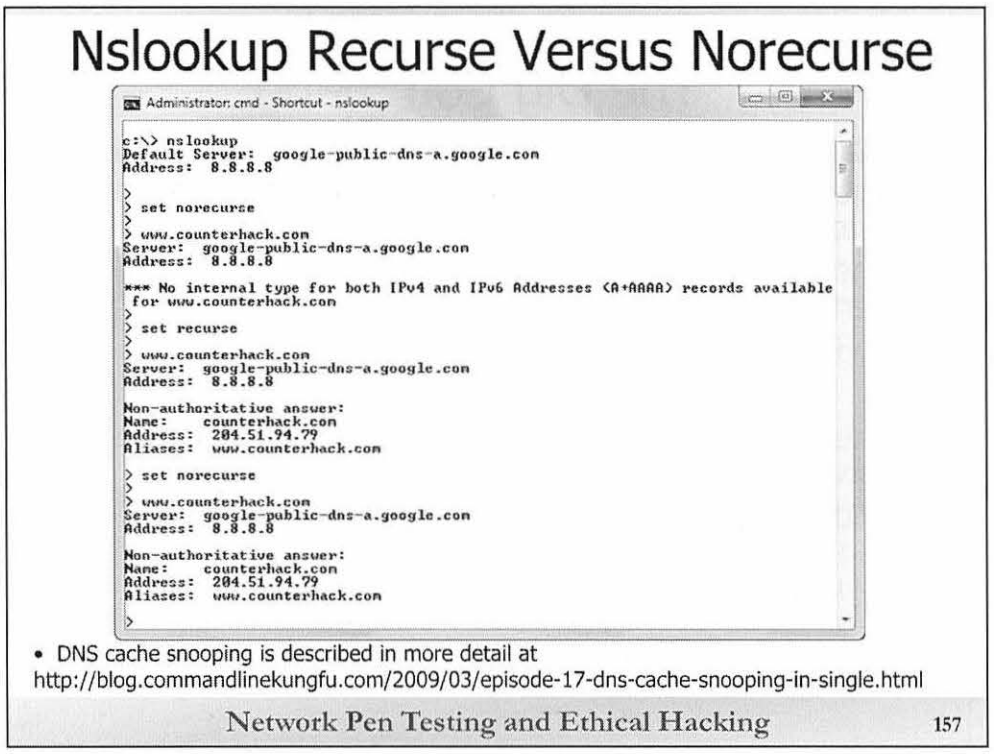
A zone transfer asks the DNS server to transmit all records it has for a given domain. If the DNS server supports zone transfers from the source IP address where we are running nslookup, a complete list of records will be displayed on the screen.

```
>ls -d [target_domain]
```

To place the zone transfer results in a file, we could simply redirect its output into a file (> filename). We can display this file from within nslookup with the view command, as follows:

```
➢ view [filename]
```

Zone transfers were designed so that secondary DNS servers could update their records from primary name servers. DNS zone transfers are carried over TCP port 53, whereas most DNS queries and responses rely on UDP port 53. Many DNS servers block zone transfers from arbitrary locations on the Internet, either by being configured to allow them only for certain addresses (the primary name server) or by a firewall blocking TCP to port 53.

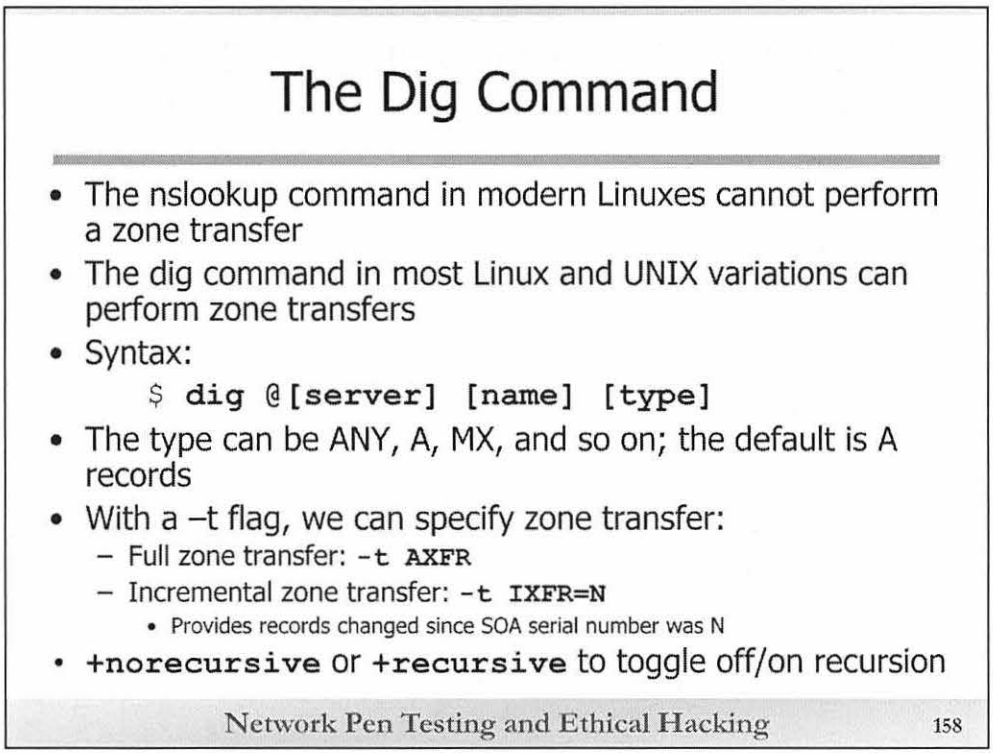


If a DNS server does not have the information we request, it can forward that request to other DNS servers to retrieve the information in a process known as a recursive lookup.

By default, nslookup will ask for recursion from the name servers it queries (the recursion desired bit [RD] in the DNS query is set to 1). The nslookup command can be configured to create queries that do not request recursion using the set norecurse syntax, which sets the RD bit to zero.

With the norecurse option set, we can determine the records that a DNS server has loaded in its cache. In the screen shot here, we have set the norecurse option and tried to resolve www.counterhack.com. We are given back a message saying that no records are available, implying that the name server didn't have the information we requested. Then, we use the set recurse option. We again tried to resolve www.counterhack.com, getting our results because the target name server used recursion to find our answer. We then ran set norecurse. When we tried to resolve the name again, we got our answer back because it was now loaded into the DNS server's cache.

This technique of investigating what a given target DNS server has cached is known as *DNS cache snooping*. The technique is described in detail by Ed Skoudis and Hal Pomeranz at the Command Line Kung Fu blog, where they show how to use this technique via a single command in both Windows and Linux.



In many recent Linux and UNIX systems, the nslookup command has been altered so that it can no longer perform zone transfers. On these systems, we can use the dig command for various kinds of DNS research, including zone transfers.

The dig command has the following syntax:

```
$ dig @[server] [name] [type]
```

The types we can specify include the abbreviations listed earlier, including A, MX, SOA, and such. To receive all kinds of records, we use the ANY type. If no type is specified, dig defaults to A (address) records.

To get dig to perform a zone transfer, we invoke it with the –t AXFR notation, as

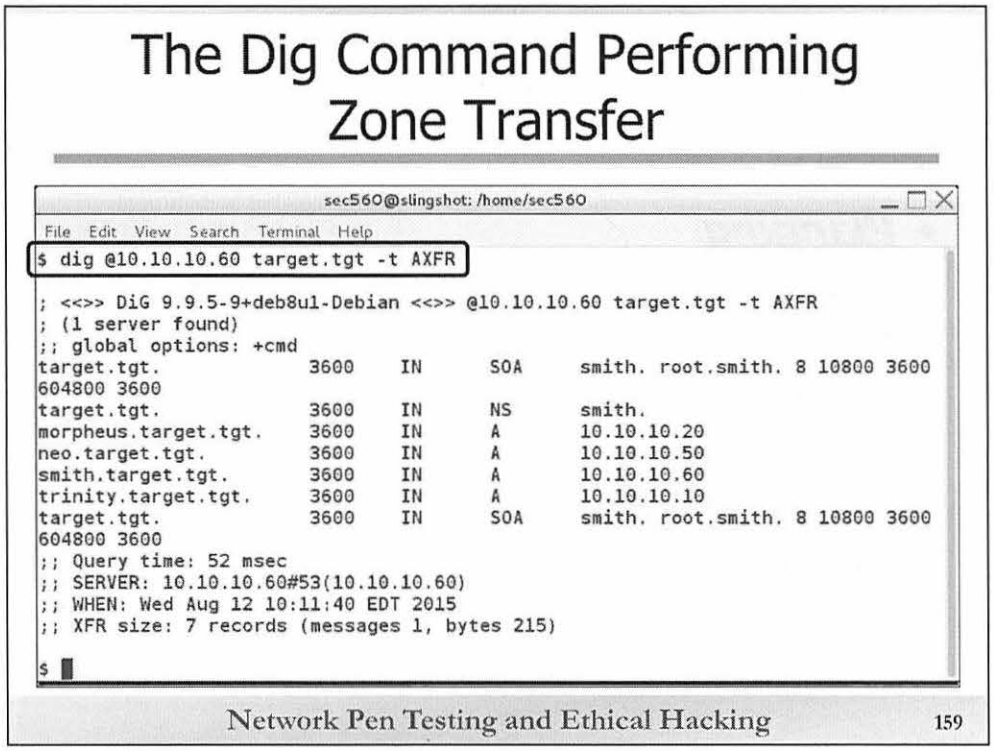
```
$ dig @[server] [domain] -t AXFR
```

This syntax will pull all information about a given domain. Alternatively, dig can perform an incremental zone transfer, pulling only recently updated records, using this syntax:

```
$ dig @[server] [domain] -t IXFR=[N]
```

N is an integer that refers to the serial number of a Start of Authority record. The incremental zone transfer request will pull all records that have changed since the SOA serial number was the N we specified in our dig request.

Dig also supports turning on or off the Recursion Desired (RD), with the `+norecursive` or `+recursive` syntax. By default, dig performs recursive searches.



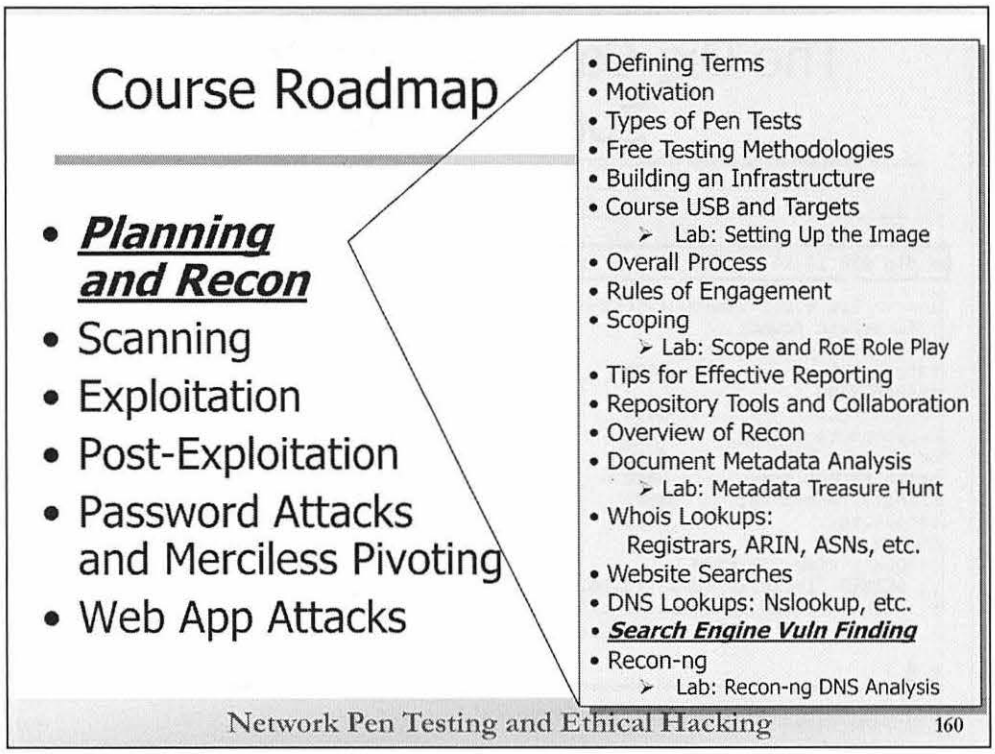
In this screen shot, we've directed the dig command to contact the name server on 10.10.10.45, asking it for information about the target domain called target.tgt. We've requested a zone transfer (-t AXFR). Putting this all together, we get the following command:

```
$ dig @10.10.10.45 target.tgt -t AXFR
```

-> good for internal pentest

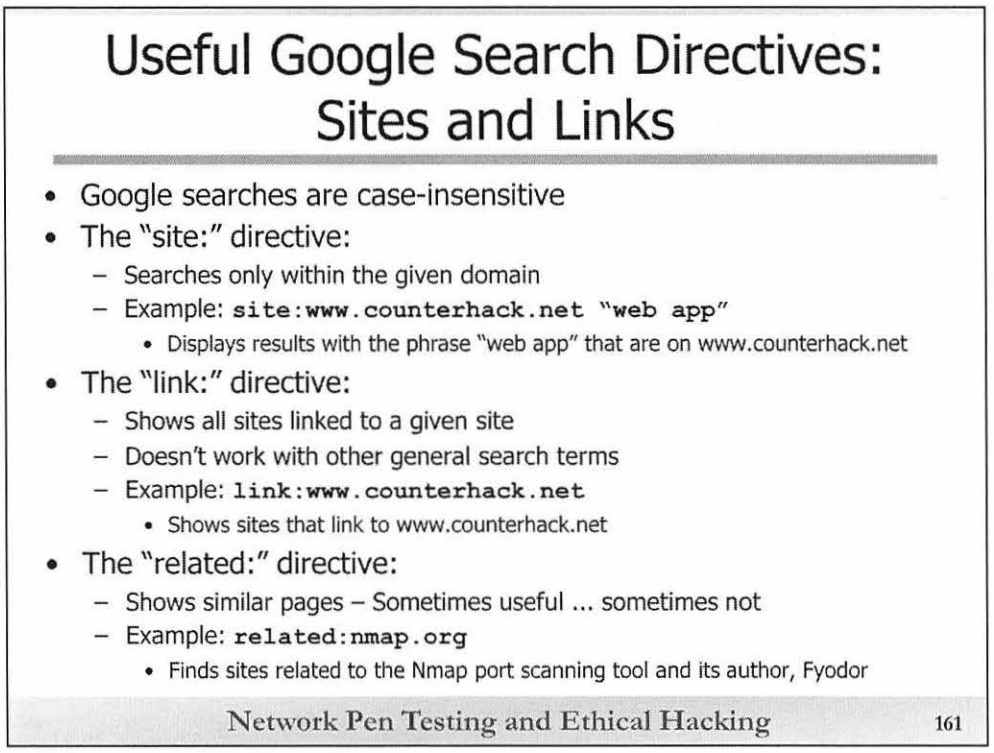
The output shows a good deal of information about potential target systems in the target.tgt domain.

Robtex - com



Our next step will be to use publicly accessible search engines to look for signs of vulnerabilities on systems. Google, Yahoo, and Microsoft's Live Search all contain a great deal of information that could indicate the presence of vulnerabilities in systems associated with the target environment. By sending the appropriate queries to the search engines themselves, we may identify vulnerable systems without actually sending any packets to those systems directly.

We'll focus on Google because of its comprehensiveness and widespread use. Similar techniques can be applied to other search engines, but their specific syntax often differs and is less powerful than Google's.



To understand how we can use search engines (and particularly Google) to find vulnerable systems, we are going to spend some time looking at more advanced search capabilities within Google. By taking the various search principles we'll discuss over the next several slides and combining them together in creative ways, we can use Google to find flawed systems in our target environments.

Let's explore directives associated with examining specific websites and domains. The "site:" directive allows an attacker to search for pages on just a single site or domain, narrowing down and focusing the search. For example, if you want to search pages only in the counterhack.net domain for the occurrence of the string "web app", you could do a search for `site:counterhack.net "web app"`. Have you ever seen a website that doesn't have its own built-in search capability (like www.counterhack.net), or one with a lame search capability? With Google's "site:" directive, you now can use Google to search for results just associated with that site, relying on the power of Google and its flexible search directives and operators. The "site:" directive can get specific, like `site:example.exampleuniversity.edu`, or broad, like `site:.edu`, which would cover all sites with a .edu suffix.

The "link:" directive shows sites that link to a given website. During recon, this directive can be used to find business partners, suppliers, and customers. To look for everything linking to www.counterhack.net, you'd do a search on `link:www.counterhack.net`. Note that "link:" searches look for links that match exactly the given domain name used in the search. That is, a search for `link:counterhack.net` would look for links of the form "http://counterhack.net/[whatever]", but would *not* identify links to the website www.counterhack.net. For those, you'd have to search for `link:www.counterhack.net`.

The "related:" directive shows pages that have similar content and links to the searched page. Based on Google's patented Page Rank algorithm for analyzing pages, this search directive could return information about subtle business relationships that might be missed in other search types. For example, if a given target bank has a business partnership with another bank, there may be no direct links between the sites. However, both may contain similar text terms, and they both may link to the same series of third-party sites. The "related:" directive then gives the tester a chance of finding out about these relationships.

Useful Google Search Directives: Page Titles and URLs

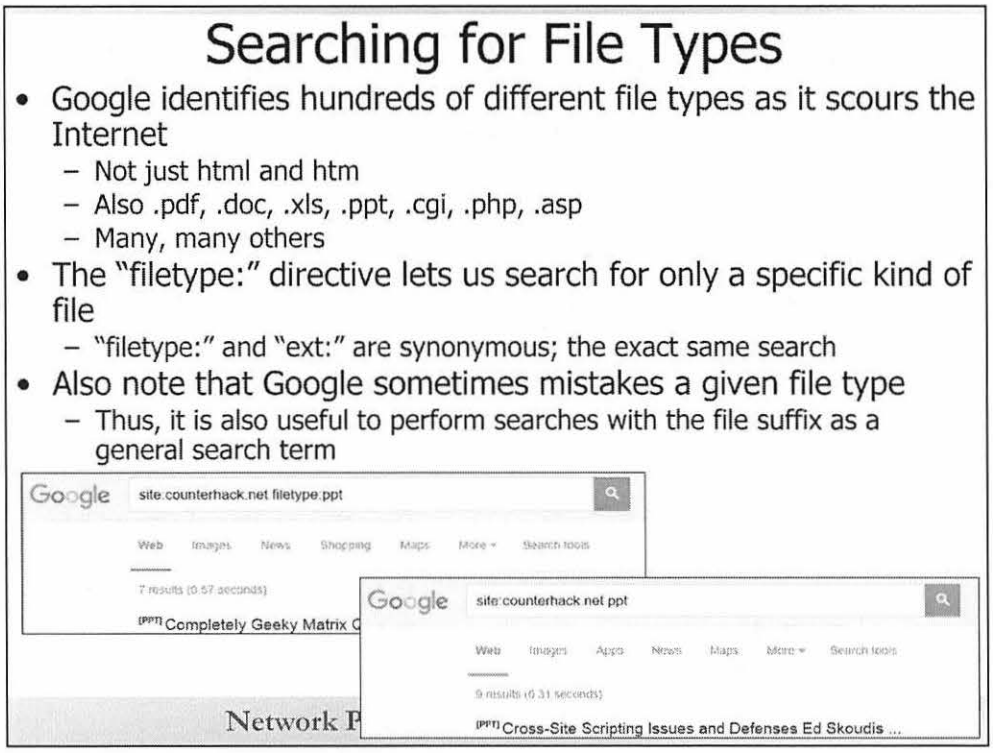
- The "intitle:" directive:
  - Shows pages whose title matches the search
  - Ex: `intitle:index.of passwd`
    - Finds indexed web directories with the word "passwd" in the directory listing, possibly an /etc/passwd file

- The "inurl:" directive:
  - Shows pages whose URL matches the search criteria
  - Example: `inurl:viewtopic.php`
    - Finds a script included in phpBB, a set of scripts for running a web-based forum, with a history of significant flaws

Network Pen Testing and Ethical Hacking 162

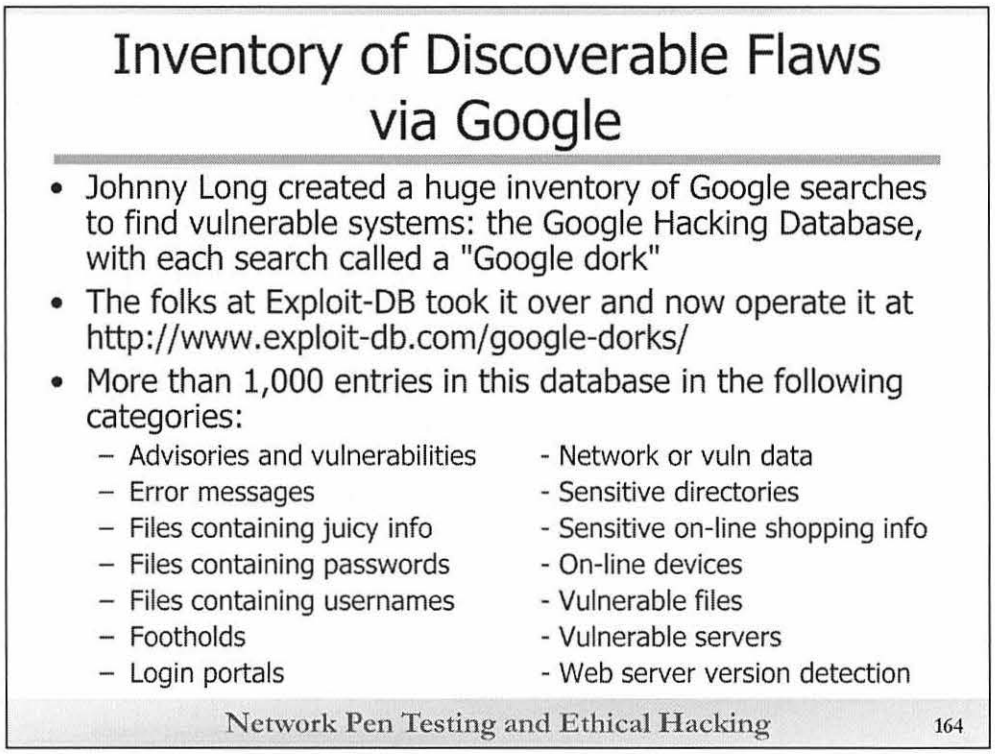
Next, let's look at searches associated with page titles and URLs. Often, we want to find pages with specific text in their titles. The title is the name of the page, often displayed at the top of the browser window. We can perform searches within this title text by using the "intitle:" directive. One of the most useful title types to look for involves directory indexing on a website. Some web servers are configured to allow users to send requests not only for individual pages (for example, index.html), but for directories (for example, /exampledirectory/). On websites configured with directory index functionality, the web server will return to the browser the contents of that directory, with an auto-generated page with a title that includes the text "Index of". Thus, we can search for `intitle:index.of passwd` to look for directories that have a file in them called passwd. We may find an /etc/passwd file. Legitimate /etc/passwd files will have lists of user accounts on the target machine. If shadow passwords are not used, it also might have encrypted passwords suitable for cracking. Note that a number of the passwd files in interesting domains discoverable by Google searches are either honeypot /etc/passwd files or sites that may try to exploit a browser surfing there! So, be careful when doing this kind of search, unless you restrict it using the "site:" directive to domains that are more trustworthy.

The "inurl:" directive lets us search for specific terms to be included in the URL of a given site. This can be helpful in finding well-known vulnerable scripts on web servers, including CGI, ASP, JSP, PHP, and others. For example, searching for `inurl:viewtopic.php` finds sites with URLs that contain "viewtopic.php" in them. That script is commonly associated with the phpBB suite of tools for implementing web-based discussion forums. Historically, there have been numerous flaws in phpBB implementations, so locating those sites is helpful if a new vulnerability is discovered.



Often, we want to search Google for some specific kinds of files. Most of the web consists of html and htm pages, but there are numerous other kinds of files that interest us. As Google scours the Internet finding new websites and adding their pages to its search directory, it recognizes several hundred different file types, allowing us to search for those types of files. For example, we can look for .pdf files. Or to make things more interesting, we can search for .xls files, which are commonly associated with Excel spreadsheets. We might also look for .ppt files to find PowerPoint presentations. Some organizations inadvertently put sensitive .xls or .ppt files on their websites, for which we could search.

To perform such searches, we have two options. The first is to rely on Google's "filetype:" directive followed by the suffix of the file we want to find. Note that the "ext:" directive does the exact same thing as the "filetype:" directive; it is exactly the same search. For example, we can search for PowerPoint files in the counterhack.net domain by looking for `site:counterhack.net filetype:ppt` or `site:counterhack.net ext:ppt`. Our second option is to look for the suffix of the file as a general search term. Sometimes Google gets confused about a file type, messing up the appropriate association and omitting that given file from the filetype: results. Using the file suffix as a general search term without the filetype: or ext: directive will usually give us more results because not only will we get files that have that suffix in their name, but we will also get web pages that merely include the text associated with that suffix. For example, if we search for `site:counterhack.net ppt`, we will not only get PowerPoint files, we'll also get a series of web pages that include the text ppt.



Several years ago, Johnny Long created a list of useful Google searches to find vulnerable systems. He called each individual search a GoogleDork, and the entire inventory of all these searches is known as the Google Hacking Database (GHDB).

