

HACKING 101

Técnicas básicas de ataque y Defensa







¿ Quien?



Jorge Alejandro Cajas Mota

jac.mota@gmail.com @cajasmota https://github.com/Motojo/Slides







¿ Quien?



















Contenido

Temas

- ¿Hacker?
 - Definición
 - Tipos
- ¿Por donde empiezo?
 - Herramientas
 - Reconocimiento
 - Vulnerabilidades
 - Pruebas
 - i Lotería!
- Vulnerabilidades comunes
 - XSS
 - SQL Injection
 - Web Services
 - Frontend validations
- Bonus



¿Hacker?

Definición

- Es un Experto en una Ciencia
- Le apasiona el conocimiento
- Se divierte aprendiendo cosas nuevas
- Le gustan los retos difíciles.
- Es revolucionario, comparte su conocimiento

- NO es un pirata informático
- NO es un criminal
- NO es un intruso en sistemas ajenos



¿Hacker?

Tipos



¿Hacker?

Tipos





Herramientas















Herramientas









Reconocimiento

- ¿Quién es la Víctima?
- ¿Qué protocolos de comunicación utiliza?
- ¿Están usando algún framework?
- ¿Tienen alguna vulnerabilidad común?
- ¿Qué espero obtener?

ANONIMATO





Vulnerabilidades

- OWASP Vulnerabilities List
- Exploit DB
- Mala Configuración
- Malas practicas de programación
- Ingeniería Social
- Exposición de datos sensibles



Pruebas

- No dejemos un rastro, puede ser problemático.
- No violemos la ley, también es problemático.
- Usemos las herramientas adecuadas
- Muchas pruebas se pueden hacer 1 sola vez

ANONIMATO





i Lotería!



XSS

Cross Site Scripting:

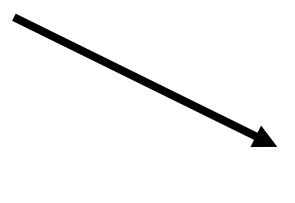
Es un vector de ataque que intenta inyectar código malicioso en nuestros sitios web.

Normalmente se produce por malas prácticas de programación o desconocimiento del vector de ataque.



XSS





	Search:	<h1>H</h1>	acked			
Results for:						
Hacked						
No results were found.						

XSS

- Nunca confiemos en la Data que escribe el usuario.
- Escapemos los caracteres, ejemplo "<" → "<"
- Nunca renderizemos la data del usuario directamente

```
echo "Results for" . $_GET['search'];
echo "Results for" . Htmlspecialchars($_GET['search']);
```



SQL Injection

SQL Injection:

Es un vector de ataque que intenta inyectar código SQL en nuestras consultas del servidor hacia la Base de Datos para obtener más información de la que se le permite o destruir nuestros datos

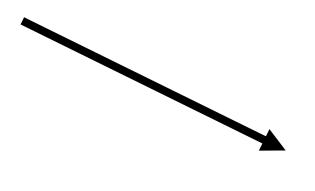
Normalmente se produce por malas prácticas de programación o desconocimiento del vector de ataque.





SQL Injection

Enter your last name: Smith Go!								
SELECT	SELECT * FROM user_data WHERE last_name = ?							
USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT		
102	John	Smith	2435600002222	МС		0		
102	John	Smith	4352209902222	AMEX		0		



Enter your last name: Smith' OR 1=1 -- Go!

SELECT * FROM user_data WHERE last_name = 'Smith' OR 1=1 --'

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COU
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	МС		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0

V

Vulnerabilidades Comunes

SQL Injection

- Nunca confiemos en la Data que escribe el usuario.
- Sanitizemos nuestros campos
- Usemos parámetros de SQL en lugar de concatenar la data en un solo String.

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

Web Services

Autenticación

Estamos en la era de los Servicios WEB, ya sean SOAP o REST

Si no somos cuidadosos con nuestros servicios podemos exponer más datos de los deseados, por lo que debemos de manejar protocolos de autenticación y autorización.

No debemos publicar o exponer a internet más de la cuenta



Web Services

Enumeración

Los ataques de enumeración se pueden dar de distintas formas y con distinto objetivo.

En cualquiera de los casos, el atacante desea obtener un listado de recursos, usuarios o datos en general apoyándose ya sea en mensajes de error muy descriptivos o Web Services sin la verificación adecuada.



Web Services

Enumeración de Usuarios

Uno de los errores más comunes es dar más información de la cuenta en los mensajes de error, por ejemplo en login o registro, que el atacante sepa cuales cuentas ya existen, o son invalidas, etc.

