Trabajo Práctico Evaluatorio - Segundo Parcial - Introducción a la Ciberseguridad - TDS17

Alumno: CRISTIAN COLLINS

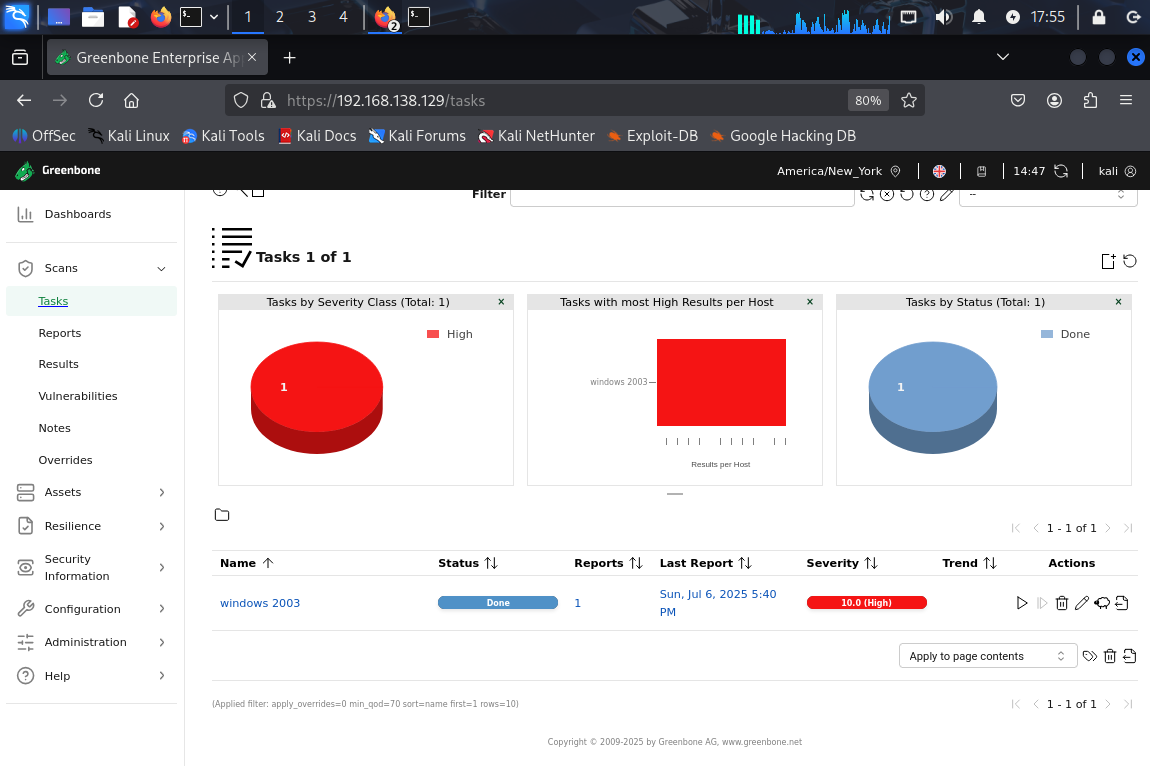