Today, the folks who run the Exploit-DB took over where Johnny left off and make a searchable list of the updated GHDB available online at the URL shown on this slide. There are more than 1,000 different searches in the GHDB that can find several varieties of security flaws and related issues, all by simply searching Google.
The GHDB is divided into numerous categories. All the individual categories are listed on the slide, but some of the most important and interesting to us follow:

- **Advisories and vulnerabilities**: These searches find vulnerable systems, usually by identifying a known flawed CGI script using the inurl directive or a page with known flaws identified with the intitle directive.
- **Files containing juicy info**: These searches find files that are often associated with caches and logging. Although they don't look for passwords directly, this cache and log information could be useful in learning more about the target organization.
- **Files containing passwords**: Numerous tools generate files that contain either clear text passwords or encrypted/hashed passwords. These searches identify when such files are available via a web server.
- **Footholds**: These searches locate sites where an attacker may get a foothold that can later be used to compromise the server. A lot of these searches find admin login pages for various common web-based software environments. The Login portals category is similar.
- **Network or vulnerability data**: These searches find pages that hold logs and/or configuration information about network devices, such as firewalls, VPNs, routers, Intrusion Detection Systems, and so on.
- **Sensitive online shopping info**: These searches find websites that may reveal sensitive online shopping info, including customer orders, weak shopping cart implementations, and overly detailed product information.
- **Online devices**: This category of searches helps locate web-based video cameras, printers, and various kinds of appliances.
- **Vulnerable servers**: These searches locate web servers that may have a vulnerability, a category similar to Advisories and vulnerabilities.

exploit - db. com.

Some Interesting Samples from the GHDB

- PGP and GnuPG private key rings:
  `intitle:index.of intext:"secring.skr"|"secring.pgp"|"secring.bak"`
- Shell history files in interesting domains:
  `site:somethinginteresting intitle:index.of bash_history`
- Robots.txt file with excessive disallow lines:
  `robots.txt disallow filetype:txt`
- Nessus scan results:
  `intitle:"Nessus Scan Report" "This file was generated by Nessus"`

Network Pen Testing and Ethical Hacking 165

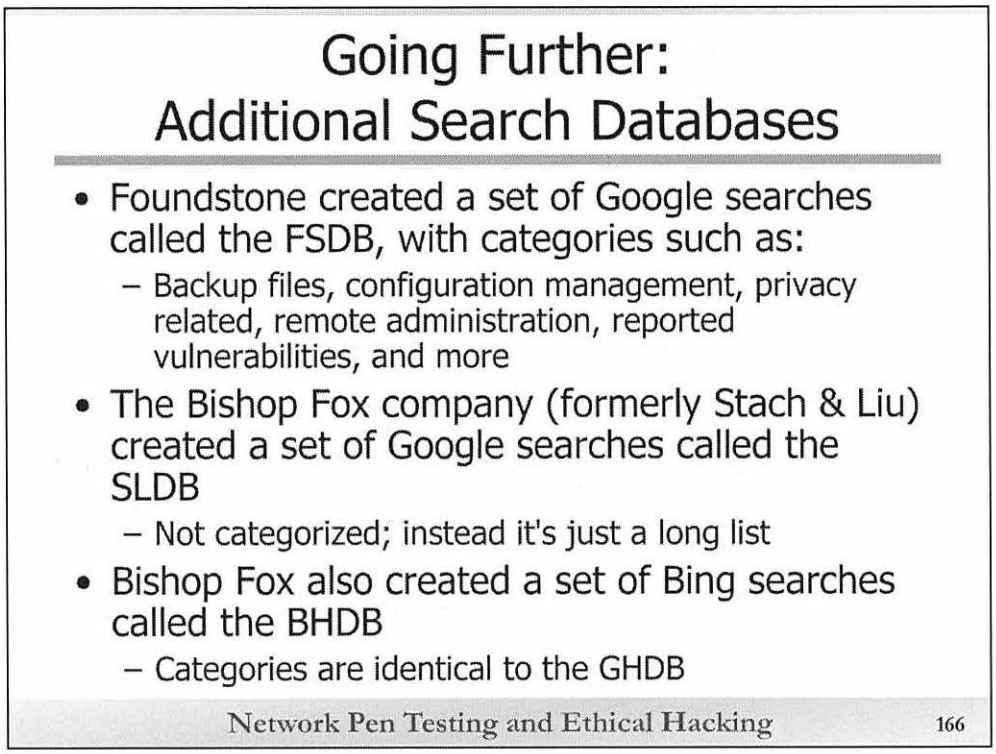
Although the GHDB includes many hundreds of eye-opening Google searches to find vulnerable sites or sensitive data, let's explore a handful of them to get a feel of the power of the GHDB and Google.

The first search on this slide will look for websites with directory indexing, in which we have files named secring.skr, secring.pgp, or secring.bak. These are the common filenames where private keys are stored for the Pretty Good Privacy (PGP) and Gnu Privacy Guard (GnuPG) tools. Although the private keys are encrypted with the user's passphrase, having them available on a website and searchable via Google is a bad idea. An attacker could grab the files and launch a password guessing attack against them.

The next search on the slide finds bash shell history files on websites, showing the commands some user typed into the shell. By putting in an interesting domain with the `site:` directive, we may find history files with sensitive information in them.

For the next search, we look for robots.txt files. A website administrator can use a robots.txt file to tell well-behaved search engine crawlers to ignore certain directories or pages on the website, with the Disallow syntax in this file. That way, those pages or directories won't be searchable or cached by the search engines. But, robots.txt is a double-edged sword, indicating where sensitive items may be stored. With the Google search on this slide, we can find websites that have a large number of disallow lines, indicating that a large number of individual directories or pages on the site have been disallowed. They might have something interesting to hide.

We can even ask Google to find sites that have output files from the Nessus vulnerability scanner. Somehow, this file was placed in a directory under the document root of a web server, so that Google could find it and make it searchable. A list of vulnerabilities of a target site could be useful for an attacker. We can save time running a vulnerability scanner by analyzing the results of earlier scans already done by the target organization.



Based on the pioneering work of the GHDB, other organizations began developing series of search-engine queries to find vulnerable systems and sensitive information disclosure. In particular, Foundstone developed its Foundstone Database (FSDB), which includes categories such as backup files (created by various file system backup tools), configuration management (allowing an attacker to view or alter the configuration of a system via a web-based interface), privacy-related (including sensitive personal information), remote administration, reported vulnerabilities (misconfigured and unpatched systems), and more.

The folks at the Bishop Fox security consulting company (formerly called Stach & Liu) created a set of Google searches called the SLDB. It's not broken into individual categories but is instead a long list of entries not found in the GHDB.

Bishop Fox personnel went further and created a set of searches to find vulnerable systems and information disclosure via Microsoft's Bing search engine. Its results is known as the Bing Hacking Database (BHDB), which includes the same categories of vulnerabilities found in the GHDB.

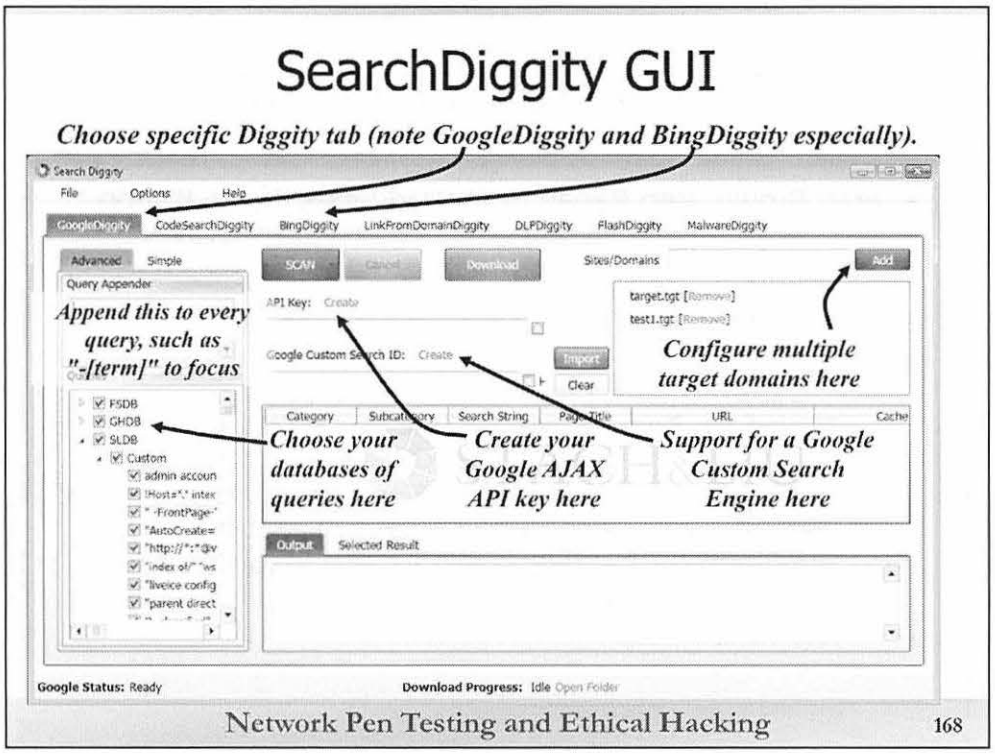
One of the most powerful search engine vulnerability finding tools is the SearchDiggity suite, created by Fran Brown, Rob Ragan, and Brad Sickles of the Bishop Fox Company, and distributed for free.

SearchDiggity is a Windows GUI that ties together several other command-line tools created by Bishop Fox. Each of these individual tools is represented as a tab in the GUI of the overall SearchDiggity tool. The breadth of this suite is astounding and includes the flagship search tool GoogleDiggity. This component supports searches from the FSDB, GHDB, and SLDB for one or more domains specified by the user. Queries are submitted using Google's AJAX API, which is not a violation of the Google terms of service, unlike other tools that scrape Google's normal search page.

SearchDiggity also provides a GUI tab for BingDiggity, Bishop Fox's Bing search tool that uses the BHDB. This tool submits queries using the Bing API, requiring a free Bing API key from Microsoft.

The DLPDiggity tab within SearchDiggity focuses on Data Leakage Prevention. It has the capability to search both Google and Bing for evidence of websites that are leaking sensitive personally identifiable information, such as Social Security numbers, bank account numbers, and other related data.

SearchDiggity also includes the MalwareDiggity feature, which performs both Google and Bing searches to see if there is any evidence of a website in a user-selected domain attempting to exploit browsers that access the website. By conducting searches using the site: directive, MalwareDiggity looks for different types of drive-by downloads and browser exploits attackers frequently deploy on websites that they've compromised to attack browsers that surf to those sites.



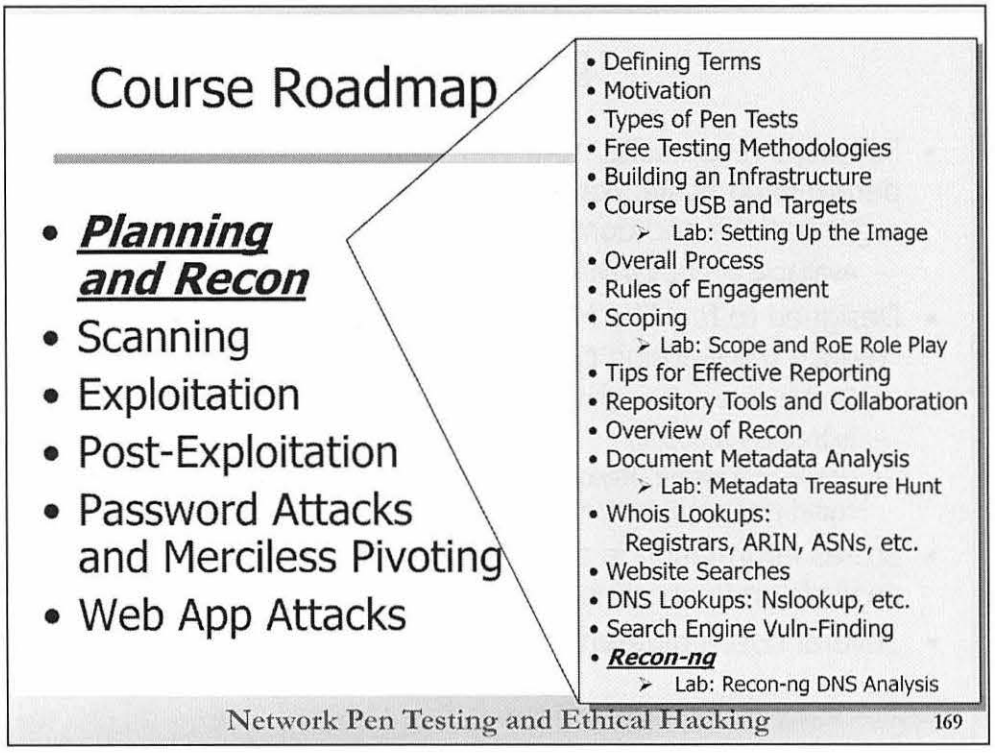
On this slide, you can see the SearchDiggity GUI. Along the top of the GUI are various tabs to access the individual search components of SearchDiggity, including GoogleDiggity, BingDiggity, and more.

Within each tab, the configuration is similar. The GoogleDiggity tab includes a selection box in the lower left to select the FSDB, GHDB, and/or SLDB, or configure individual categories or even individual searches within those groups by simply checking a box. We can also specify an individual string to append to every query, allowing us to customize the tool even further. Sometimes a penetration tester may place a -[term] here to eliminate all results that include the given term, keeping answers more focused.

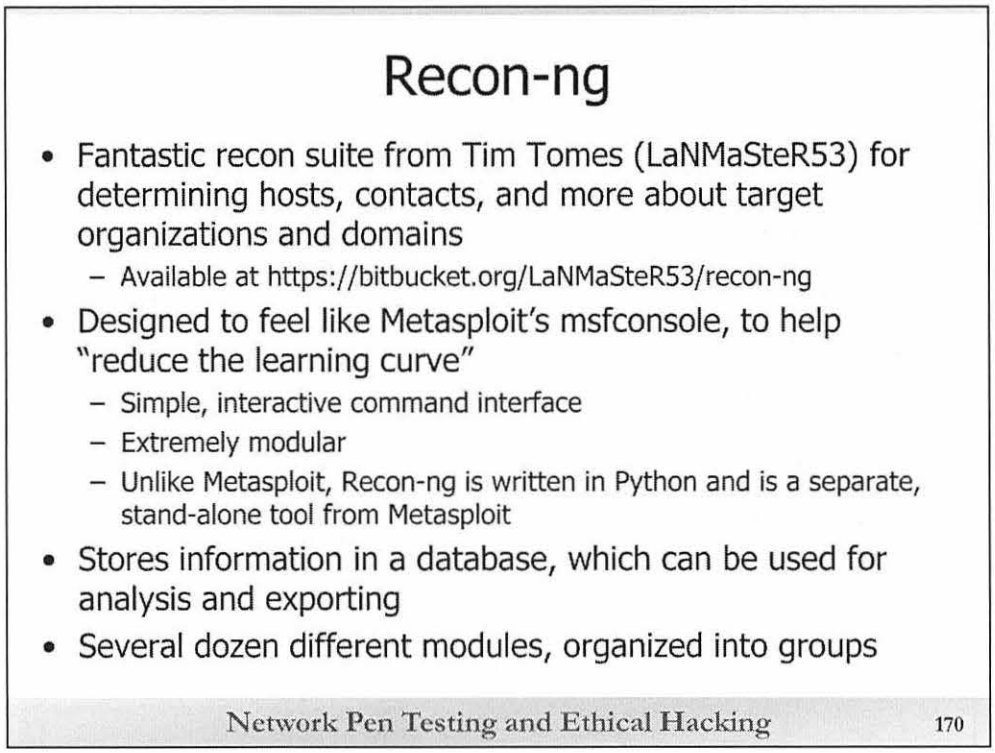
Near the middle of the screen, we have a field where we can enter a Google AJAX API key. If you don't have one yet, you can click the Create link to go to Google to get one. We can also specify a Google Custom Search Engine (which Google lets organizations create to have a small slice of Google specifically tailored to their organization searchable on the Internet). And, finally, we can search for an arbitrary number of domains by simply adding to the list on the upper right. GoogleDiggity will prepend a site:[domain] directive to each search for each domain name we enter here.

The configuration for BingDiggity is virtually identical but allows us to select items from the BHDB and create a Bing API key from Microsoft.

Consider how flexible this tool is with its GoogleDiggity and BingDiggity features. We can choose a huge number of queries and run them against an arbitrary number of domains, harvesting search data in a manner that doesn't violate the terms of service of either Google or Bing.



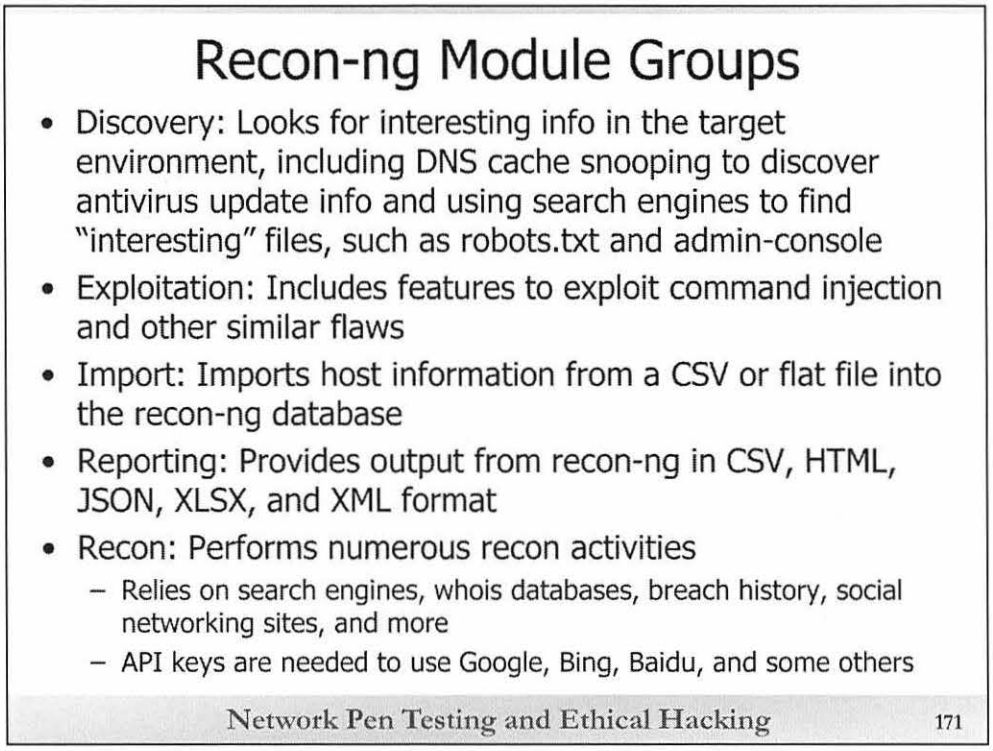
Now that we've looked into various aspects of reconnaissance, let's look at a tool that brings many of these ideas and techniques together: Recon-ng. We'll also do a lab focused on the tool, where you query DNS servers for useful information in a penetration test.



Recon-ng is a framework that pulls together numerous different reconnaissance capabilities, all within a single interactive command interface. Written by Tim Tomes, Recon-ng was designed to feel like the msfconsole interface for the Metasploit framework to help ease the learning curve for this tool.

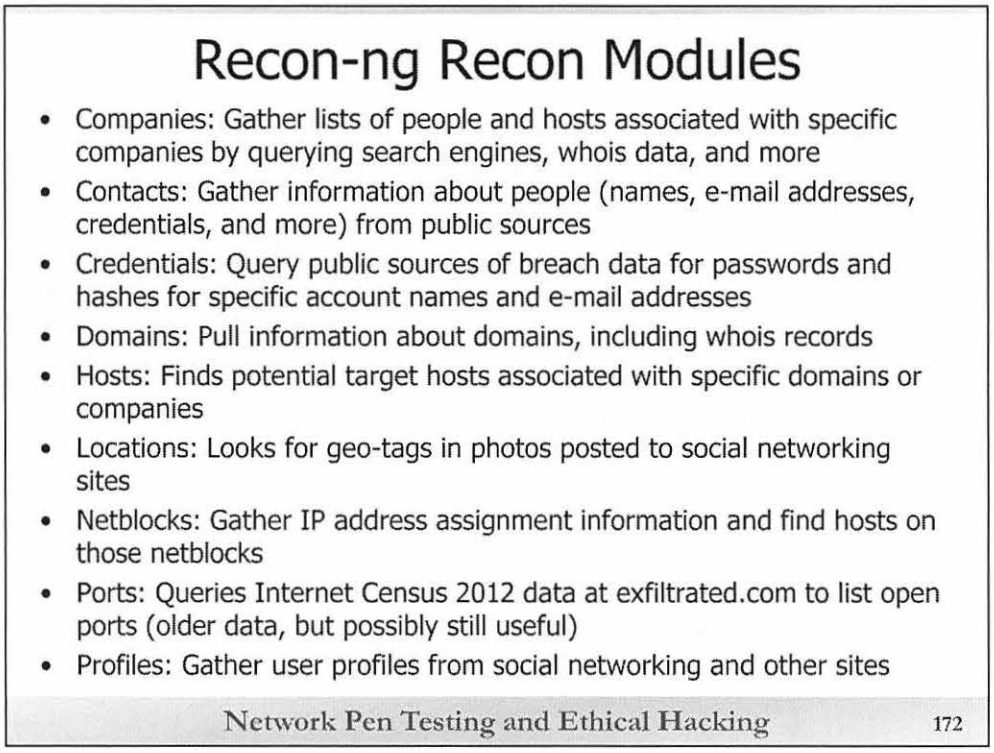
As it gathers information about hosts and contacts associated with a target organization or domain, it stores the results in a database. A penetration tester can query this database or export it in a variety of forms.

Built of several dozen modules, a penetration tester can choose which module to use, configure variables for the module (such as target domain name and/or API key for accessing a search engine), and then run the module to pull back recon data. Recon-ng is written in Python and is designed so that other developers could create modules to extend its functionality.



The modules in Recon-ng are divided into groups, with each group focused on specific tasks a penetration tester might perform. The following module groups are included in Recon-ng:

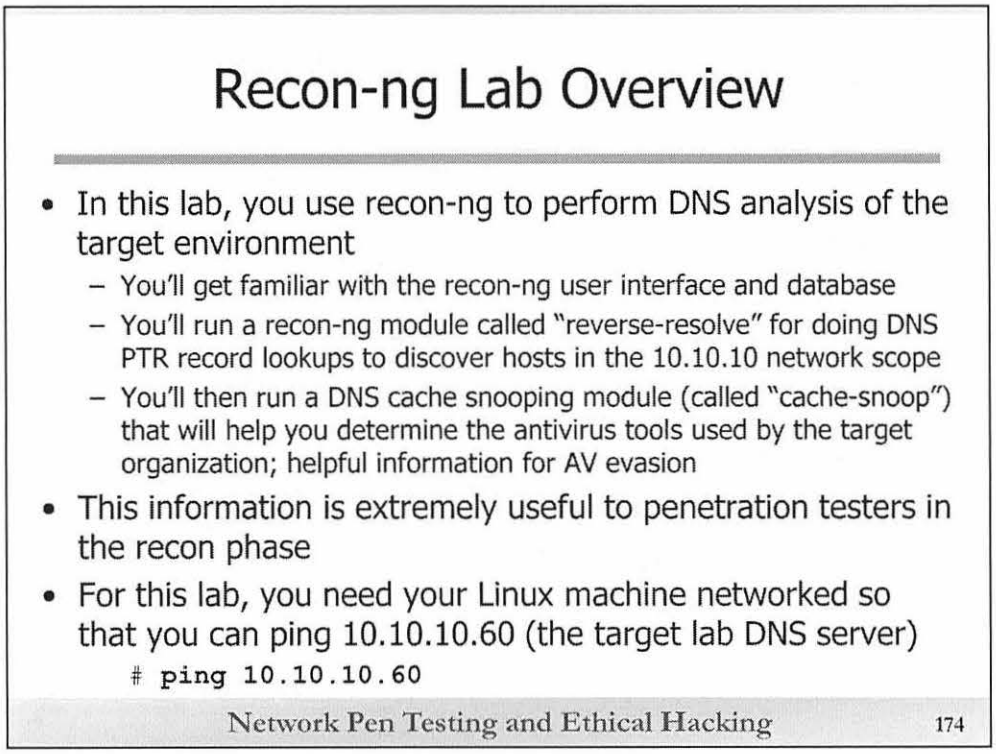
- **Discover:** The modules in this group help a penetration tester learn more about a target environment. In particular, this group includes a DNS Cache Snooping module to determine which antivirus vendor's DNS records are cached by the target organization (a good sign that the target organization uses the given AV product), as well as a module that performs web searches for files that reveal admin interfaces and robots.txt files in target domains.
- **Exploitation:** This group includes modules that actually try to exploit a target machine. Currently, its focus is on exploiting web application command injection flaws, but in the future, additional exploit modules may be placed here.
- **Import:** These modules allow penetration testers to pull data about hosts, domains, and so on into the Recon-ng framework. Currently, CSV files and flat files that have one host per line are supported.
- **Reporting:** These modules allow a penetration tester to export Recon-ng findings in a variety of formats.
- **Recon:** These reconnaissance modules are the heart of the Recon-ng framework, pulling information from many different places back into Recon-ng, including search engines, whois databases, breach history (where account names, e-mail addresses, passwords, or hashes are published), social networking sites, and more. For some of the search engines (including Google, Bing, Baidu, and others), a penetration tester will need to get a free API key from the search engine company and import it into Recon-ng to be able to query those sites. Let's zoom into the different types of recon modules in this category on the next slide.