					Password I	Reset
Username:	rapid7	That user doe	s not exist.		Username:	rapid7
Password:	•••••				Submit	
Submit					That userna	ame does not exist.
		Username:	admin			
		Password:		The password is incorrect.		
		Submit				

Web Services

Enumeración de Usuarios

ASHLEY MADISON HACKING: WHAT
HAPPENED WHEN MARRIED MAN WAS
EXPOSED

The man never thought he'd be blackmailed just a couple of weeks after signing up to the site

Web Services

Enumeración de Recursos y directorios

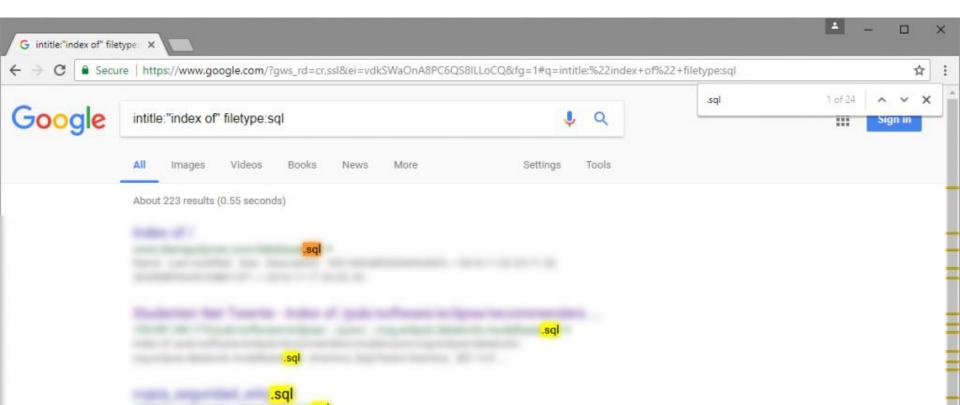
Esto se da por malas configuraciones en nuestros servidores, que permiten al atacante hacer un escaneo de nuestro servidor encontrando directorios y recursos (posiblemente vulnerables), y es uno de los ataques más exitosos utilizando Google Hacking

Index of /

<u>Name</u>	Last modified	Size Description
secret/	2017-01-27 15:40	1
priv/	2017-01-27 15:41	-
edit/	2017-01-27 15:40	-
dir1/	2017-01-27 15:40	-
config.ph	2 2017-01-27 15:40	11K

Web Services

Enumeración de Recursos y directorios



Web Services

Enumeración de datos:

Esta se da cuando nuestros servicios no ejecutan las validaciones adecuadas en las consultas o modificaciones a nuestra base de datos.

La forma común de los Web services con esta vulnerabilidad es:

http://server.com/user/{id}

http://server.com/documents/{id}



Frontend Validations

- Recordemos que nuestro código del Frontend está en manos del cliente... y un posible atacante.
- Validaciones de autenticación y autorización siempre deben de complementarse del lado del servidor.



Username : admin Password : admin

301

302

303

304 305

306 307

308

309

310

311 312 313

314 315

316

317 318

319

320

321

Vulnerabilidades Comunes

Frontend Validations

```
babelHelpers.classCallCheck(this,
    _this11 = babelHelpers.possibleConstructorReturn(this, babelHelpers.getPrototypeOf(AeiView).call(this))
    this11.admin = !1;
    return this11
                                  [{
babelHelpers.createClass
    key: "render",
    value: function render() {
        return (0,
        fiusacApp.html$1) ( templateObject8 d6f246f0211611e996bf4db44374c7b8(
                                                                                  , this.admin ? (0,
        _fiusacApp.html$1)(_templateObject9_d6f246f0211611e996bf4db44374c7b8()
}, {
    key: "firstUpdated",
        if ("/payment" == window.app.queryParams.path)
            this.admin = "1819" == firebase.auth().currentUser.uic
}], [{
    key: "properties",
    get: function get() {
        ceturn &
```



Frontend Validations

```
iew):

▼ Watch
ctorReturn(this, babelHelpers.getPrototypeOf
                                                          call(this))
                                                                           this.admin = true: true
                                                                           window.app.queryParams.path = '/payment': "/payment"
                                                                           firebase.auth().currentUser.uid = "1819": "1819"
                                                                          ▼ Call Stack
                                                                                                     Not paused
ect8_d6f246f0211611e996bf4db44374c7b8( , this.admin ? (0,
9 d6f246f0211611e996bf4db44374c7b8())
                                                                          ▼ Scope
                                                                                                    Not paused
                                                                          ▼ Breakpoints
roperties) {
Params.path)
                                                                                              cache=f757d794c59f6197ed201d2132ed7478:f...
e.auth().currentUser.uid
                                                                                               templateObject8_d6f246f0211611e996bf4...

    XHR/fetch Breakpoints

    DOM Breakpoints
```

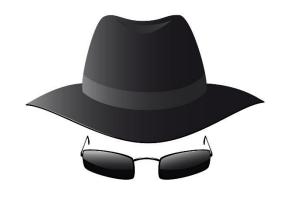
Bonus



WebGoat

https://github.com/WebGoat





HACKING 101

¿Preguntas?







i Muchas Gracias!



Jorge Alejandro Cajas Mota

jac.mota@gmail.com @cajasmota https://github.com/Motojo/Slides