Matrícula: 600061

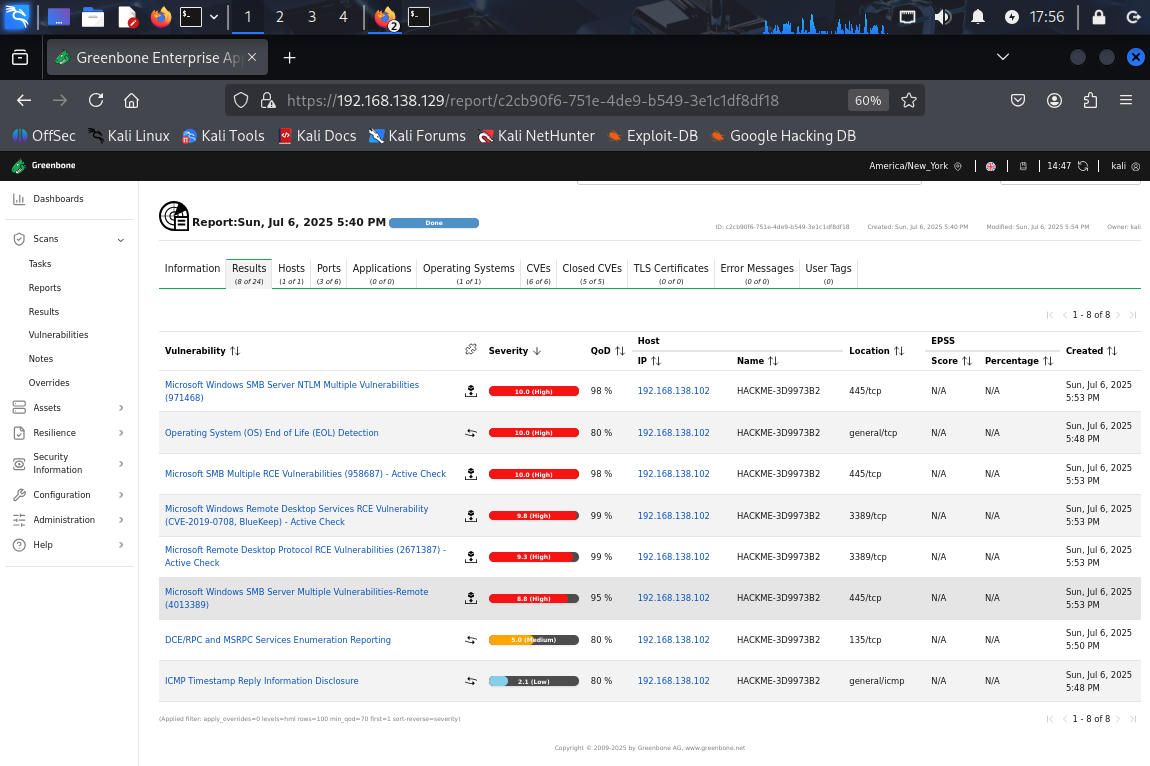
# Explotación en Windows XP/2003 con Greenbone + Metasploit

Windows 2003

Evidencias:

- Captura del informe de Greenbone

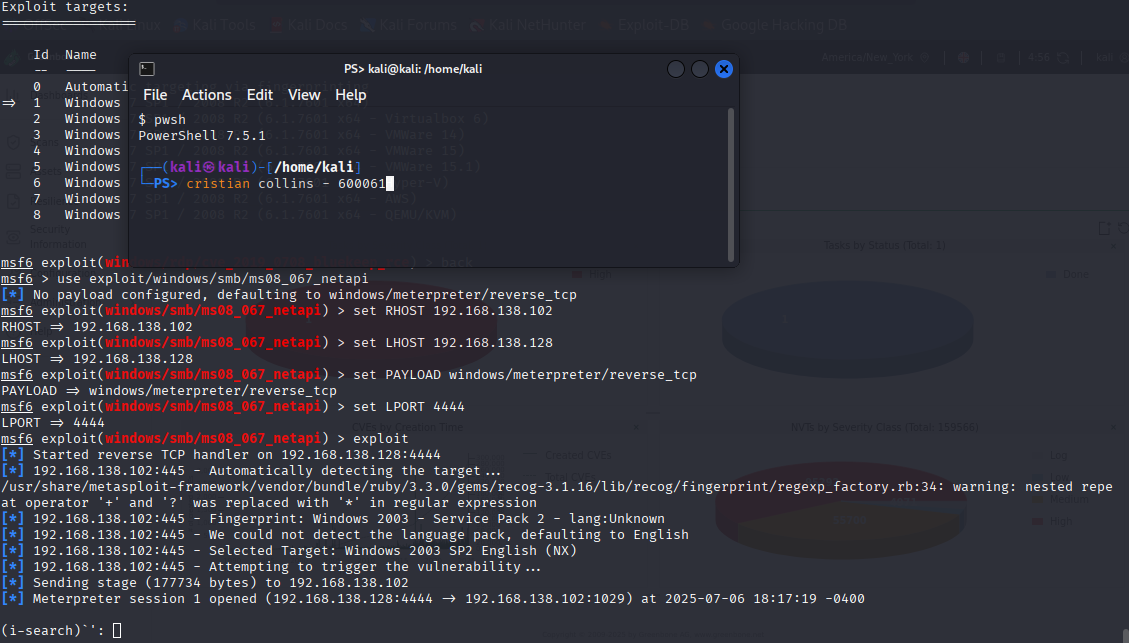


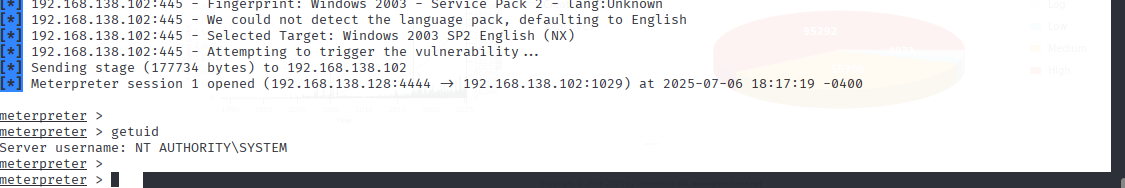


| **Vulnerabilidad** | **Severidad** | **Puerto** | **CVE / Referencia** |
| --- | --- | --- | --- |
| Microsoft Windows SMB Server NTLM Multiple Vulnerabilities | Alta (10) | 445/tcp | CVE-... (varias relacionadas) |
| Operating System (OS) End of Life (EOL) Detection | Alta (10) | general | - |
| Microsoft SMB Multiple RCE Vulnerabilities (958687) | Alta (10) | 445/tcp | CVE-... |
| Microsoft Windows Remote Desktop Services RCE (CVE-2019-0708) | Alta (9.8) | 3389/tcp | CVE-2019-0708 (BlueKeep) |
| Microsoft Remote Desktop Protocol RCE Vulnerabilities (2671387) | Alta (9.3) | 3389/tcp | CVE-... |
| Microsoft Windows SMB Server Multiple Vulnerabilities-Remote | Alta (8.8) | 445/tcp | CVE-... |
| DCE/RPC and MSRPC Services Enumeration Reporting | Media (5) | 135/tcp | - |
| ICMP Timestamp Reply Information Disclosure | Baja (2.1) | general | - |

| **Sistema Operativo** | **CVE** | **CWE** | **Descripción** | **Exploit** |
| --- | --- | --- | --- | --- |
| Windows Server 2003 | CVE-2019-0708 | CWE-284 | BlueKeep: Vulnerabilidad crítica RCE en RDP que permite ejecución remota sin autenticación | exploit/windows/rdp/cve\_2019\_0708\_bluekeep |
| Windows Server 2003 | CVE-2008-4250 | CWE-119 | MS08-067: Vulnerabilidad crítica en SMB que permite ejecución remota sin autenticación | exploit/windows/smb/ms08\_067\_netapi |
| Windows Server 2003 | CVE-2017-0144 | CWE-284 | EternalBlue: Vulnerabilidad en SMBv1 que permite ejecución remota (WannaCry) | exploit/windows/smb/ms17\_010\_eternalblue |
| Windows Server 2003 | CVE-2010-2729 | N/A | SMB Remote Code Execution mediante vulnerabilidades múltiples NTLM | exploit/windows/smb/smb\_relay |
| Windows Server 2003 | N/A | N/A | Fin de soporte (EOL) del sistema operativo; sin parches de seguridad disponibles | No aplica (mitigación: migración) |