Recon-ng supports several different types of modules in the recon group, each pulling back or manipulating recon data in specific ways. More than 60 recon modules are available, pulling specific kinds of data from a variety of sources. These recon modules are sorted according to the type of data they handle. The types of data (and therefore the type of recon modules) include

- **Companies:** These modules pull data associated with specific target organizations, querying search engines, whois data, and more.
- **Contacts:** Modules in this group focus on individual people, attempting to gather names, e-mail addresses, and credentials from public sources.
- **Credentials:** These modules query published information associated with large-scale breaches to determine if given contacts (names or e-mail addresses) have had their password or hashes publicly disclosed in the breach.
- **Domains:** These modules gather information about specific domain names, particularly focusing on whois records.
- **Hosts:** These modules search for individual hosts in the target organization.
- **Locations:** These modules look for information associated with specific geocoordinates, including photos posted to social networking sites with geo-tags in them.
- **Netblocks:** These modules look for groups of IP addresses organized as Netblocks, including queries to Regional Internet Registrars (RIRs).
- **Ports:** The modules in this category search a public database at exfiltrated.com to pull information about which ports were discovered listening on specific IP addresses during a 2012 port scan of the entire Internet. Although that's older data, it is still possibly useful to penetration testers today.
- **Profiles:** Modules in this group pull detailed information about target organizations and users based on social networking profiles and general web searches.

To become more familiar with Recon-ng, let's perform a lab using the tool. In this lab, we use Recon-ng to gather useful information from a target's DNS infrastructure, relying on a couple of particularly useful Recon-ng modules: one in the Recon group, the other in the Discovery group.



In this lab, you gain familiarity with the user interface of Recon-ng and use it to gather useful information from the target organization's DNS infrastructure.

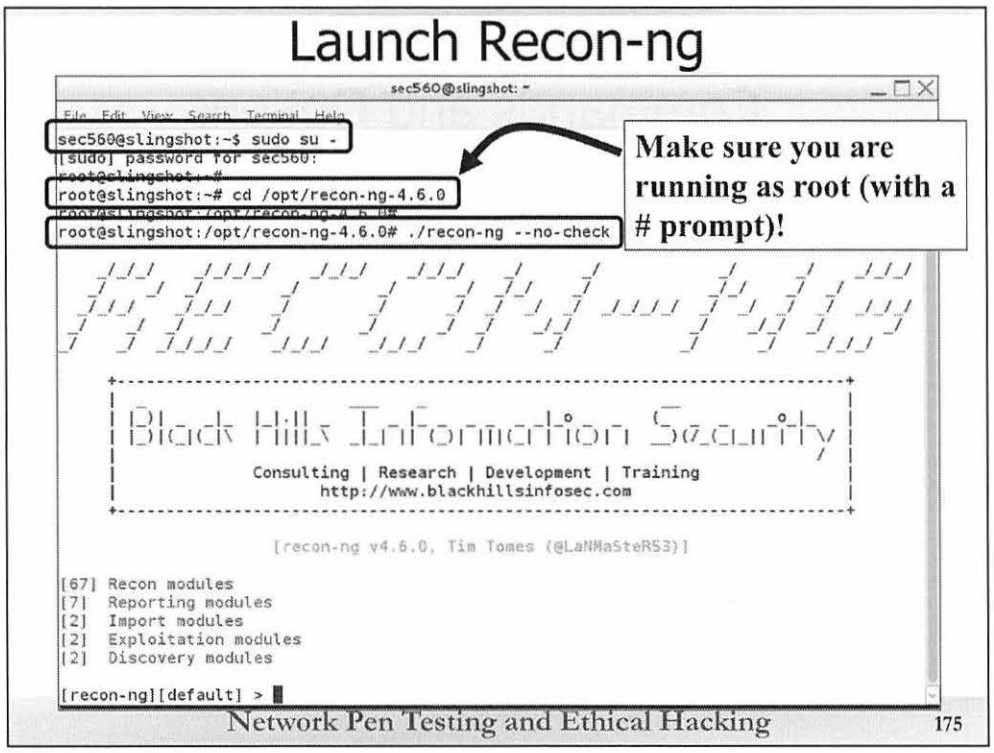
In particular, you run a Recon-ng module called reverse-resolve, which takes a netblock of IP addresses and sends PTR (reverse record) lookups to a DNS server to determine which of those IP addresses resolve into names. That's a useful feature for a penetration tester because it can help you identify hosts that could be included in your scope, provided that these hosts have PTR records in DNS. Many organizations provide PTR records for important hosts on the Internet, so this technique can be helpful during the reconnaissance phase.

Next, you'll use a Recon-ng module called "cache-snoop" that performs DNS cache snooping against a target DNS server. This module looks for cached DNS records associated with a couple dozen antivirus firms' signature update site DNS records. If the target organization relies on any of those AV products, they likely will perform regular updates of their AV signatures, which will leave residual records associated with the AV company cached in the organization's DNS servers. By identifying those cached entries, a penetration tester can determine which AV products the target organization is using, a helpful piece of information useful in evading the organization's AV product.

For this lab, you need to be connected to the 560 target network environment. Make sure you can ping 10.10.10.60 (where there is a DNS server) before you begin the lab:

```
# ping 10.10.10.60
```

If your ping doesn't work, that means your Linux guest machine isn't properly connected to the 560 target network. Consult the network configuration section earlier in this book, or reach out to an instructor or TA for assistance.



Start the lab by making sure you are running with root privileges (with a # prompt). You can get root privileges by running:

```
$ sudo su -
```

Then, type in your sec560 account password. You should now have a # prompt instead of a $ prompt, indicating you have root privileges.

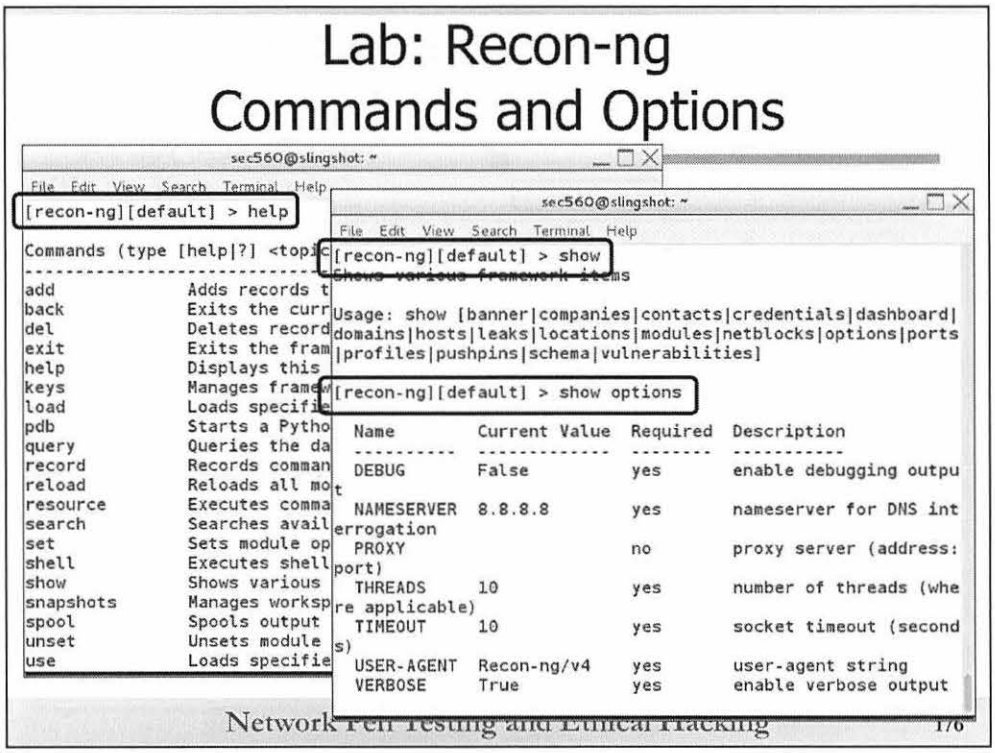
Next, navigate to recon-ng's directory:

```
# cd /opt/recon-ng-4.6.0
```

Then, launch recon-ng with the --no-check option so that it doesn't try to download the latest version of the tool. In penetration testing, it is typically preferable to use a vetted version of a tool that you know works well instead of a brand-new version that hasn't received as much testing and scrutiny:

```
# ./recon-ng --no-check
```

On the screen, you should see ASCII art announcing RECON-NG. You'll also see an inventory of the types of modules, including Recon, Reporting, Import, Exploitation, and Discovery.



To become familiar with Recon-ng's user interface, let's explore its help feature:

```
[recon-ng] > help
```

Here, you can see all the commands supported by Recon-ng. One of the most important commands is the show command because it lets you look at Recon-ng's options, configuration, and variable settings. Let's run show by itself to see the various items we can explore using show:

```
[recon-ng] > show
```

Here, we can see that we can "show banner" to get version information. We can likewise run show followed by an item type in Recon-ng's database, such as show hosts or show domains. We'll do that later in the lab.

To see the variables set in Recon-ng, run:

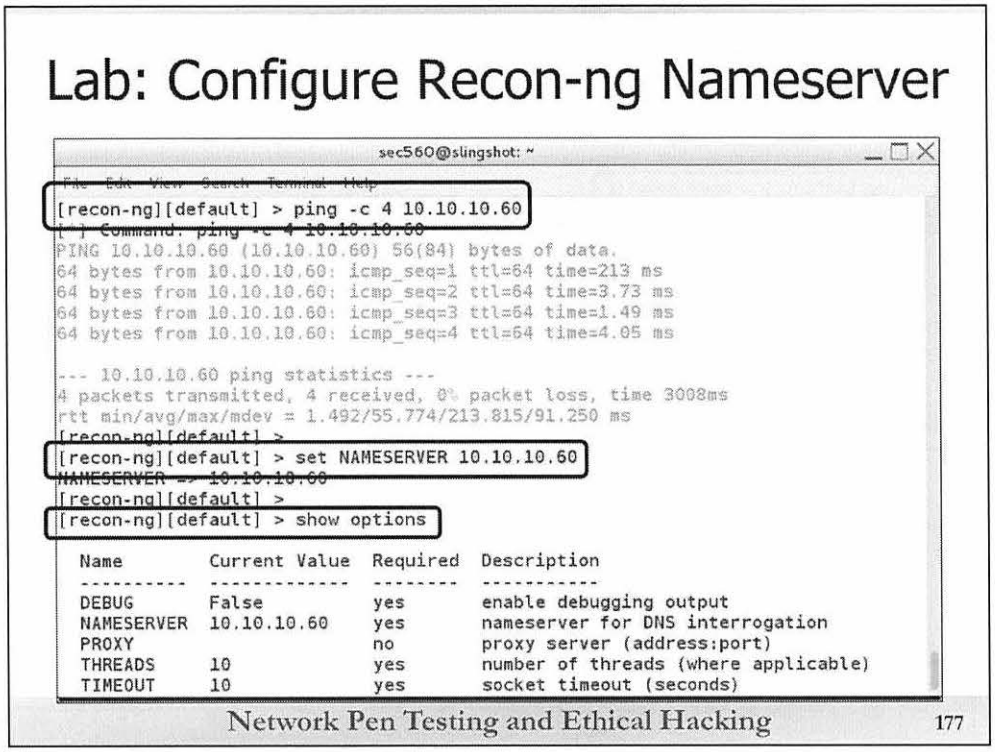
```
[recon-ng] > show options
```

Here, you can see that by default, Recon-ng resolves information using the 8.8.8.8 name server provided by Google. We'll change that shortly to our target organization's DNS server.

Before we do that, though, let's take a quick look at Recon-ng's database structure, so you can see the tables and their columns where Recon-ng will store data:

```
[recon-ng] > show schema
```

Note that there is a domains table, a hosts table, and several other tables that are automatically populated as we run various Recon-ng modules.



The Recon-ng prompt handles a variety of Recon-ng commands. But when it receives a command it doesn't recognize, Recon-ng passes that command to the underlying operating system shell for execution. This is handy because it means we can run general-purpose commands at the Recon-ng prompt. Let's try it by running a ping command to ping 10.10.10.60 four times (-c 4 for a count of four):

```
[recon-ng] > ping -c 4 10.10.10.60
```

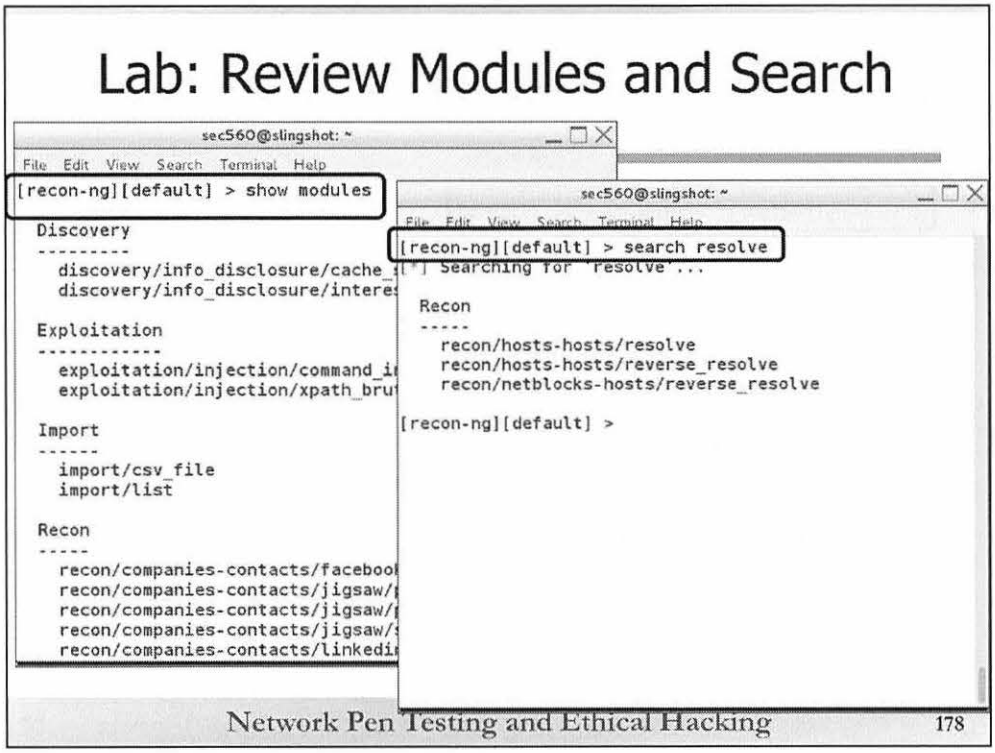
Your ping should work and you should see its output. It's important to note that Recon-ng does NOT have a ping command. Instead, Recon-ng is simply taking our ping command and handing it to the underlying shell for execution.

To start performing recon against the target organization's DNS server, let's configure Recon-ng to use that name server, as follows:

```
[recon-ng] > set NAMESERVER 10.10.10.60
```

Now, when we run show options, we can see that the original 8.8.8.8 name server has been altered to 10.10.10.60.

```
[recon-ng] > show options
```



Let's now explore the various modules Recon-ng has:

```
[recon-ng] > show modules
```

Here, you'll see different groups of modules, including Discovery, Exploitation, Import, Recon, and more. Under each module group, you can see the individula modules, totallying several dozen.

Often, a penetration tester has a sense of the type of module he would like to use, but doesn't know the full module name or its path to access it in Recon-ng. If that's the case, we can use the search command to find specific modules based on strings in the module's name or path.

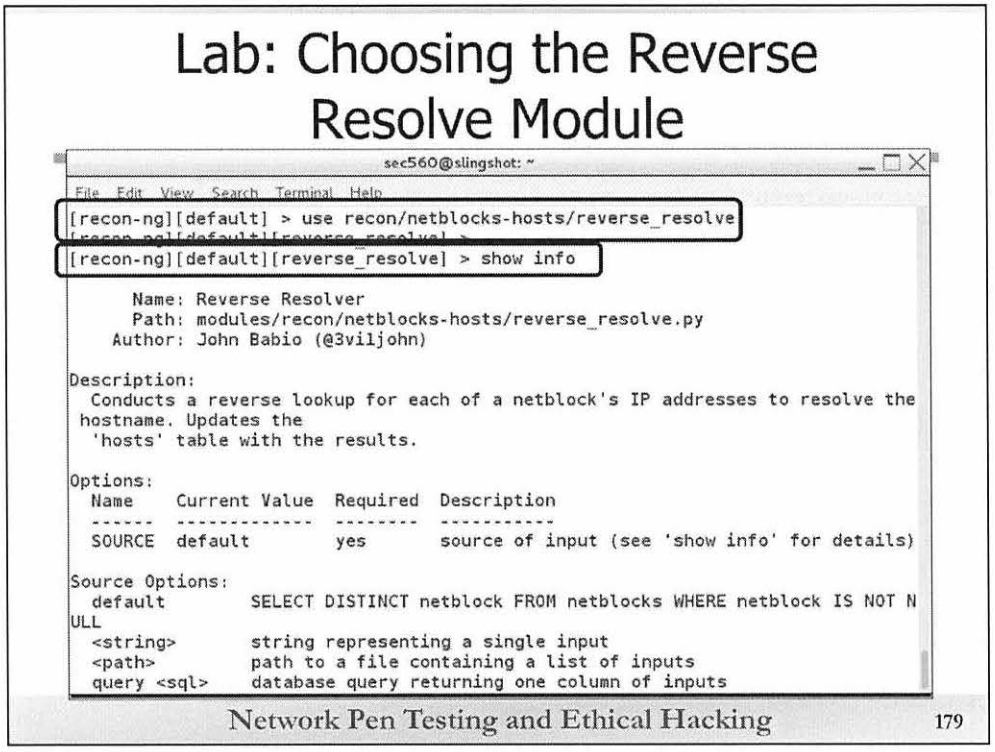
Suppose, for example, we wanted to find modules that would resolve names (via either a forward or reverse DNS lookup). We could simpy run search resolve. Do that now:

```
[recon-ng] > search resolve
```

Here, we can see several modules associated with resolving names. Notice that their paths all start with recon, as they are in the recon module group.

For this lab, we'd like to iterate through a given target Netblock (10.10.10) to see which host IP addresses have an associated PTR record. This is a useful way to find hosts and explore our scope in a penetration test. Of course, not every host on the Internet has a PTR record, but many DMZ systems do, and we can use this module to help identify them.

To achieve this, we'll use the recon/**netblocks**-hosts/reverse_resolve module. There is also a recon/**hosts**-hosts/reverse_resolve, which takes as its input individual IP addresses. We'll use the netblocks module, though, as we've been given the full 10.10.10 network as our target scope.



Let's select that recon/netblocks-hosts/reverse_resolve module with the use command, followed by the full path to the module:

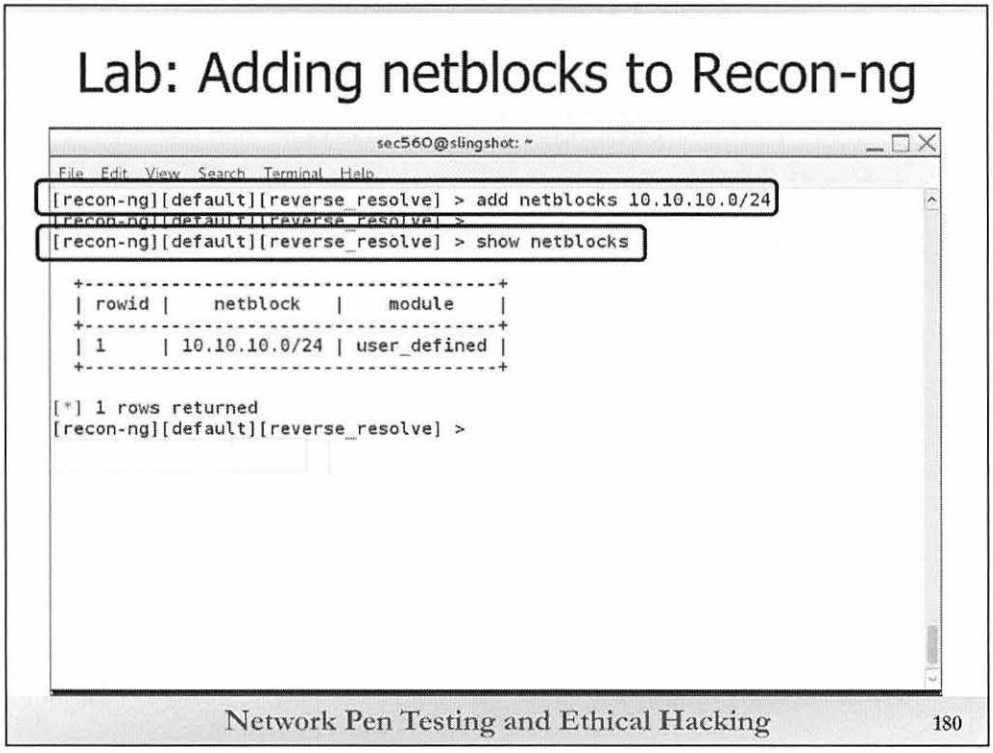
```
[recon-ng] > use recon/netblocks-hosts/reverse_resolve
```

Now, to get the details of that module we can run show info:

```
[recon-ng] > show info
```

Here we see a brief description of the module, plus the different variables it supports. For this module, the SOURCE variable specifies where the information about our target netblock comes from. By default, Recon-ng simply looks in the netblocks table. We can specify other places, including a string that contains a single netblock or a path to a file that contains a list of netblocks, one per line.

For this lab, we'll leave it as its default, and place a netblock in the netblocks table next.



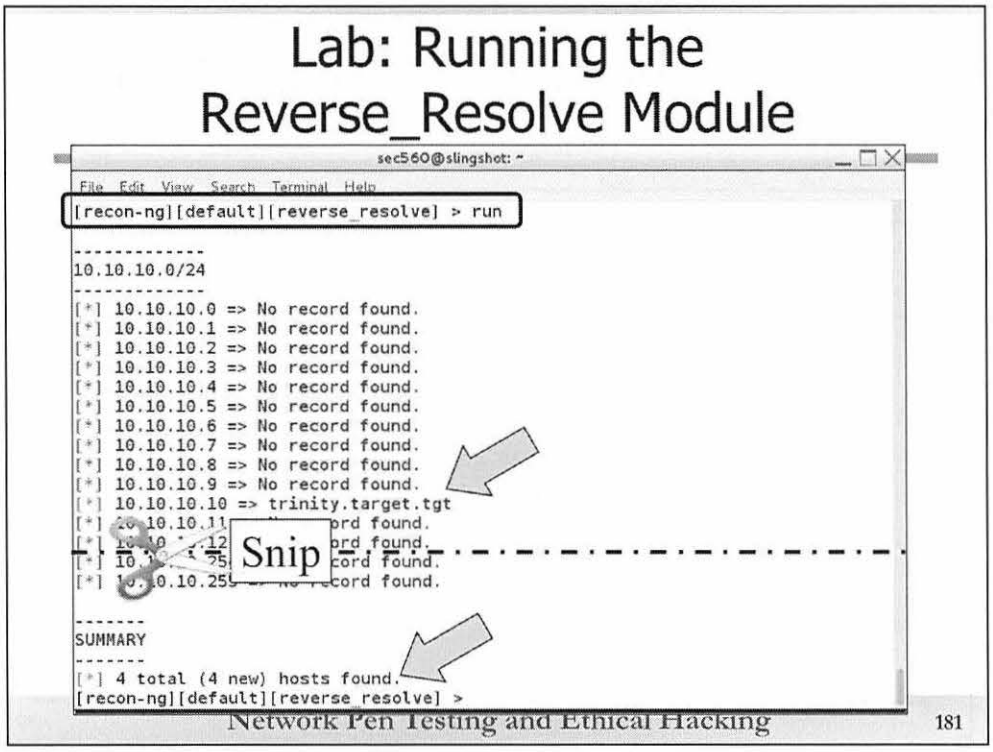
Let's add a netblock to the netblocks table using the add command:

```
[recon-ng] > add netblocks 10.10.10.0/24
```

We can now look at the netblocks table to see our information there:

```
[recon-ng] > show netblocks
```

Because we left the SOURCE for the reverse_resolve module to default, Recon-ng will pull this information from the database to do the PTR lookups.

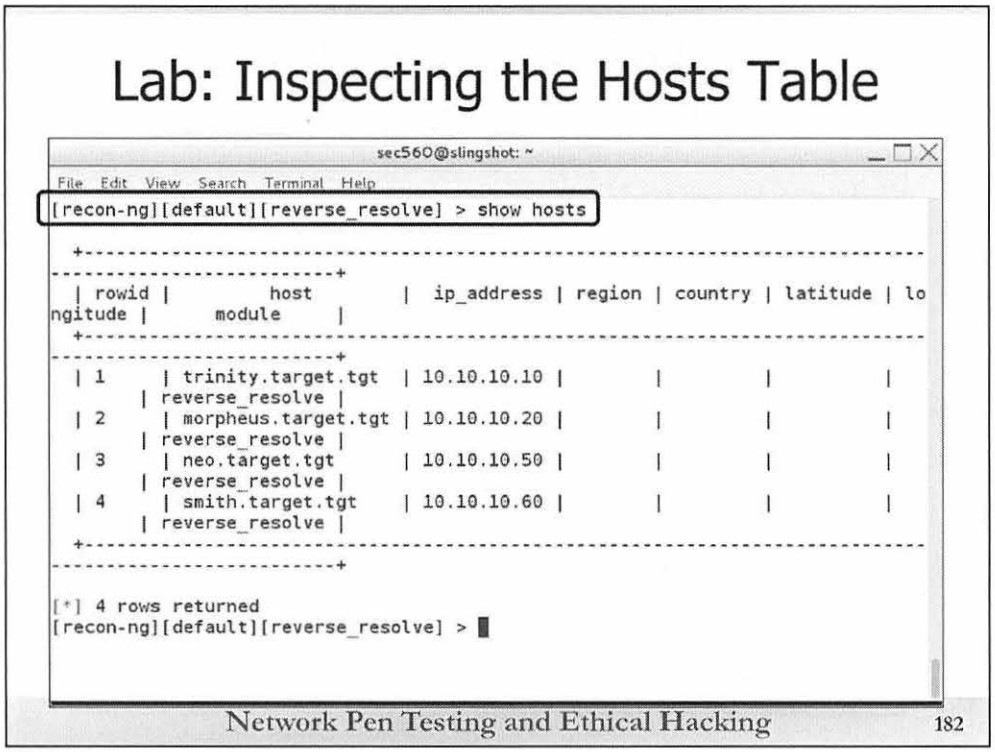


With our module configured, we can now run it as follows:

```
[recon-ng] > run
```

In the output, we can see it sending a PTR query for each IP address in 10.10.10, looking for a response. For most of the IP addresses, no record will be found. But for 10.10.10.10, 10.10.10.20, 10.10.10.50, and 10.10.10.60, it should get a PTR record response, displaying that information on the screen.

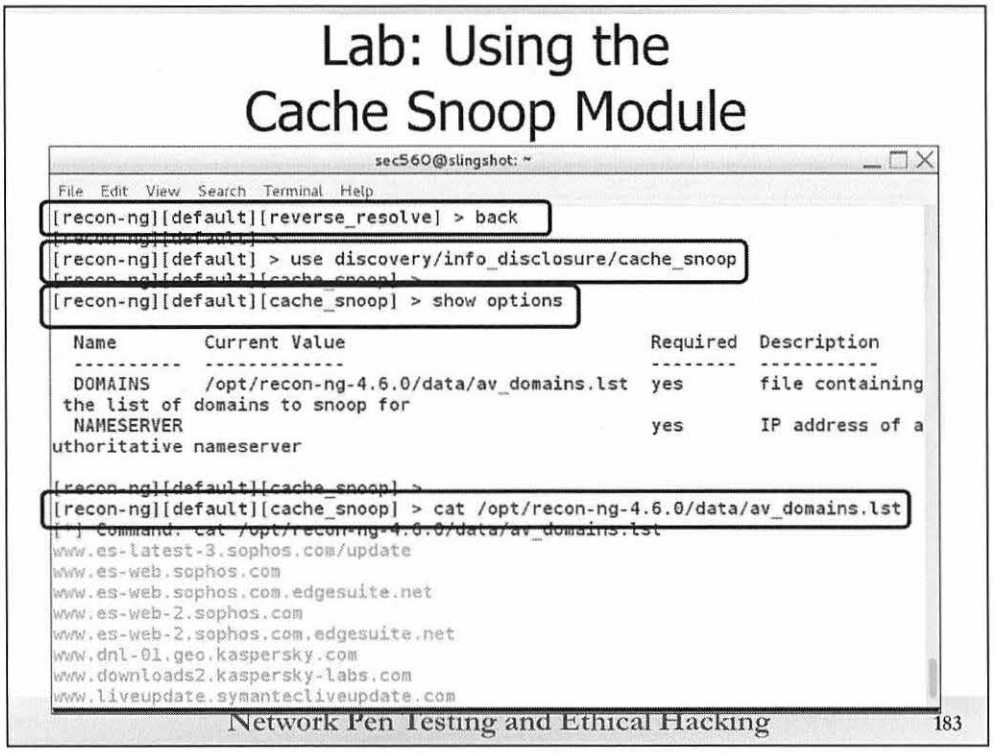
When the module is finished running, it will show us how many hosts it found. It should find four.



In addition to scrolling back on the screen to see what Recon-ng found, we can also look at the hosts table because the reverse_resolve module automatically populates it. Let's look at our newly discovered hosts:

```
[recon-ng] > show hosts
```

Note that we have a domain name and IP address for each of the hosts based on the returned PTR record. The hostnames are all associated with the target.tgt domain (which we can check against our target scope) and include names such as trinity, morpheus, neo, and smith, a naming scheme based on a movie.



Now that we've gathered some hosts associated with the target environment, let's use another Recon-ng module to determine the most likely antivirus tool or tools the target organization is using. We can do that with the cache_snoop module in Recon-ng's discovery group.

We can back out of our current module to the general Recon-ng prompt using the back command:

```
[recon-ng] > back
```

We'll now use the discovery/info_disclosure/cache_snoop module:

```
[recon-ng] > use discovery/info_disclosure/cache_snoop
```

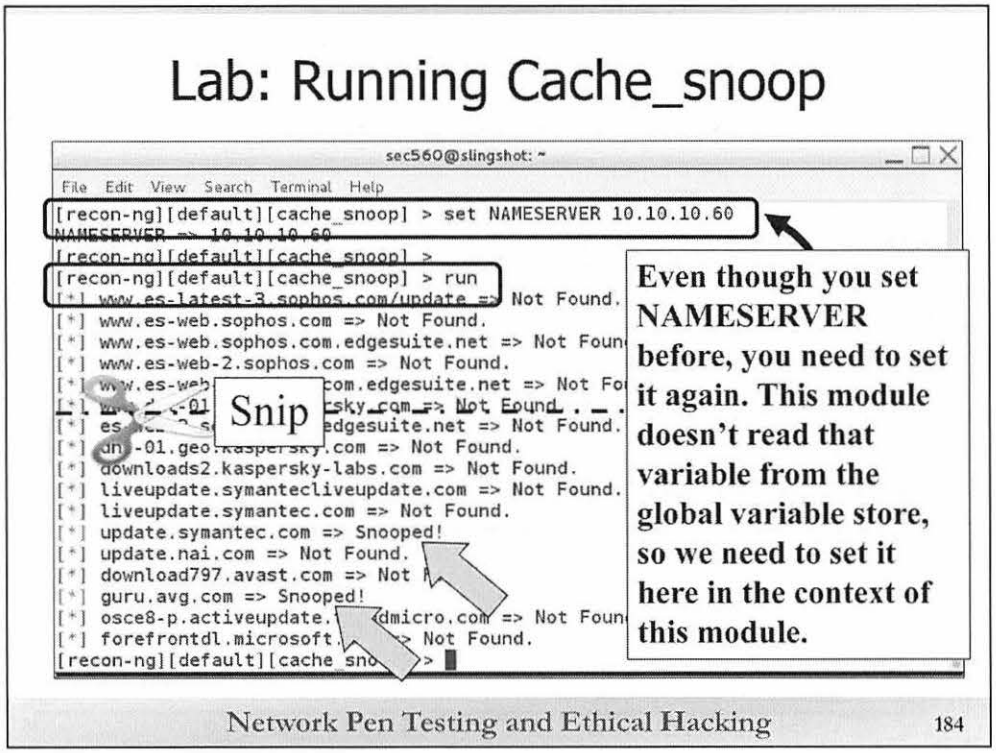
Let's look at the options for this module:

```
[recon-ng] > show options
```

Here, we see that this module needs a NAMESERVER. (Unfortunately, the module doesn't automatically populate the NAMESERVER value with the one configured overall for Recon-ng.) This module also needs a list of names that it should look for in the target DNS server's cache. By default, it searches for names stored in the av_domains.lst file that comes with Recon-ng. Let's look at the contents of that file:

```
[recon-ng] > cat /opt/recon-ng-4.6.0/data/av_domains.lst
```

Here, we can see the names of update servers for numerous different AV product firms. You could expand this list or even create your own in future penetration tests, based on different items you'd like to snoop for in a target organization's DNS cache. For this lab, we'll keep this default list, which is quite good.



We now set the NAMESERVER. (Remember that this module does NOT use the nameserver configured overall for Recon-ng so we must set it now in the context of the cache_snoop module.)

```
[recon-ng] > set NAMESERVER 10.10.10.60
```

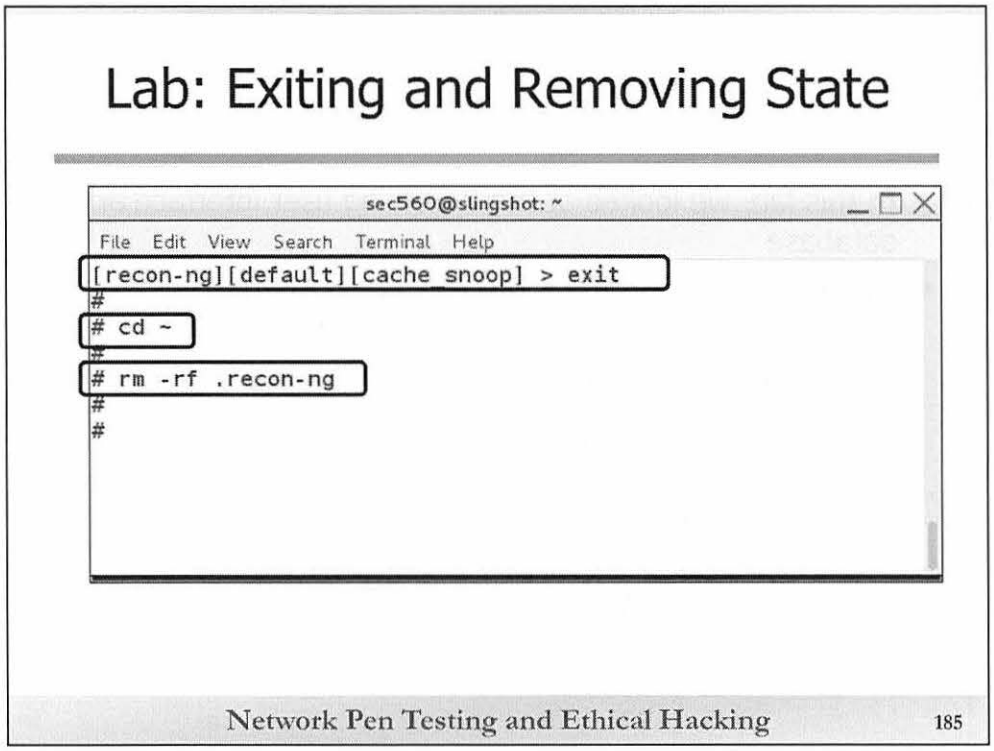
With all our settings now in place, we can run the module:

```
[recon-ng] > run
```

As the module runs, look carefully at its output. You'll note that it says Not Found. for the majority of the domain names. But, for two of them (update.symantec.com and guru.avg.com) it does show that it Snooped! a name. (That is, it found the name in the target DNS server cache.)

Thus, it is likely that the target organization is utilizing Symantec and/or AVG as its antivirus product, given that the target's DNS server was used to resolve those names recently (and the DNS Time-To-Live for those records has not expired so it remains in the cache). Of course, onfterce that TTL expires, the cached entries will be dropped.

This information about the target's AV vendor is tremendously useful in our penetration test, especially if we are going to create any malware for the target organization to send via spear phishing or other means. In **560.3**, we'll look at creating targeted malware that evades detection and perform a hands-on lab using the Veil-Evasion tool to do just that.



To finish the lab, we can exit the Recon-ng tool:

```
[recon-ng] > exit
```

We should also clean up the Recon-ng configuration file and database, which are automatically created in our home directory (~):

```
# cd ~
```

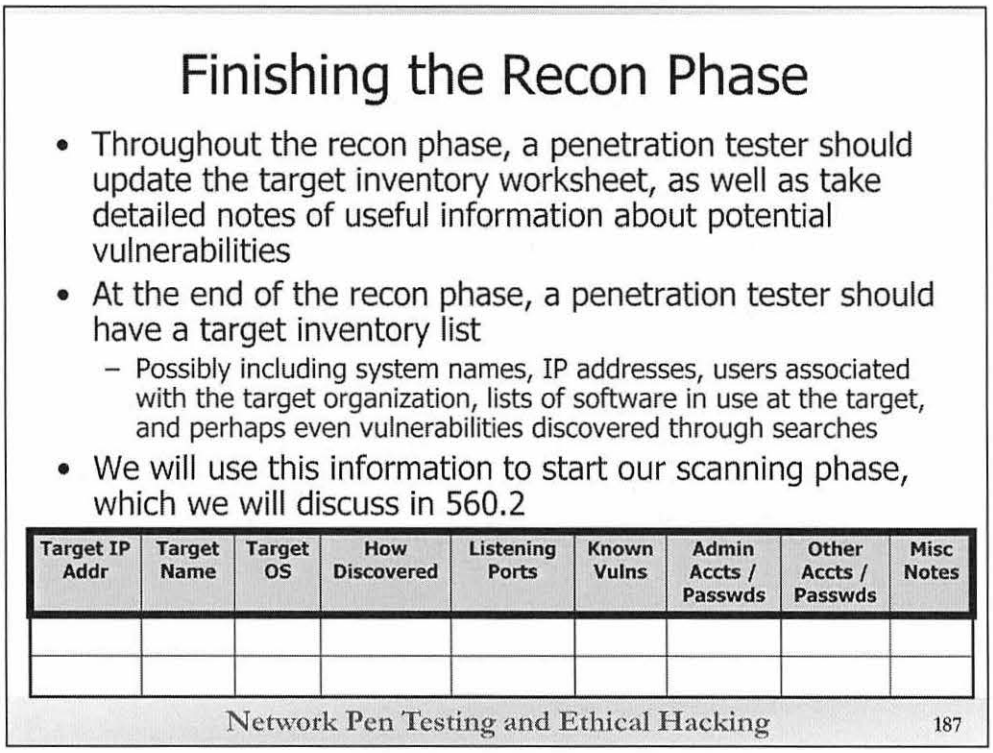
```
# rm -rf .recon-ng
```

This will remove all the information in the database as well as the custom name server configuration we set for Recon-ng.

Recon-ng Lab Conclusion

- In this lab, we looked at the Recon-ng user interface and database
- But, more important, we used it to perform useful reconnaissance about the target organization
  - We iterated through a netblock given in our scope (10.10.10.10) to identify specific hosts based on PTR records: 10.10.10.10, 10.10.10.20, 10.10.10.50, and 10.10.10.60
  - We determined the likely antivirus products used by the target organization: Symantec and AVG
  - Each of these items will be useful as we move forward with the scanning exploitation phases of our penetration test

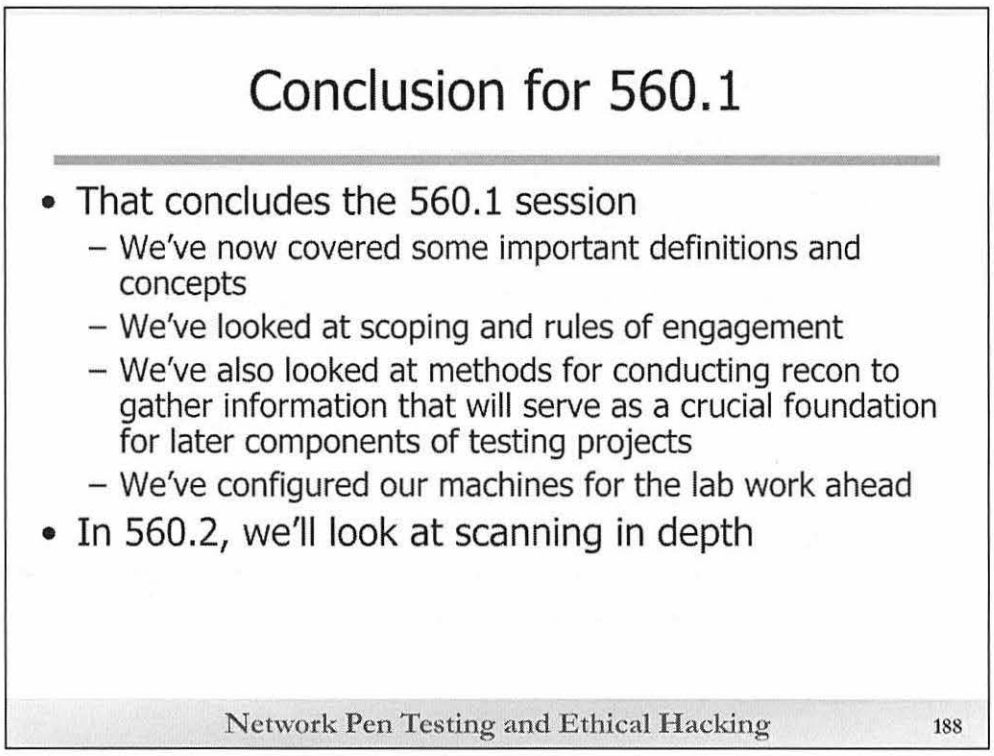
In this lab, we have used Recon-ng to get familiar with its user interface and look at its database. More important, though, we ran Recon-ng to pull some highly useful information about the target organization. Particularly, we iterated through a target netblock given to us in our scope to identify individual target host IP addresses. This information will be useful as we move onto the scanning phase of our penetration test, to be covered in **560.2**. And, perhaps even more important, we determine the likely antivirus products in use by the target organization. That information will be extremely useful as we move into the exploitation phase of the penetration test, in **560.3**.



Throughout the reconnaissance phase, a penetration tester should continue to populate the target inventory worksheet for each newly discovered system. In addition, the tester should make detailed notes of other information assets and potential vulnerabilities identified during this phase.

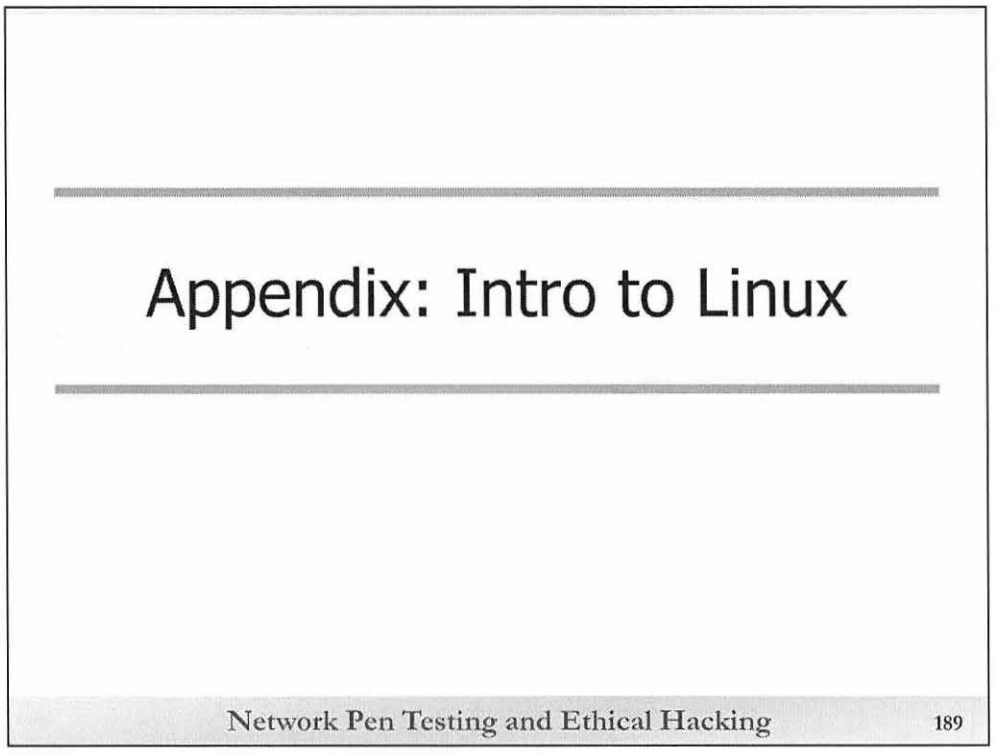
At the end of the recon phase, a tester should review the target inventory list, adding any final information that was discovered and highlighting specific areas in the Misc Notes section that require further analysis and exploration.

This target inventory list will be a crucial input into our next phase, Scanning, which is described in 560.2.

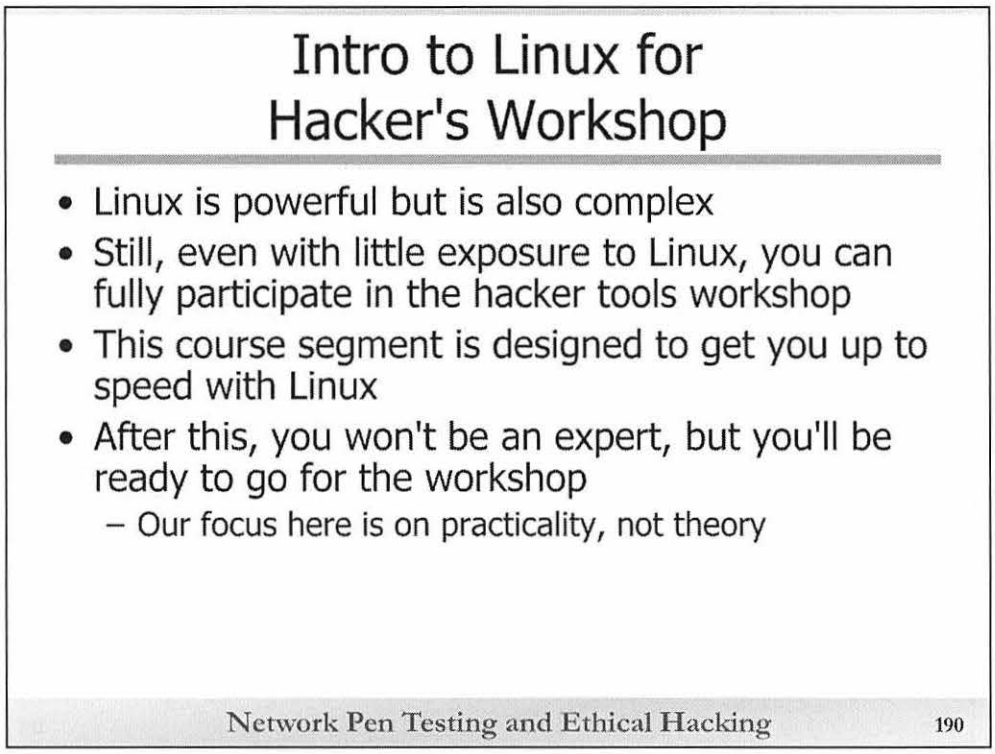


We now conclude our 560.1 section, in which we've addressed some important concepts in penetration testing and ethical hacking. Among the most important topics for the day have been proper scoping and formulation of Rules of Engagement. We also discussed the recon phase used in many penetration tests and ethical hacking projects, gathering information that will act as a firm foundation that testers will leverage for the remainder of a testing project. We've also configured our machines to prepare for the lab work we'll perform throughout the remainder of the class.

In our next section, 560.2, we'll take an in-depth look at scanning, the process used by penetration testers and ethical hackers to determine openings in the target environment.

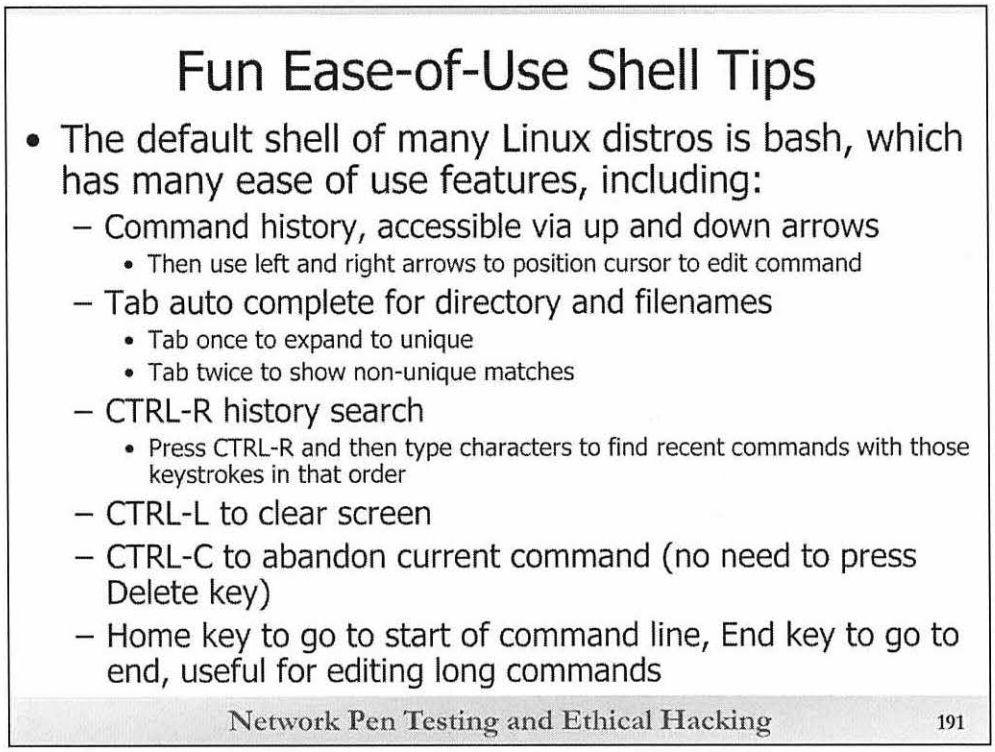


The next section contains an appendix that covers and Introduction to Linux. If you are new to Linux, you should review the section that follows because we will be relying on its concepts throughout the remainder of the class. If you already have a good working knowledge of Linux, you should not need in-depth review of the following materials, but a cursory glance at them may be a helpful refresher.



To fully participate in this class, you need a basic working knowledge of Linux. We're not expecting you to be an expert, by any means. Everything you need to know about Linux for the workshop will be covered in this introductory workshop.

We will not be covering Linux installation. You should have done that before coming to the session, as described in the class requirements.



Throughout this session, we use bash as a command shell, one of the most common command shells in Linux distributions today. This shell includes many ease of use features that make interacting with Linux simpler. You should memorize each of these items, as they will save you much time and effort, making Linux a lot friendlier for you.