- Captura de sesión en Metasploit





| **Sistema Operativo** | **CVE** | **Descripción** | **Exploit usado** |
| --- | --- | --- | --- |
| Windows Server 2003 SP2 | CVE-2008-4250 | Desbordamiento de búfer en el servicio SMB que permite ejecución remota sin autenticación. | exploit/windows/smb/ms08\_067\_netapi |

**Análisis breve de la vulnerabilidad, explotación y remediación**

**Vulnerabilidad:**

MS08-067 (CVE-2008-4250), asociada a CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer), es una vulnerabilidad crítica de desbordamiento de búfer en el servicio Server de Windows. Permite la ejecución remota de código sin autenticación mediante una petición SMB especialmente diseñada. Afecta sistemas como Windows XP y Windows Server 2003 SP2 sin parches.

**Explotación:**

Usamos el módulo exploit/windows/smb/ms08\_067\_netapi de Metasploit Framework desde Kali Linux.  
Tras configurar los parámetros correctos (RHOST, LHOST, PAYLOAD, etc.), se ejecutó el exploit y se logró una **sesión Meterpreter** en la máquina víctima con IP 192.168.138.102.  
Desde esa sesión se puede tomar control total del sistema: ejecutar comandos, capturar información, subir archivos o crear persistencia.

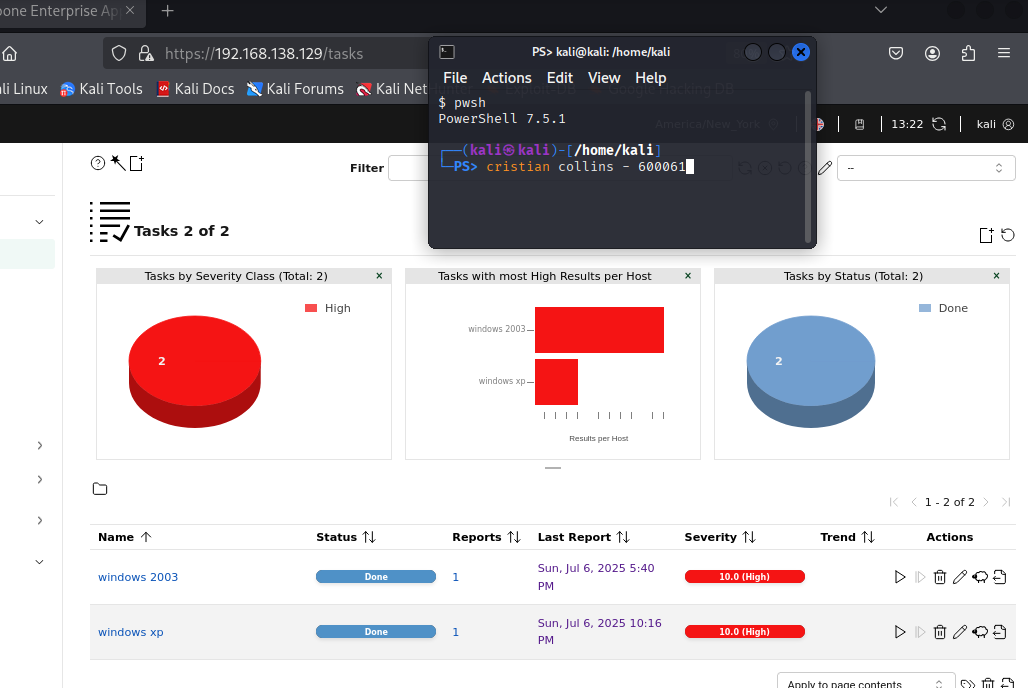
**Medida de remediación:**

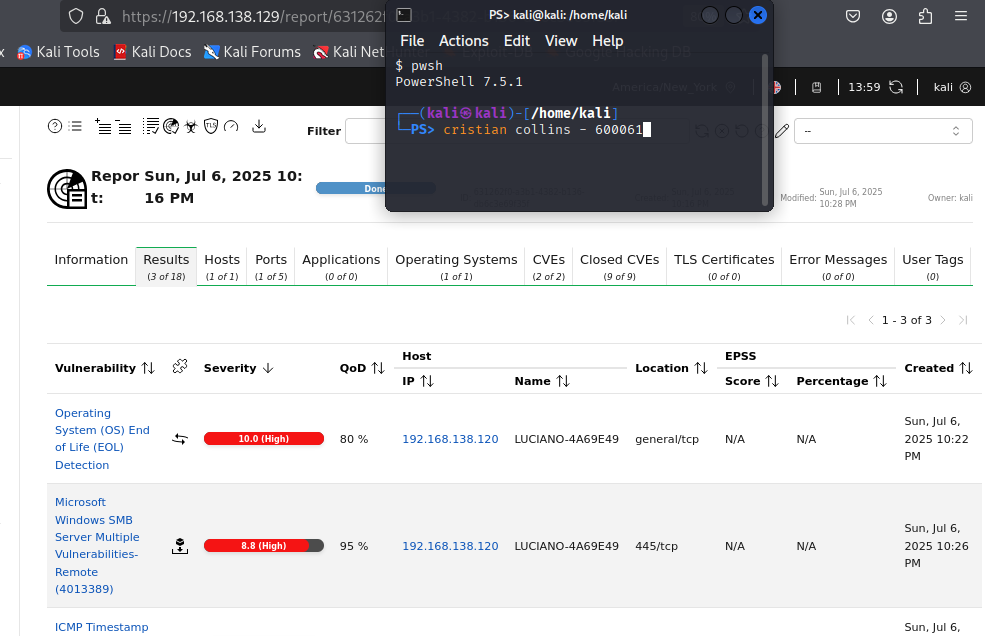
* **Actualizar el sistema operativo**: Windows Server 2003 está fuera de soporte desde 2015. Se recomienda migrar a un sistema operativo actualizado y con soporte vigente (como Windows Server 2016+).
* **Aplicar el parche de seguridad MS08-067** proporcionado por Microsoft para corregir el desbordamiento de búfer.
* **Restringir el acceso al puerto TCP 445** desde redes externas mediante firewall, para prevenir accesos no autorizados al servicio SMB.
* **Deshabilitar SMBv1** si no se necesita (es obsoleto y altamente vulnerable).

Windows xp

Evidencias:

- Captura del informe de Greenbone

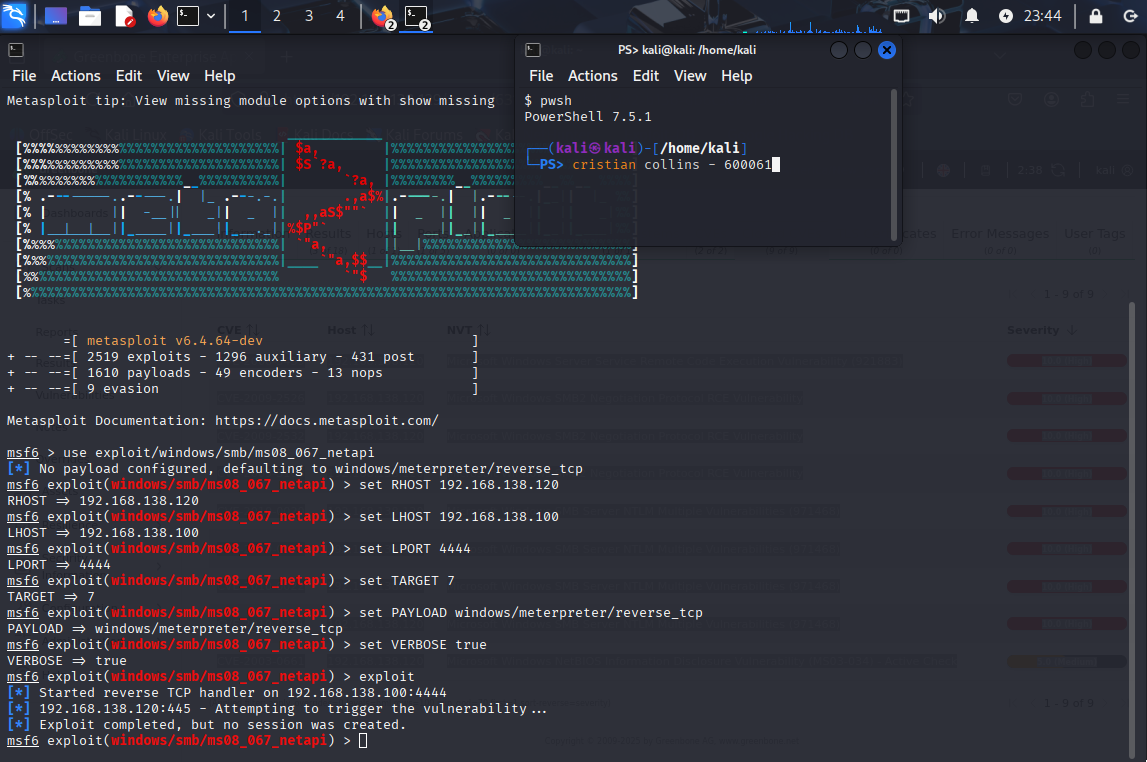




| **Vulnerabilidad** | **Severidad** | **Puerto** | **CVE / Referencia** |
| --- | --- | --- | --- |
| Operating System (OS) End of Life (EOL) Detection | Alta (10) | general/tcp | N/A |
| Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389) | Alta (8.8) | 445/tcp | N/A |
| ICMP Timestamp Reply Information Disclosure | Baja (2.1) | general/icmp | N/A |

| **Sistema Operativo** | **CVE** | **CWE** | **Descripción** | **Exploit (Metasploit)** |
| --- | --- | --- | --- | --- |
| Windows XP | N/A | N/A | Fin de soporte: el sistema operativo está desactualizado, sin parches de seguridad (EOL). | No aplica |
| Windows XP | N/A (MS4013389) | CWE-119 | Múltiples vulnerabilidades en SMB Server que permiten ejecución remota (RCE). | exploit/windows/smb/ms08\_067\_netapi (relacionado) |
| Windows XP | N/A | CWE-200 | Divulgación de información a través de respuestas ICMP Timestamp. | No aplicable a Metasploit |

- Captura de sesión en Metasploit



| **Sistema Operativo** | **CVE** | **Descripción** | **Exploit usado** |
| --- | --- | --- | --- |
| Windows 10 Pro | CVE-2017-0143 | Vulnerabilidad crítica de ejecución remota en el servidor SMBv1 (EternalBlue). | exploit/windows/smb/ms17\_010\_eternalblue |

**Análisis breve de la vulnerabilidad, explotación y remediación**

**Vulnerabilidad:**  
MS17-010 (CVE-2017-0143), asociada a una vulnerabilidad crítica de ejecución remota de código en el protocolo SMBv1 de Microsoft Windows, conocida como EternalBlue. Esta vulnerabilidad permite que un atacante remoto ejecute código arbitrario sin necesidad de autenticación previa, explotando un fallo en el manejo de paquetes SMB. Afecta sistemas Windows que tienen habilitado SMBv1, incluyendo Windows 7, Windows 10 y Windows XP sin parches.