Bash, like many other shells, remembers your shell history, letting you access it by pressing the up and down arrows to access and edit recent commands, which you can rerun by simply pressing Enter.

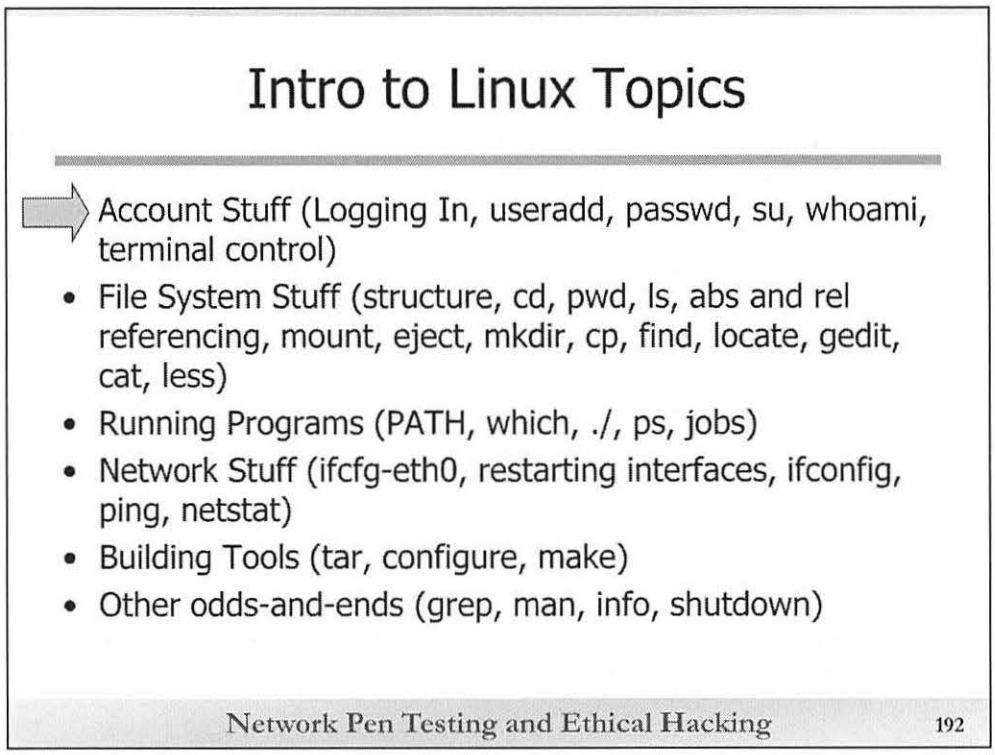
After you choose a previous command, you can press the left and right arrow keys to position your cursor to edit the command.

Also, bash supports tab auto-complete for the names of directories and files. When accessing something in the file system, just press Tab for the shell to expand it to a unique name that matches what you've typed so far. If there are multiple items that match what you've typed (that is, there is nothing unique yet), you can press Tab again to show the names of all files or directories in your current working directory that match what you've typed so far. That is, Tab expands to a unique value, and Tab-Tab shows all items that match what you've typed so far if nothing is unique.

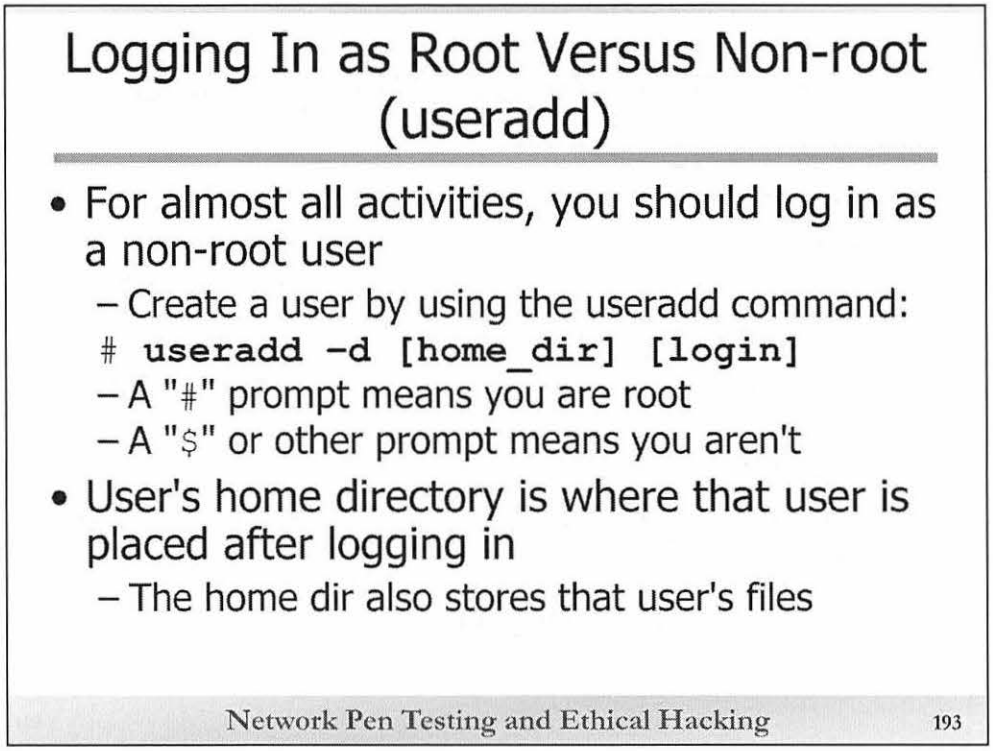
You can also search your history in bash by pressing CTRL-R at the start of a command line. Then, start typing characters, and bash jumps back to the most recent command that has the characters you typed in that order. You can then press Enter to rerun that command or the left or right arrow keys to edit the command.

The CTRL-L option clears the screen, or you can simply type **clear**. The CTRL-C command lets you abandon the current command and get back to the command prompt. There is no need to delete the current command by holding down the backspace or Delete keys. Just press CTRL-C to get rid of the current command.

The Home key included on some keyboards lets you jump to the beginning of a command line, whereas the End key lets you jump to the end. These options can help you jump around in long commands to make altering them easier.



Here's an outline describing the topics we'll cover. We'll start with Account stuff.



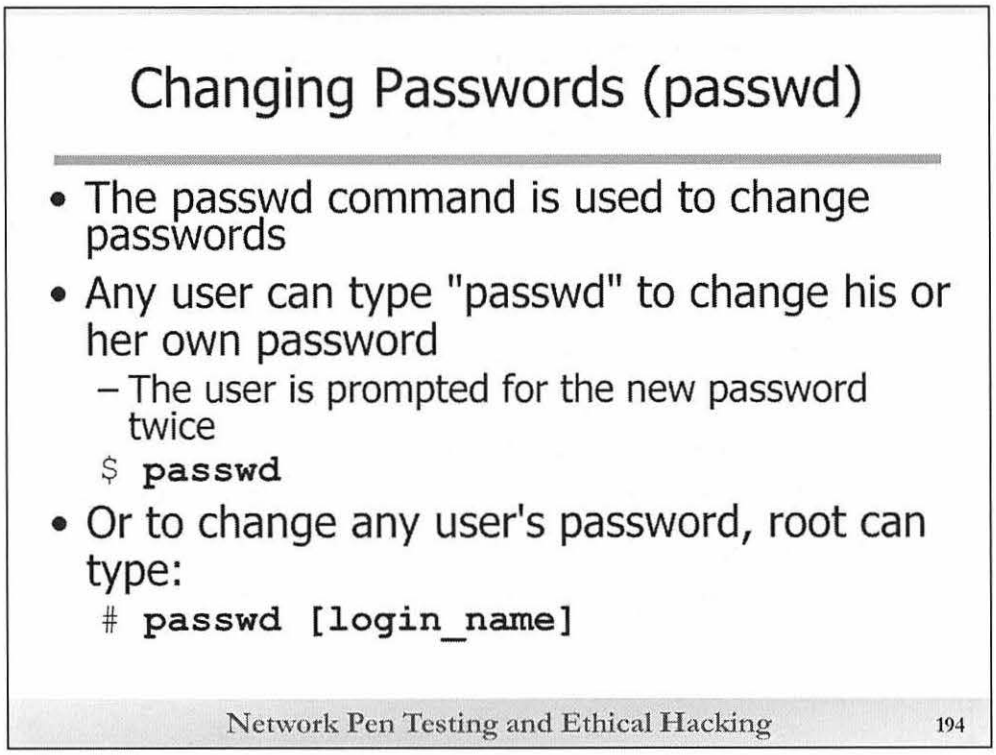
Go ahead and create a non-root account on your system.

Keep an eye on your prompt. If it's a "$", you just aren't root. If it's a "#", you are root.

As root, type the following:

```
# useradd -d /home/fred fred
```

The login account "fred" will be created, with a home directory of /home/fred. The system will automatically assign a non-root userID to the account. The userID is just a number associated with this account for the purposes of assigning permissions. The home directory is where config files and other personal files for this account are stored.



Currently, the fred account we created cannot be used because we haven't yet set a password. (The password isn't blank; the account is just disabled until we enter a password.) We need to set a password for the new fred account by typing:

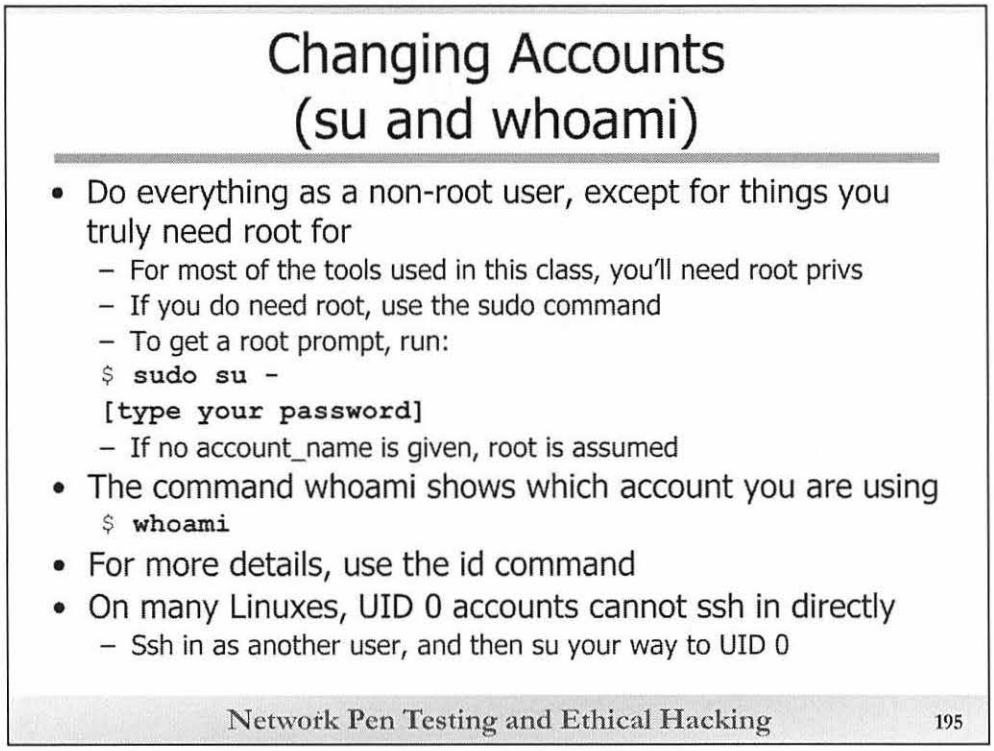
```
# passwd fred
```

[type account password here]

[retype account password to verify]

If fred wanted to change his own password, fred would type (from the fred account):

```
$ passwd
```



If you are logged in already, you run commands with the privileges of another account via the sudo command.

To get a root prompt, you could run:

```
$ sudo su -
```

Then, you can type in your accounts password, and it if has sudo rights to run a shell as root, you'll get a root prompt on your system.
The whoami command shows who you are currently logged in as.

Type the following:

```
# whoami
```

Given the # prompt at the beginning of this command, you will likely see "root" on the output. Try:

```
# su fred
```

(Notice that the prompt changed!)

```
$ whoami
```

Here, you should see that you are now "fred". You can exit your most recent su by running:

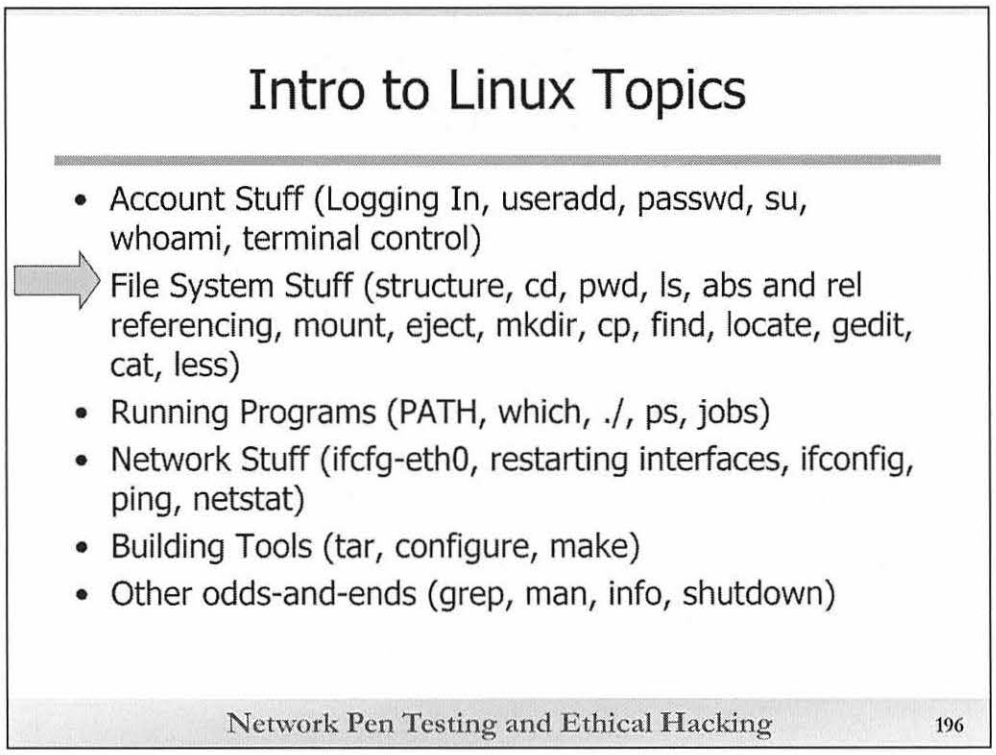
```
$ exit
```

(The exit means that we are leaving the user fred, and returning to root.)

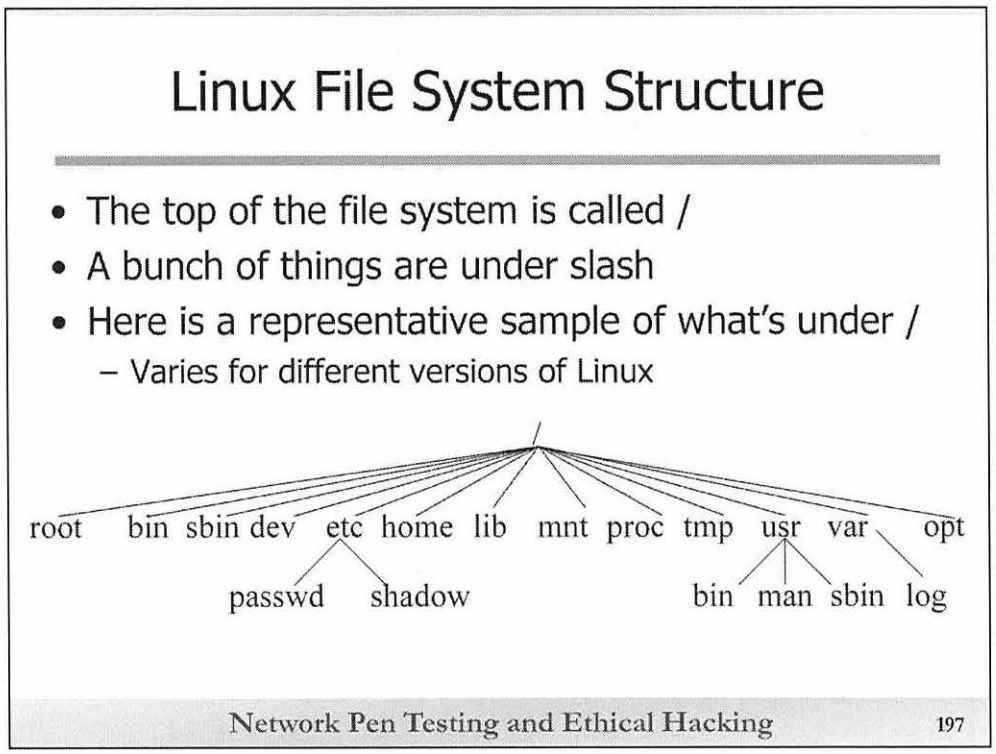
```
# whoami
```

Now, you should be root again (note the # prompt).
For even more details about your current user id and privileges, use the id command:

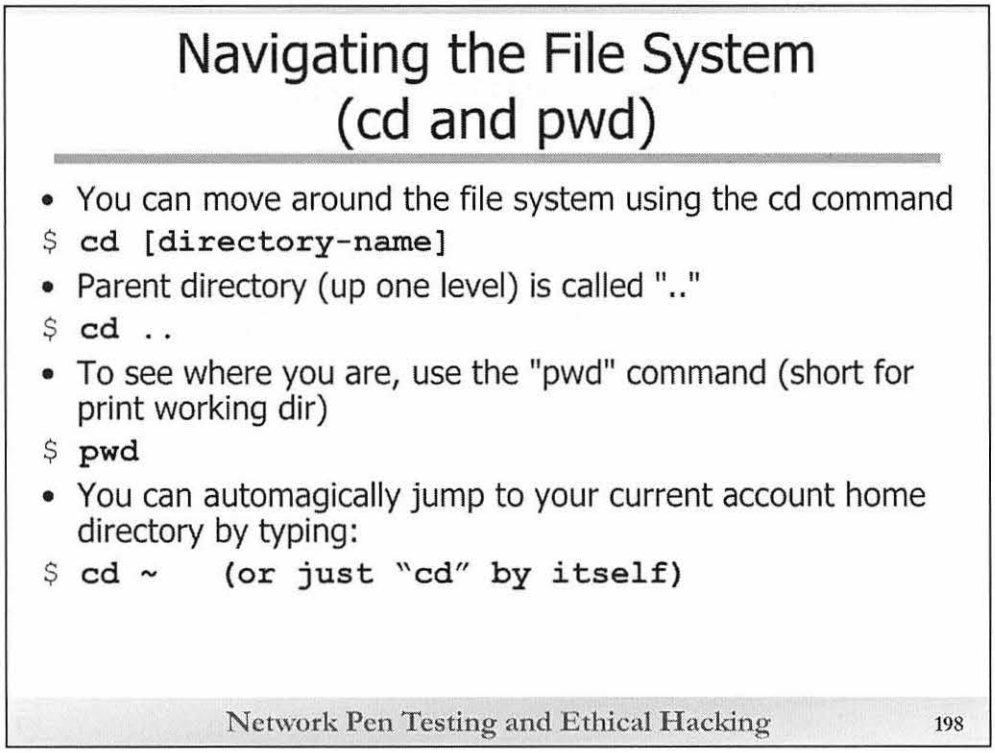
```
# id
```



Here's our pesky outline again. Let's cover File System Stuff next. This is the longest section, simply because so much of Linux is oriented around its file system.



- Executable program are stored in /bin and /sbin.
- /root is the root login account's home directory. This is hugely important because if you log in directly as root, this will be your initial location in the directory structure. If you log in as an individual user other than root, you'll be put in that user's directory, typically somewhere inside of /home.
- /dev stores devices (drives, terminals, etc.)
- /etc holds configuration items, like the account information (stored in /etc/passwd) and hashed passwords (stored in /etc/shadow).
- /home contains user's home directories.
- /lib contains common libraries.
- /mnt is where various remote and temporary file systems (CD-ROMs, floppies, etc.) are attached.
- /proc is a virtual file system used to store kernel info.
- /tmp is for temporary data, and is usually cleared at reboot.
- /usr holds user programs and other data.
- /var hold many different items, including logs (/var/log/).
- /opt stores optional items and is often a location for specialized tools that have been added to a distribution.



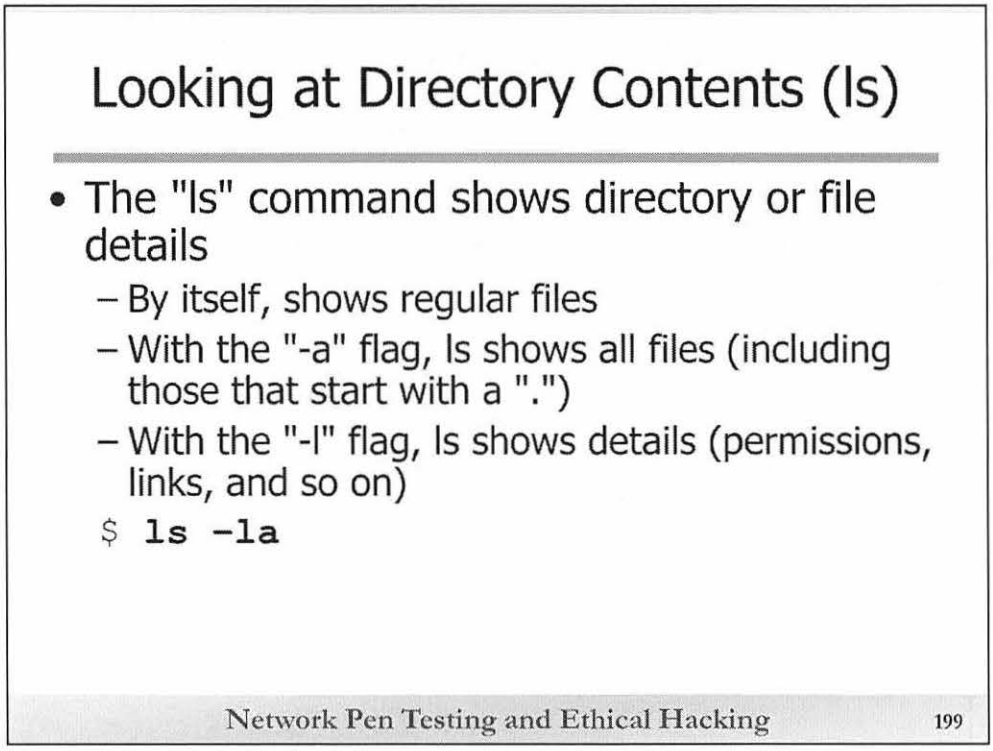
If you are following along, let's change to the top-level directory:

```
$ cd /
$ pwd
```

What do you see?

```
$ cd ~
$ pwd
```

What do you see?



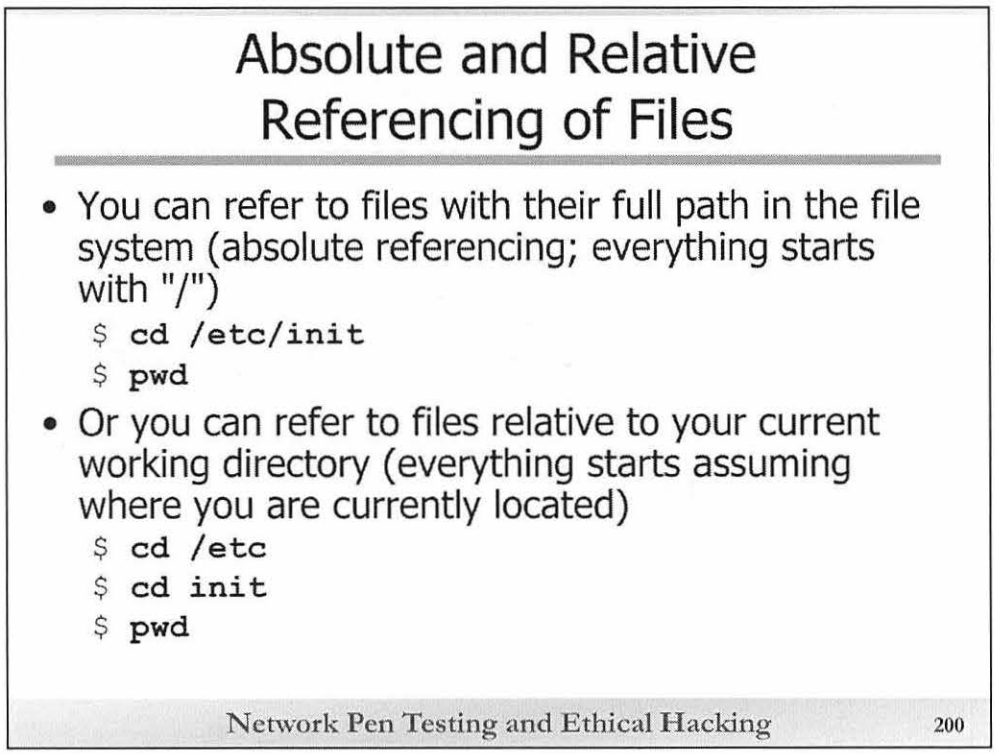
If you are following along, type:

```
$ cd /etc
$ ls
```

What do you see?

```
$ ls -la
```

You now are looking at details associated with your /etc directory. System configuration information is stored here.



For any file, you can refer to it using the relative reference (based on your current working directory), or the absolute reference.

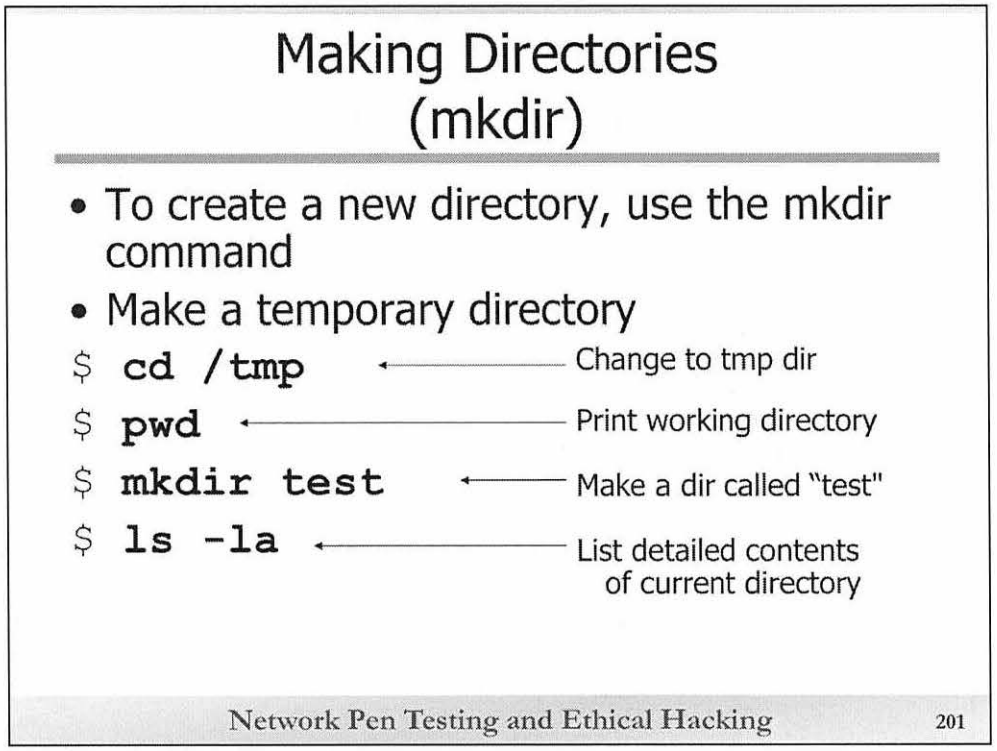
Try the following, using absolute referencing for the directory:

```
$ cd /etc/init
$ pwd
```

Or you can do it in two steps, using relative references:

```
$ cd /etc
$ pwd
$ cd init    ← Note that we dropped the leading /
$ pwd
```

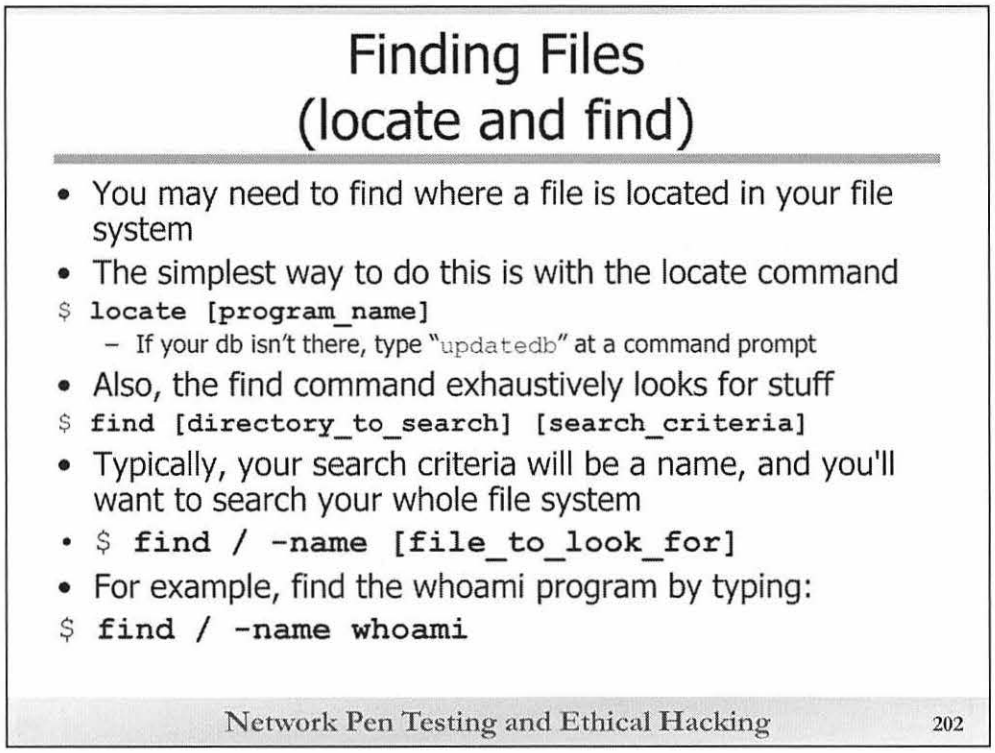
What do you see?



To create a directory, use the mkdir command. Let's create a test directory in our /tmp directory.

```
$ cd /tmp
$ pwd
$ mkdir test
$ ls -la
```

What do you see?



The locate command is an efficient way to determine where files are located on the system. It consults a local database installed and updated by the system administrator for files that are frequently sought. It runs quickly and doesn't consume a lot of resources. However, it cannot locate items that are not loaded into its database.

To try locate, type:

```
$ locate whoami
```

If your system complains that there isn't a locate database or that it's out of date, you can manually update the database by typing the command:

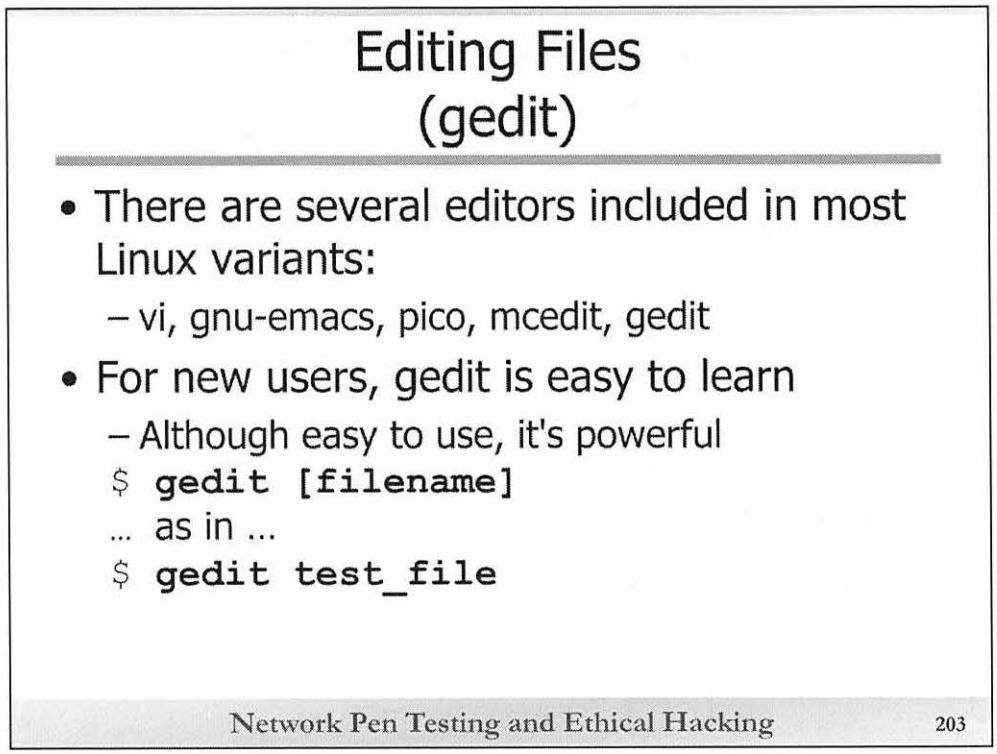
```
# updatedb
```

To do a comprehensive search of the directory, you can use the find command. This command consumes a lot of resources. Several finds running simultaneously will slow a Linux system to a crawl. Still, find is the best way to find something if locate doesn't work.

Let's try to find a file on the file system. Type the following:

```
$ find / -name whoami
```

What do you see?



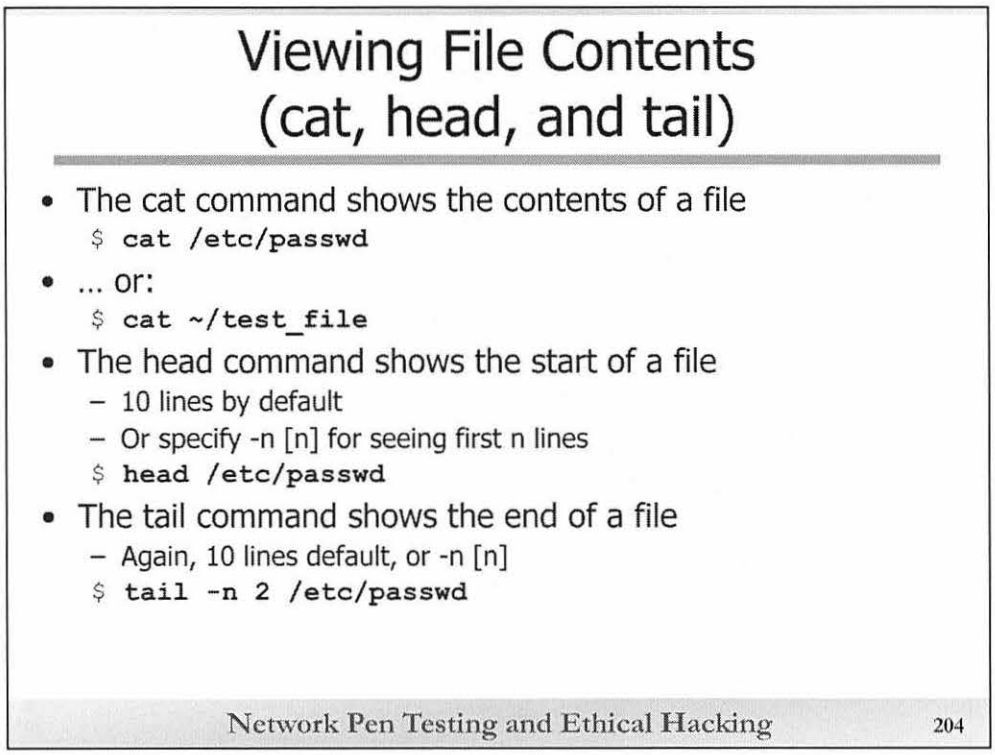
You may need to edit a file at some point. You can use any editor you are comfortable with. If you are new to Linux, you should consider using gedit, one of the easiest editing tools commonly installed in Linux. If you have a GUI, you can use gedit.

Let's create and edit a file:

`$ cd ~`	(change to the home directory)
`$ gedit test_file`	(let's edit and create a file named "test_file")

Now, edit your file. Type in a bunch of junk. Use the function keys to save it.

I told you gedit was easy!



So, you just edited a file. How can you see its contents? You can use the cat command:

```
$ cat ~/test_file
```

Also, you can look at other files:

```
$ cat /etc/passwd
```

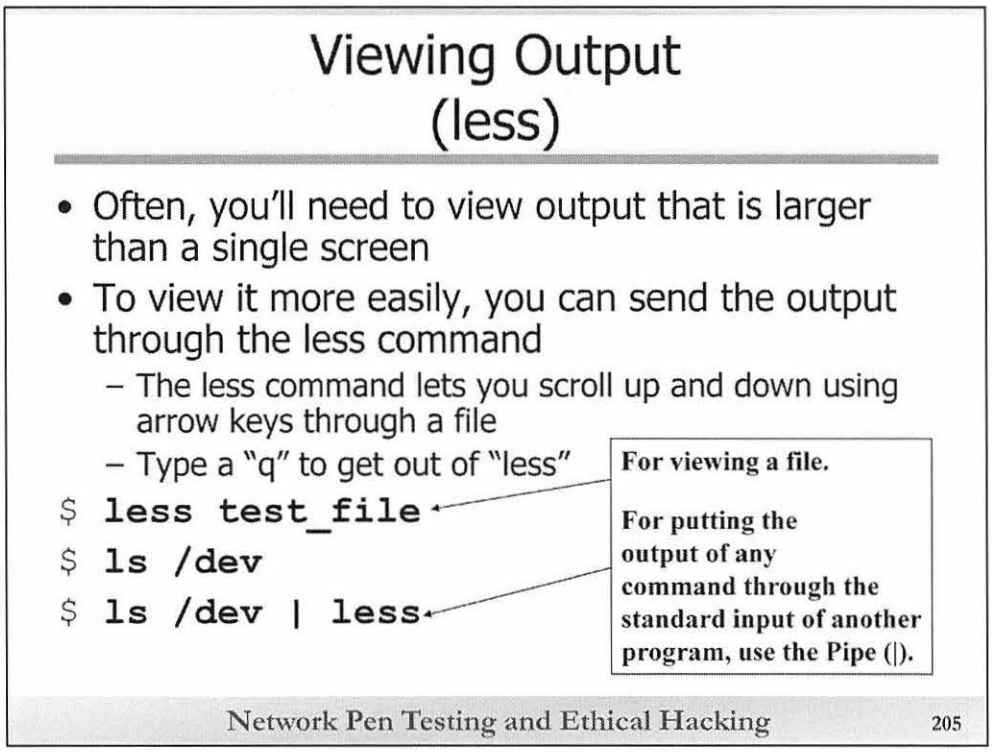
This shows the contents of the password file! (Note that on most Linux installations, the passwords are stored in another file, /etc/shadow). Typically, in most modern UNIX installations, /etc/passwd just contains account information.

Alternatively, we can view portions of files using the head or tail commands. The head command shows the first 10 lines of a file by default. By specifying head -n [n] [filename], we can view just the first n lines. Similarly, the tail command shows the last 10 lines of a file by default, or we can use the -n [n] syntax to view a different number of trailing lines. Consider the following commands:

```
$ head /etc/passwd
```

```
$ head -n 1 /etc/passwd
```

```
$ tail -n 2 /etc/passwd
```



In addition to cat, there are other commands you can use to look at files. The less command is one of the best to use. Try typing:

```
$ less test_file
```

You should see the contents of the file.

In addition to looking at files, the less command can also be used to help look at lengthy output from a command. Try typing:

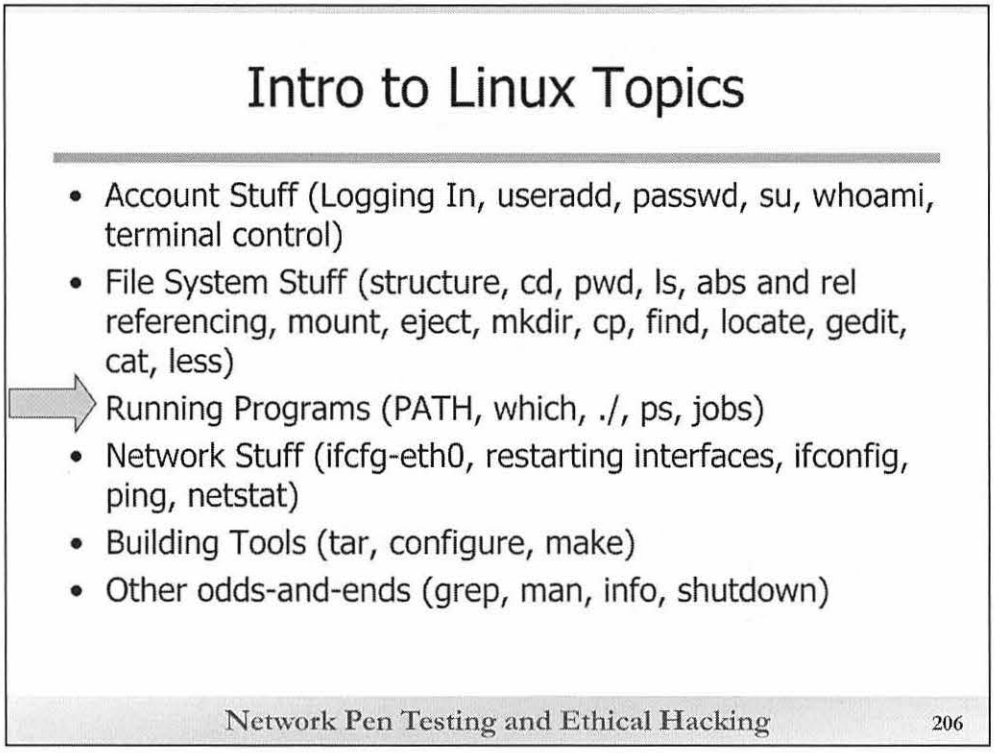
```
$ ls /dev
```

This shows you all the devices (virtual and otherwise) on your system. It's a long, unwieldy list. The less tool lets you interact with this output in a better way.

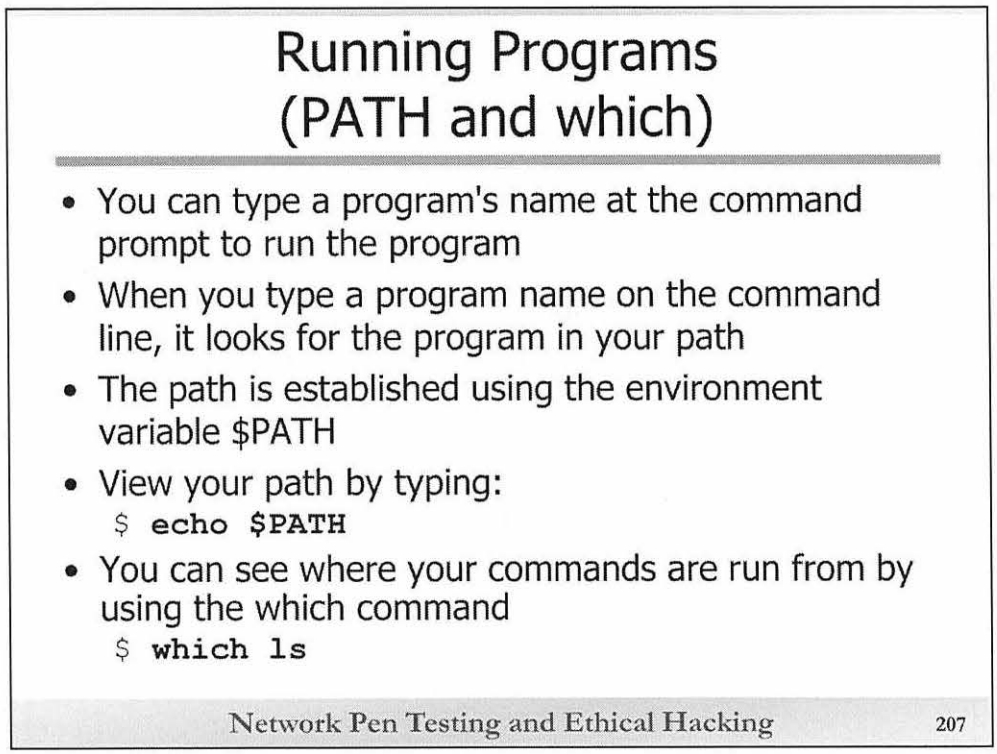
Type:

```
$ ls /dev | less
```

By piping the output of ls through less, you can now use the cursor keys to scroll up and down through the output. The space key jumps forward one page. Use the "q" key to quit. The pipe takes the output of one program and feeds it into the standard input of another program.



You guessed it ... another outline slide. We will now discuss running programs. This section is important (not that the other ones aren't important ...). People frequently mess up on this stuff and get confused because Linux works differently from Windows in running programs.



When you type a command at the prompt, the system looks in your PATH to find the right program to run.

Look at your path by typing:

```
$ echo $PATH
```

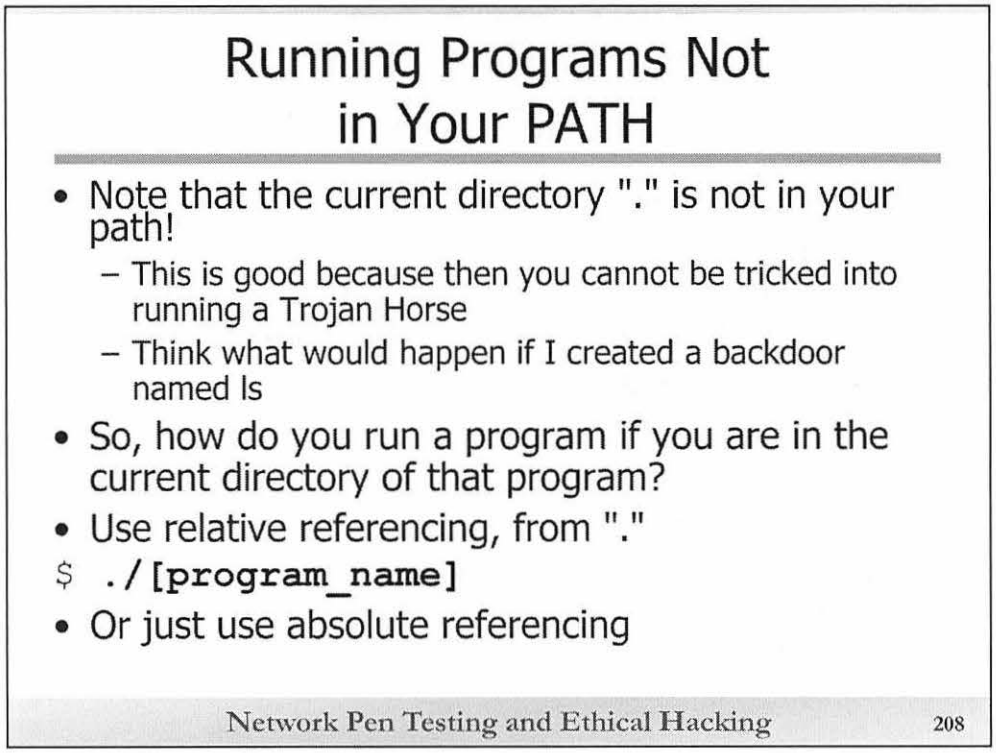
The echo command means "type the following." The $ before path means, "What follows isn't a string of characters; it's a variable." The variable we want to type is our PATH.

The result is a list of directories where the system searches for programs based on what we type at the command line. These directories are separated by a ":". If you type a program name at a command prompt, and the program isn't in your PATH, the system will tell you that it cannot find the program. You have to either refer to it absolutely or relatively, or add its directory to your PATH.

If you want to see where in your PATH a command has been found, you can use the which command. Try typing:

```
$ which ls
```

That's where your "ls" program really is!



This is an important point that confuses people, because UNIX functions differently from Windows on this issue.

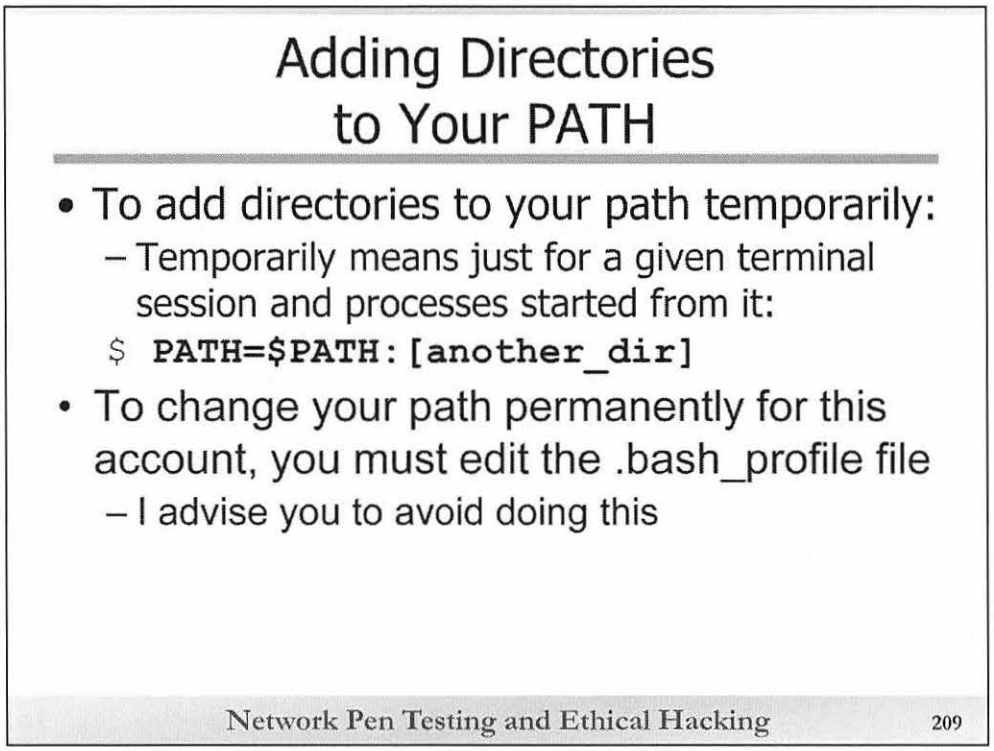
For security reasons, your current working directory (the one shown by "pwd"), also referred to as ".", is not in your PATH. That's a good thing! If "." were in your path, an evil attacker could name an evil Trojan Horse program ls and put it in your home directory. When you ran ls to look at your home directory's contents, you'd run the evil trojan horse! For this reason, "." isn't in the path by default and shouldn't be put in your path.

This also means that if you change directories to a place in which a program file is located, you cannot just type the program's name to run it. Instead, to run the program, you have to type ./[program_name] to run it.

If the system ever complains that it cannot find a file but you can see the file in the current working directory using ls, you likely just need to start the program by typing:

```
$ ./[program_name]
```

On Windows machines, the current working directory is in your path. Therefore, if you change to a directory with an executable and type the executable's name on Windows, the program runs. Yes, it's convenient .... However, it's a security hole!



Although we DO NOT recommend it, you could add a directory to your PATH temporarily. Type the following:

```
$ echo $PATH
```

Look at your path. To change your path temporarily, you could type (NOT RECOMMENDED):

```
$ PATH=$PATH:/[another_directory]
```

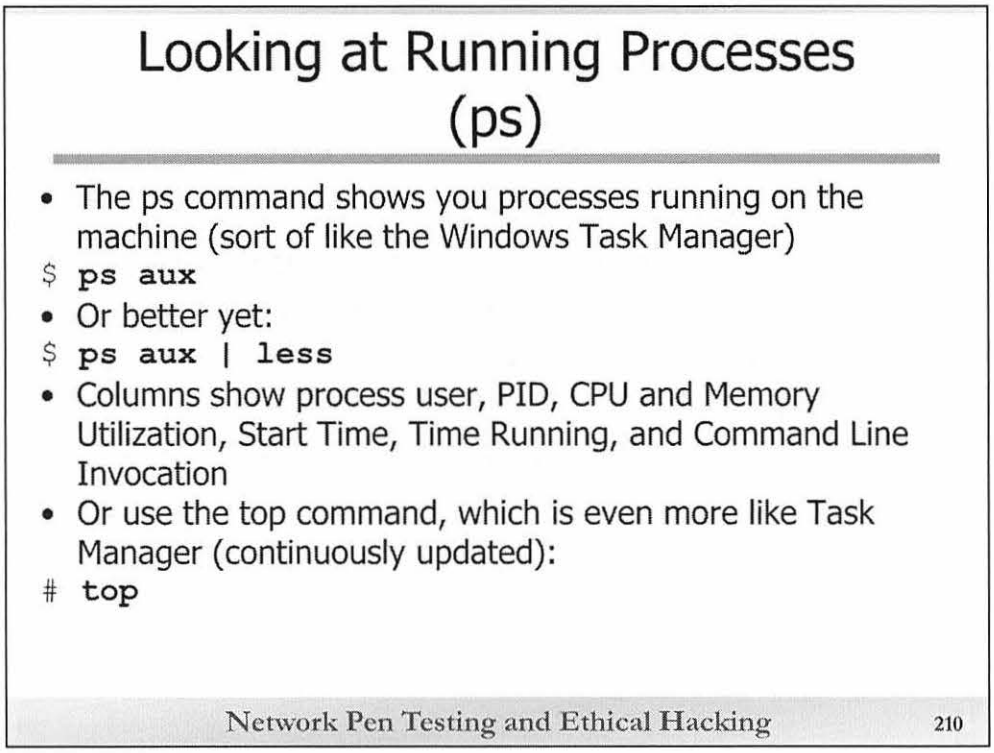
Now, type:

```
$ echo $PATH
```

Your path will now include the additional directory at its end.

This change applies only to this terminal and any processes started from this terminal. When you logout, this change goes away, and your path has its original settings.

To permanently change your path, you must edit the ~/.bash_profile file. I advise you to avoid editing this file if you are new to Linux. The default path setting is good for most purposes.



Sometimes, you need to see what processes are running. The ps command shows you a bunch of info about all running processes. Try typing:

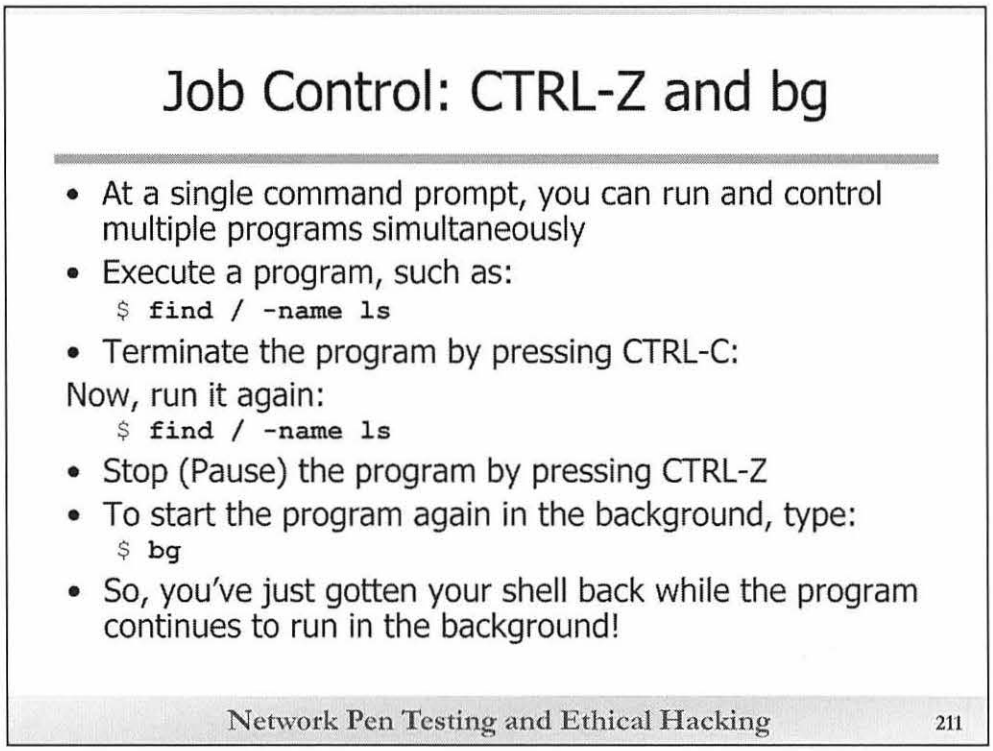
```
$ ps aux
```

You get an exhaustive (and exhausting) list of all running processes.

Let's use our little "less" trick to make this output more readable:

```
$ ps aux | less
```

Now, you can scroll up or down and get a better feeling for what's running on your system.



You can temporarily pause programs with CTRL-Z and get your command prompt back. This is quite useful because you can get your command prompt back to run more programs if you want.

Also, you can restart the paused program running in the background with the bg command. The fg command starts it running in the foreground, as you might expect.

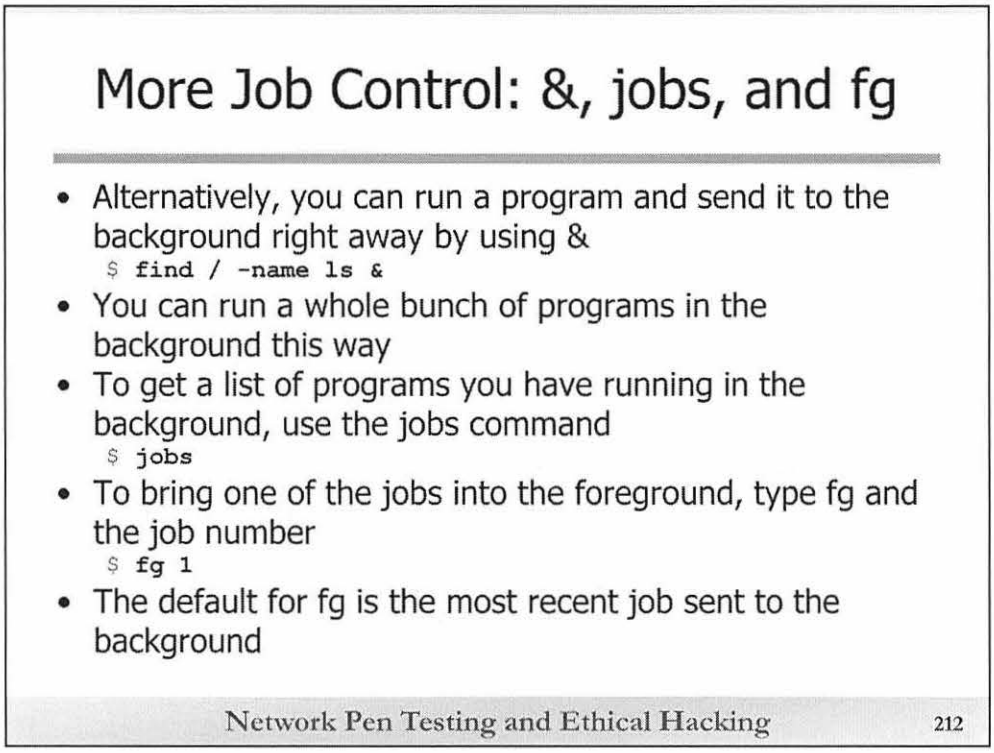
Let's try it. Type:

```
$ find / -name ls
```

Before it finishes running, press `CTRL-Z`.

Now, restart the program in the background by typing:

```
$ bg
```



The jobs command gives you a list of all programs you have kicked off that are running in the background. The fg command can also be used to restart a specific paused program in the foreground, but giving the job number after the fg command.

If the find command from the previous slide has finished, type the same command again, but this time run it in the background using the & after the command invocation.

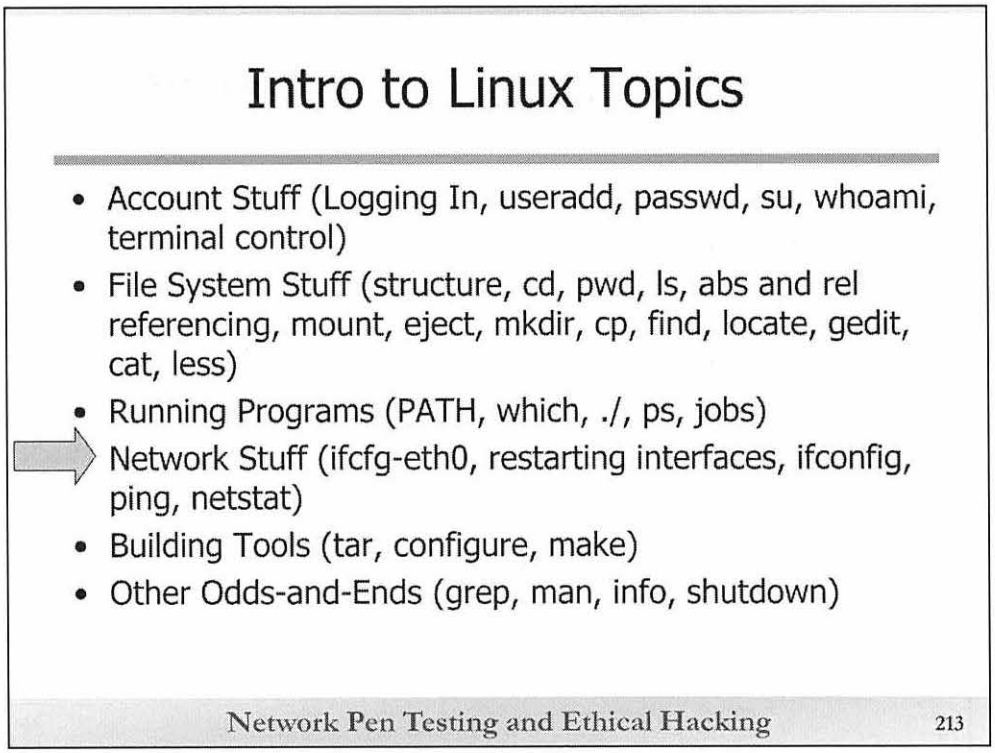
```
$ find / -name ls &
```

As it runs in the background, type the jobs command:

```
$ jobs
```

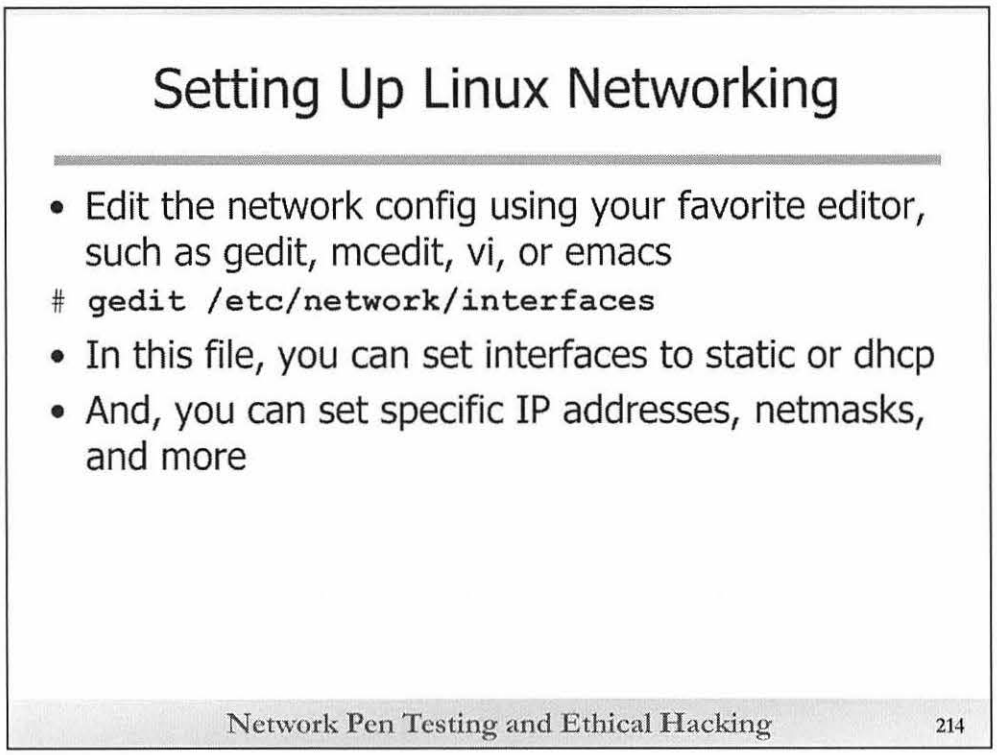
Look at the job running. You can move it to the foreground by typing:

```
$ fg 1
```

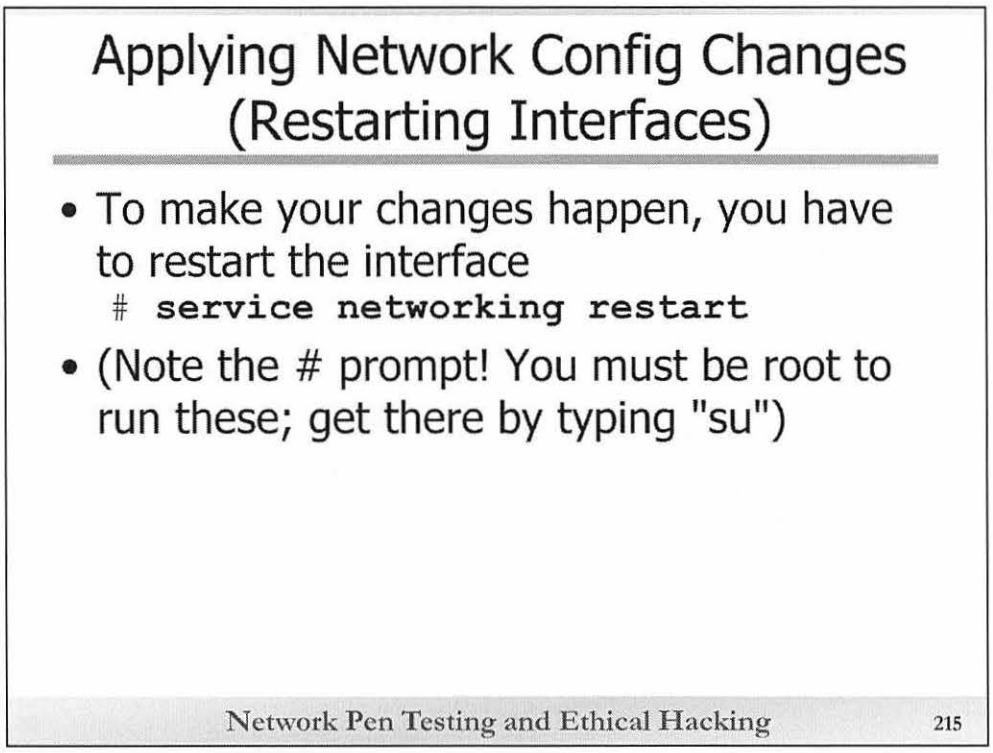


Another outline slide. Gee, these outlines are fun.

Now, we will cover getting and staying networked in Linux.

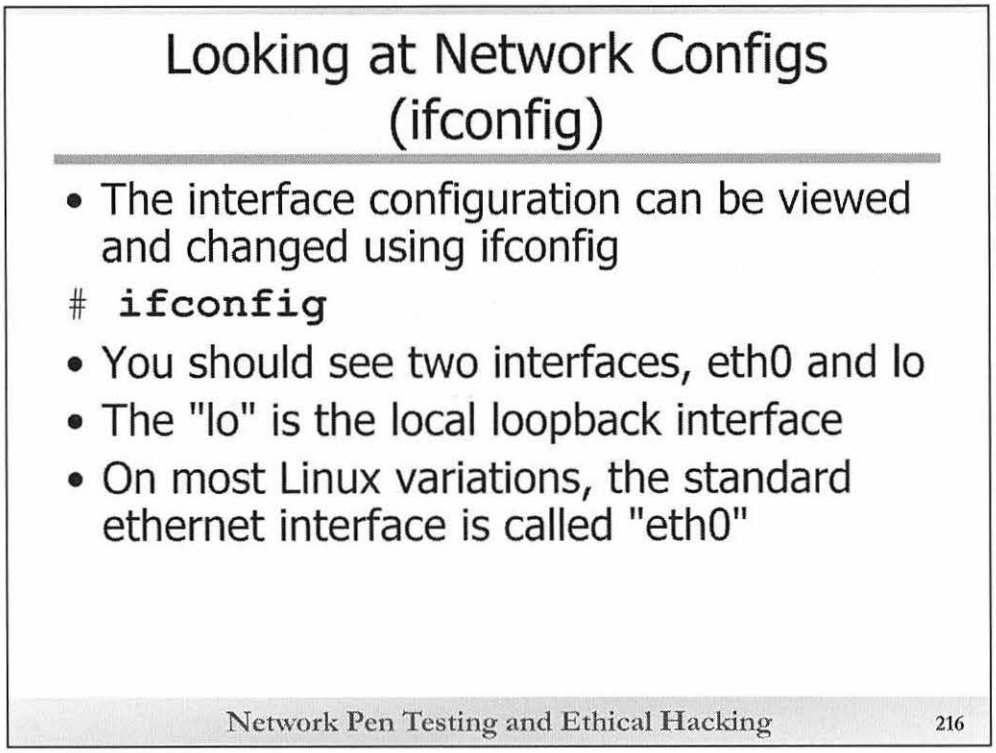


To set your network interface options in Linux, you can edit the /etc/network/interfaces file. By putting in the appropriate information, you can configure your interface for static addresses or dhcp.



If you change the interfaces file, your changes will not be applied to the interface immediately. Instead, you need to restart your interface. Type (as root):

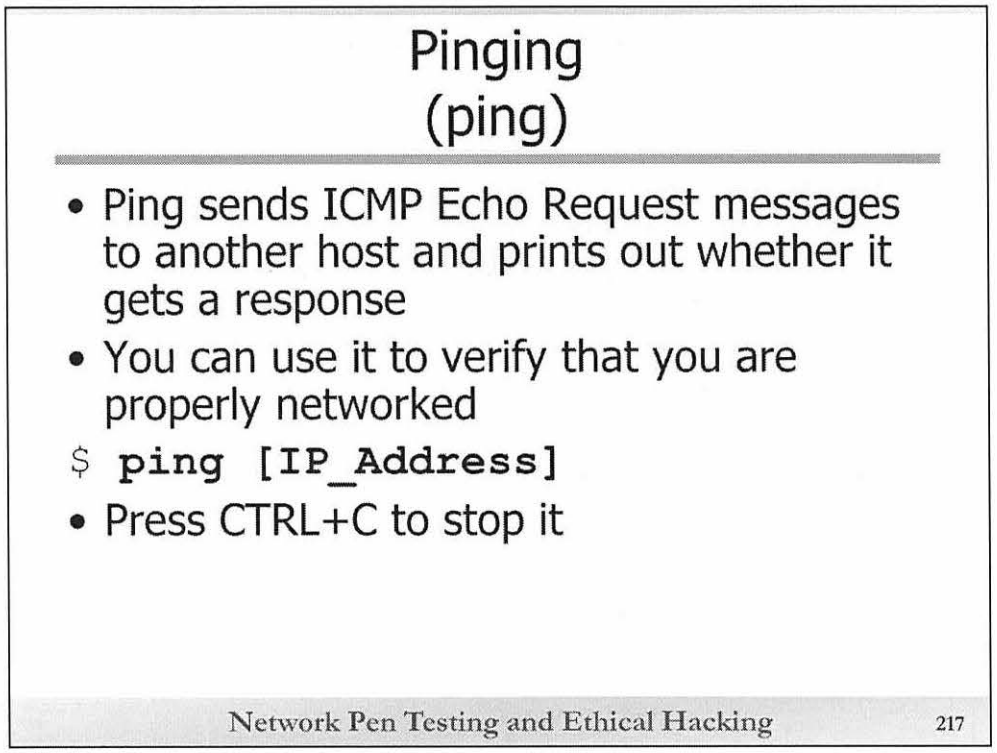
```
# service networking restart
```



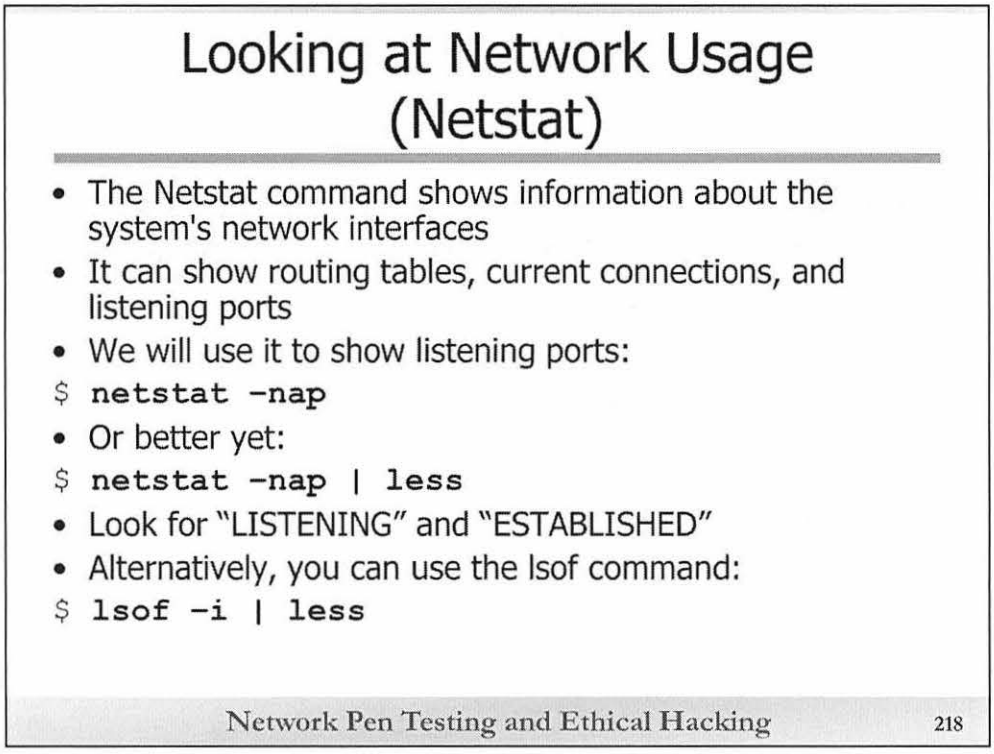
Let's see if our interface changes were applied to the system. To look at your interface configuration, type:

```
# ifconfig
```

You see your IP address, netmask, MAC address, and various other nifty items. If you have one ethernet card, you see two interfaces" the local loopback interface with the address 127.0.0.1 and your ethernet interface, called eth0.



To verify that you are properly networked, you can ping another machine. The ping command is similar, but not identical, to the Windows ping program. One of the biggest differences is that a Linux ping keeps sending pings until you press <CTRL+C> to stop it. By default, the Windows ping sends out four ICMP Echo Request packets and then stops. Linux just keeps going until you stop it.



Now look at what's using your various TCP and UDP ports. Type:

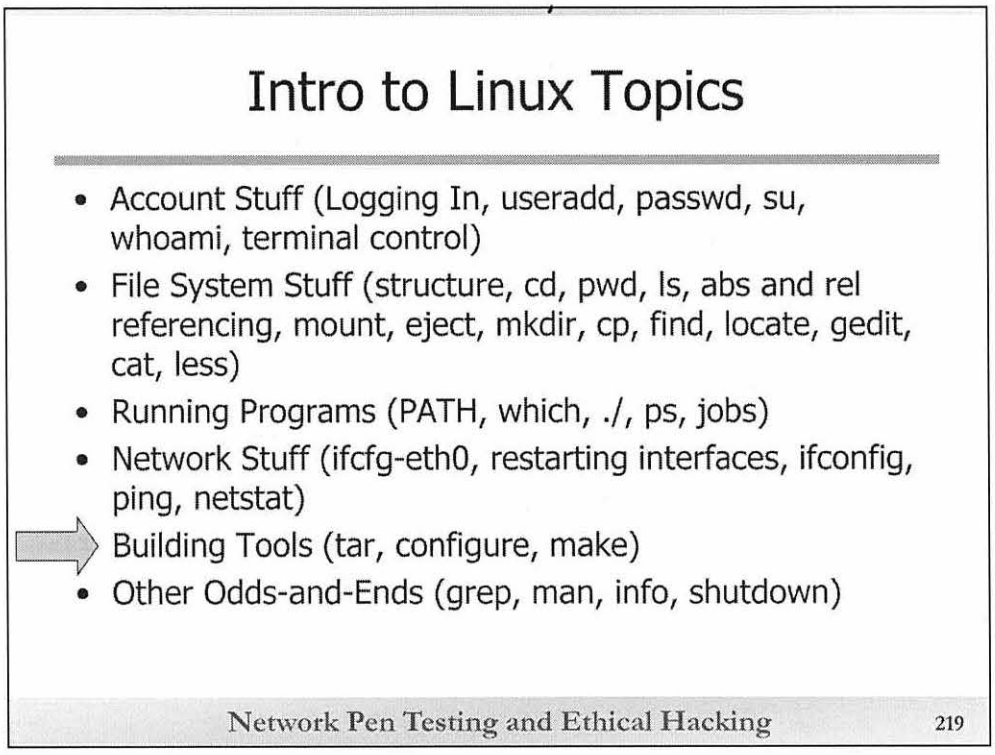
```
$ netstat -nap
```

There's a lot of stuff there. It can be a bit difficult to read as it scrolls by, so try this:

```
$ netstat -nap | less
```

You can scroll up and down through the output. We'll discuss how to do better searches through this later.

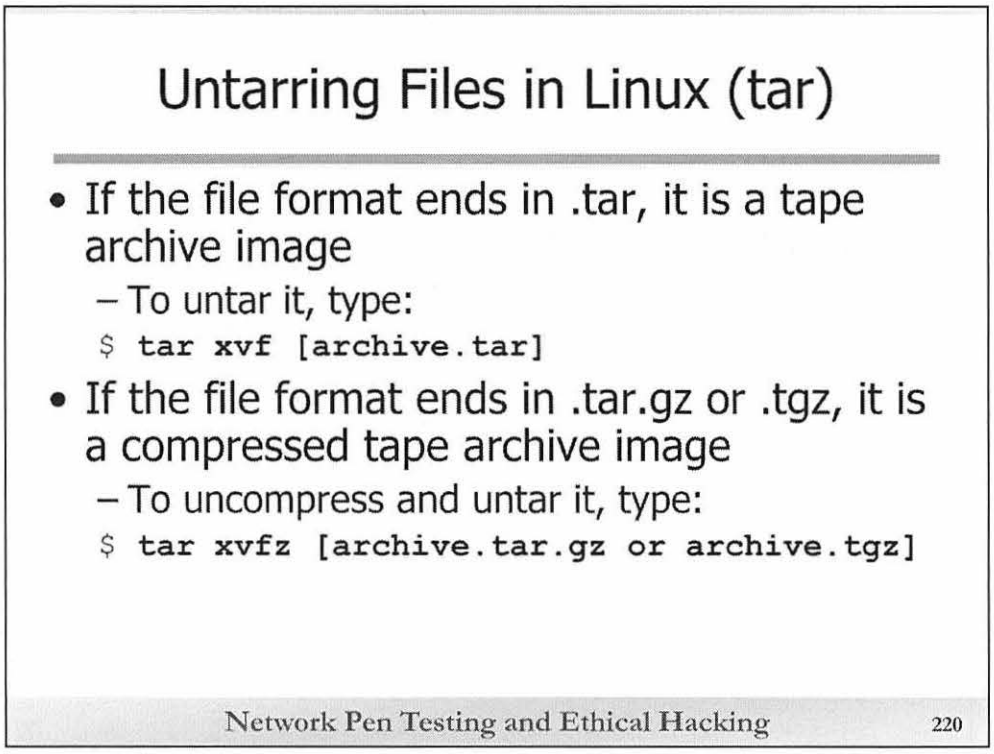
Note that various TCP and UDP ports are shown as LISTENING. These are waiting for a connection. Others may indicate that they are ESTABLISHED. These have existing connections.



You know, to use Linux in the workshop, you have to run tools. To run them, in many cases you need to build and install them. As you see from the outline slide, we'll cover building and installing tools next.

Some programs are installed by using tar files. Others are rpms. Still others use "configure" and "make." How do you know which tools use which format? Most of the tools include a README file. Look at the README file (using cat, less, or gedit) for instructions on installing the tool. The course notes for the main class also include directions for compiling and installing.

By the way, we have you compile and install the tools so that you can get experience with doing these tasks. In the wild, you may need to compile and install new versions of these and other tools, so we want to get you ready.



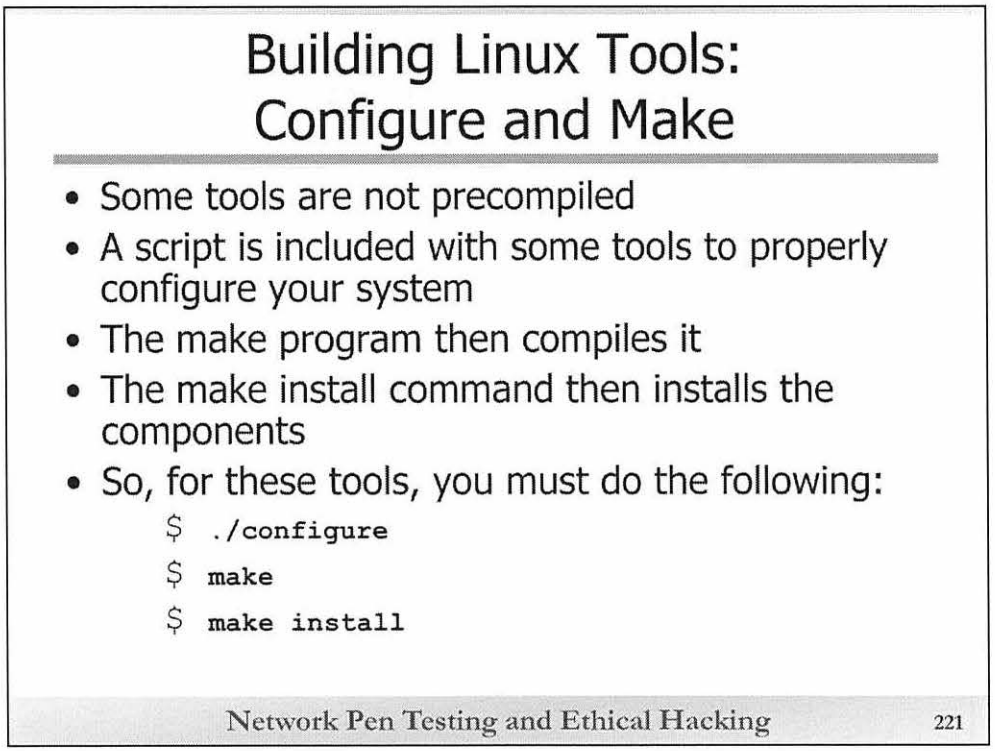
Some tools are stored at tape archives, abbreviated "tar." This doesn't mean they were on physical tapes; the lingo just lingers from the olden days. Although tapes may or may not be used, tar files are used all the time. Think of them as being like ZIP archives in Windows. You take a bunch of files and glom them together in a tar file.

To open a tar file, you use the xvf parameters. x means "Extract." v means "Be verbose; give me a lot of output to let me know what's going on." f means "Get this from a file."

If the tar file has been compressed using a tool called "gzip," its name will end with a suffix of .tar.gz or simply .tgz. To open these, you need to use the tar command with the xvf and z flags. The z flag means "Unzip this before you open the archive."

When the archive opens, all files and directories associated with it will be automatically created in the current working directory and below.

We won't demo this during the Intro to Linux mini-workshop. You'll get a chance to use tar files during the main class.



Although some programs ship as tar files and others as RPMs, many just ship with a script called "configure." You need to run this script first, which checks your environment and creates a set of options necessary to get the tool compiled on your device. After running configure, you run the make command, which compiles and builds the tool. Then, by typing **make install**, the program is loaded into the appropriate place.

We won't demo this during the Intro to Linux mini-workshop. You'll get a chance to use configure and make during the main class.

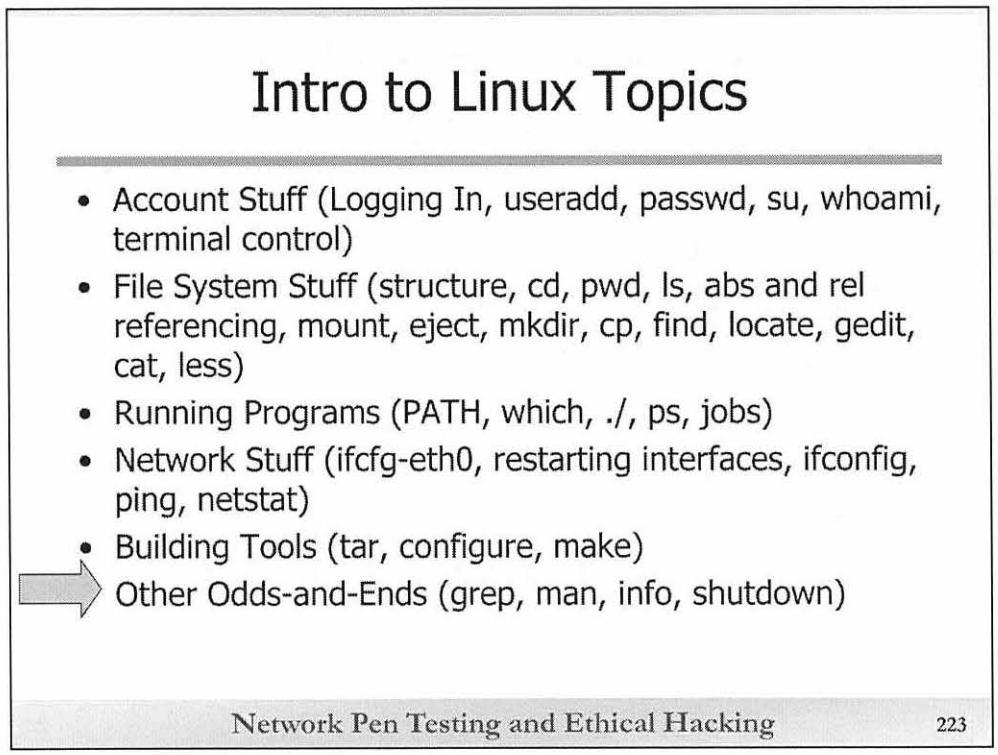
Building Linux Tools: Make

- For some tools, there is no configure script
- You simply use the make program to compile it

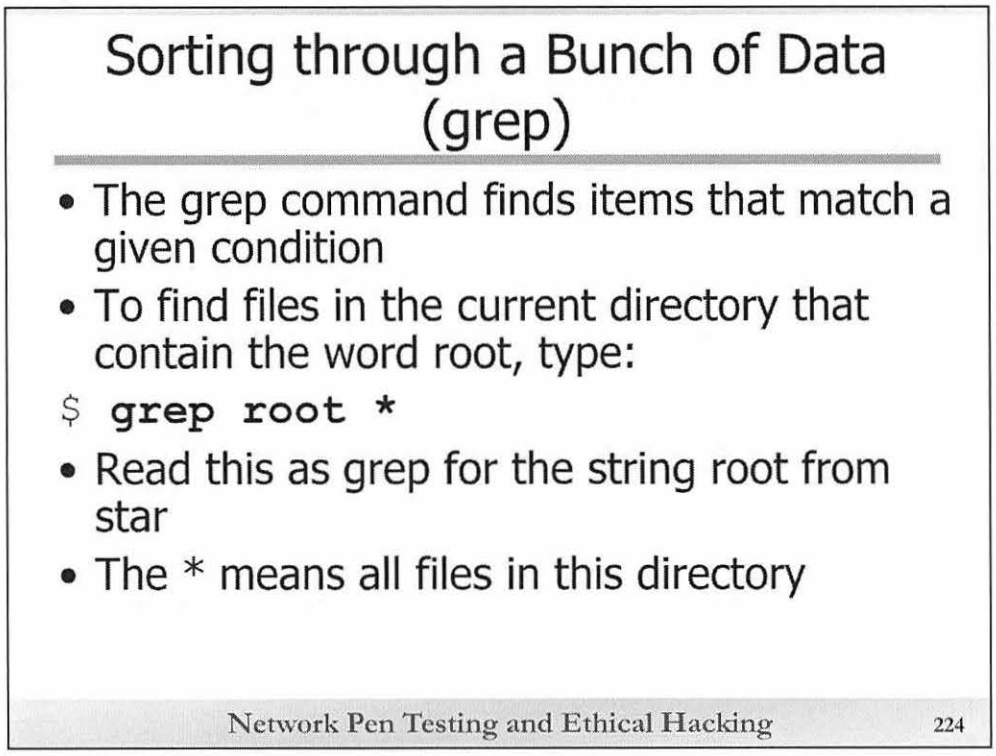
```
$ make
```

Network Pen Testing and Ethical Hacking 222

Some tools don't have a "configure" script. For these, you just run the `make` command.



Here are some other odds and ends that can help us use Linux throughout this course.



Grep is a powerful tool for finding data. It can look through files or the output from commands to identify particular strings. We will just scratch the surface of its use.

To look for a given string in a set of files in a directory, you can type:

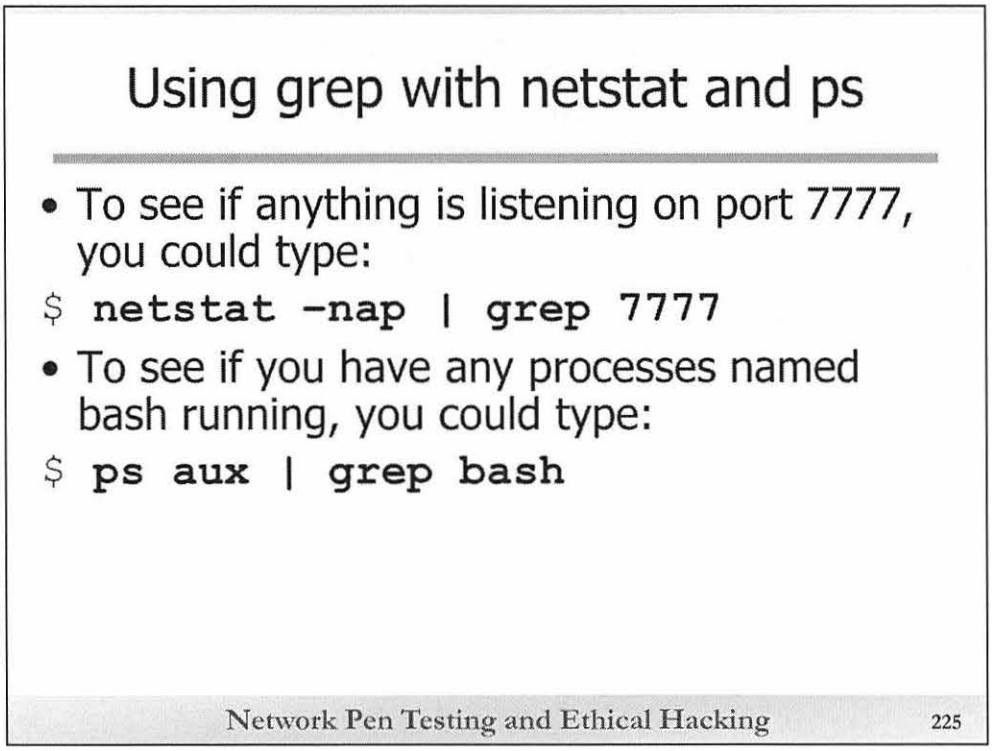
```
$ grep root *
```

This prints all occurrences of the word "root" and the file in which it appears in the current working directory. Try the following:

```
$ cd /etc
```

```
$ grep root *
```

See the word "root" in any files here? Which ones?



Grep can help isolate information about the usage of particular ports and processes.

At the command prompt, type:

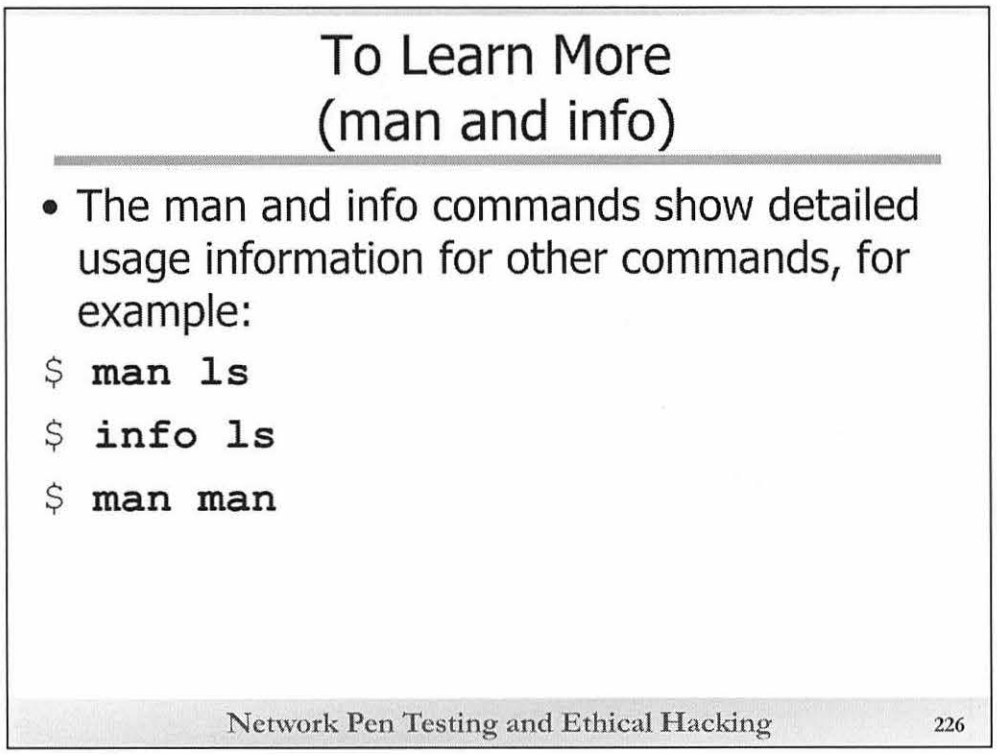
```
$ netstat -nap | grep 7777
```

This says, "Run the netstat command to show me TCP and UDP port usage, send the output to grep, and have grep show me any lines with the string 7777 in it." The results indicate if anything is listening on or using port 7777.

Likewise, you can use grep to help you find particular programs. At a command prompt, type:

```
$ ps aux | grep bash
```

This shows you all processes running the bash program (the command shell you are running) that are currently executing on your system.



To learn more about Linux, you can use man or info.

Try the following:

```
$ man ls
```

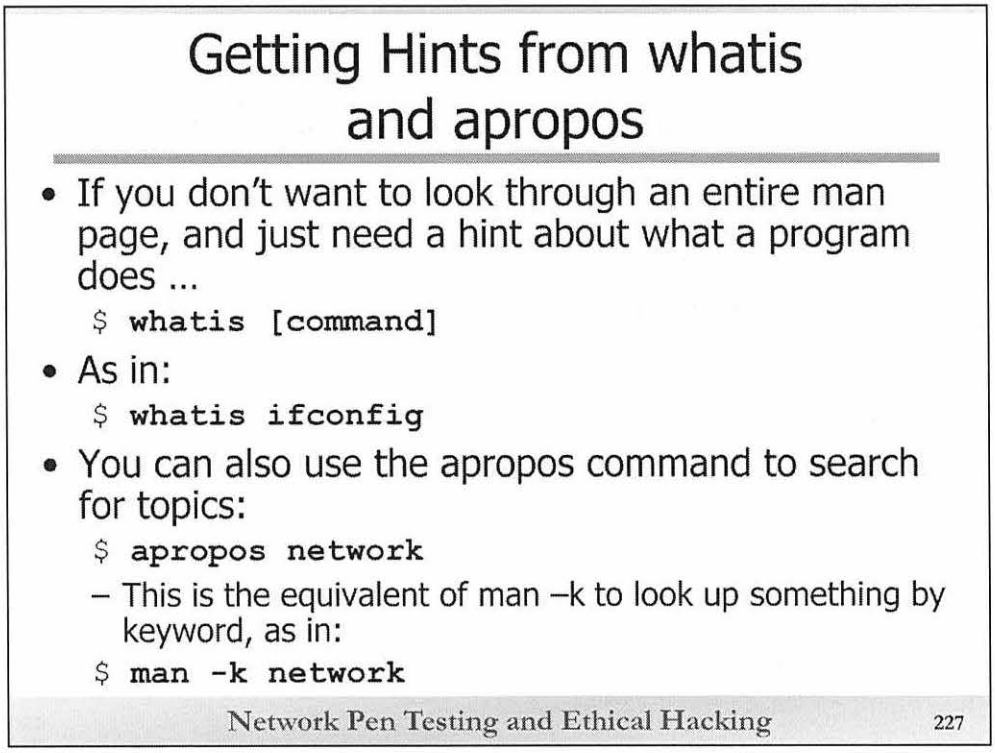
Interesting ... and chilling. The `ls` command is complex!

Also, try:

```
$ info ls
```

And, check this out to learn more about man:

```
$ man man
```



The whatis command is useful for getting hints from the system about what various commands do. It won't change your life, but it might just jog your memory about some esoteric command.

I usually just use the man page, but some people prefer whatis.

Try typing:

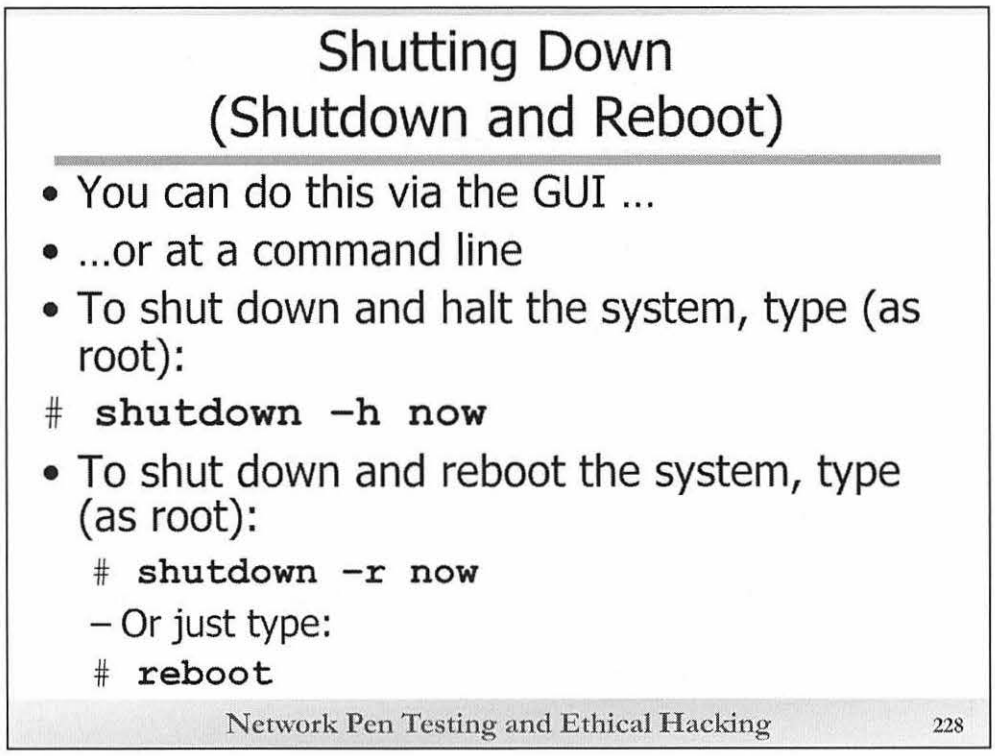
```
$ whatis ifconfig
```

You can also use the apropos command to search for topics and the commands related to those topics:

```
$ apropos network
```

This is the equivalent of man –k to look up something by keyword, as in:

```
$ man -k network
```



When you are done with Linux, you should shut it down gracefully.

You can do this from the GUI, but I usually just do it from the command prompt.

As root (you may need to su!), to gracefully shut down your system, type:

```
# shutdown -h now
```

The -h flag means "halt" the system. Of course, "now" means do it right away. You can actually schedule the system to shutdown at another time using this command, too.

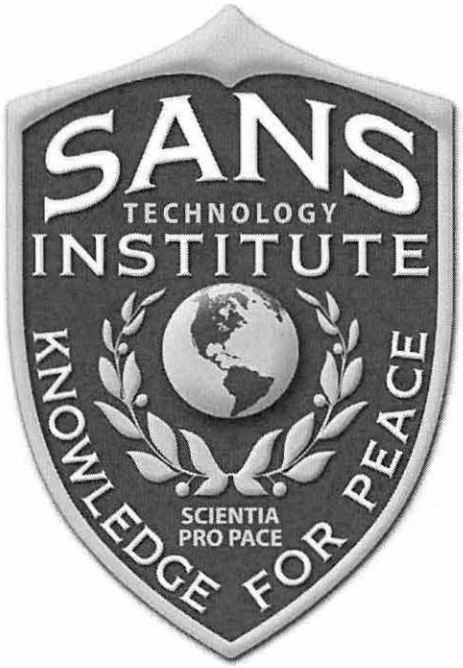
You can also use the shutdown or reboot command to reboot the machine. To reboot, I usually just type:

```
# reboot
```

Intro to Linux: Conclusions

- You have the building blocks you need to participate in the full class
- Linux is powerful but sometimes frustrating
- Refer to this section during the main class
- Ask for help from instructor/proctors/mentors if required

You are now ready for the full class Linux labs! Use your new-found Linux skills for good, not evil!



This Course is part of the SANS Technology Institute (STI) Master's Degree Curriculum.

If your brain is hurting from all you've learned in this class, but you still want more, consider applying for a Master's Degree from STI. We offer two hands-on, intensive Master's Degree programs:

- Master of Science in Information Security Engineering
- Master of Science in Information Security Management

If you have a bachelor's degree and are ready to pursue a graduate degree in information security, please visit www.sans.edu for more information.

www.sans.edu **855-672-6733** **info@sans.edu**