**Explotación:**  
Se utilizó el módulo exploit/windows/smb/ms17\_010\_eternalblue de Metasploit Framework desde Kali Linux.  
Tras configurar los parámetros necesarios (RHOST, LHOST, LPORT, PAYLOAD, TARGET 7 para Windows 10 Pro), se intentó la explotación contra la máquina víctima con IP 192.168.138.120. Sin embargo, el exploit no logró abrir una sesión Meterpreter, indicando que, aunque la vulnerabilidad está presente según el escaneo, la explotación efectiva falló en este intento.

**Medida de remediación:**

* **Actualizar el sistema operativo** y aplicar todas las actualizaciones críticas de seguridad, especialmente el parche MS17-010 que corrige esta vulnerabilidad.
* **Deshabilitar SMBv1 en el sistema**, ya que es un protocolo obsoleto y con múltiples vulnerabilidades conocidas.
* **Restringir el acceso al puerto TCP 445** mediante reglas de firewall para limitar la exposición a la red interna o externa.
* Implementar políticas de seguridad adicionales como monitoreo y detección de actividades sospechosas relacionadas con SMB.

# Campaña de Ingeniería Social con GoPhish

**Objetivo:**

Simular un ataque de phishing orientado a capturar credenciales (CUIT y Clave Fiscal) mediante una campaña personalizada usando la herramienta GoPhish.

Evidencias:

- Captura del mail falso recibido

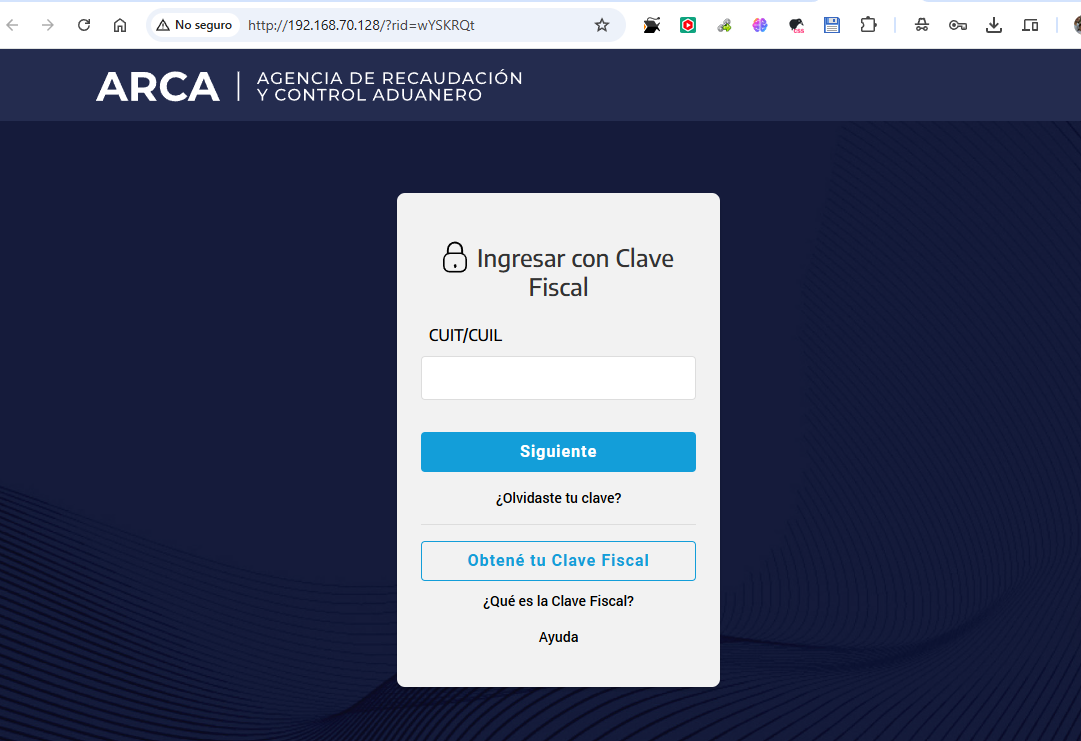
- Captura del sitio falso

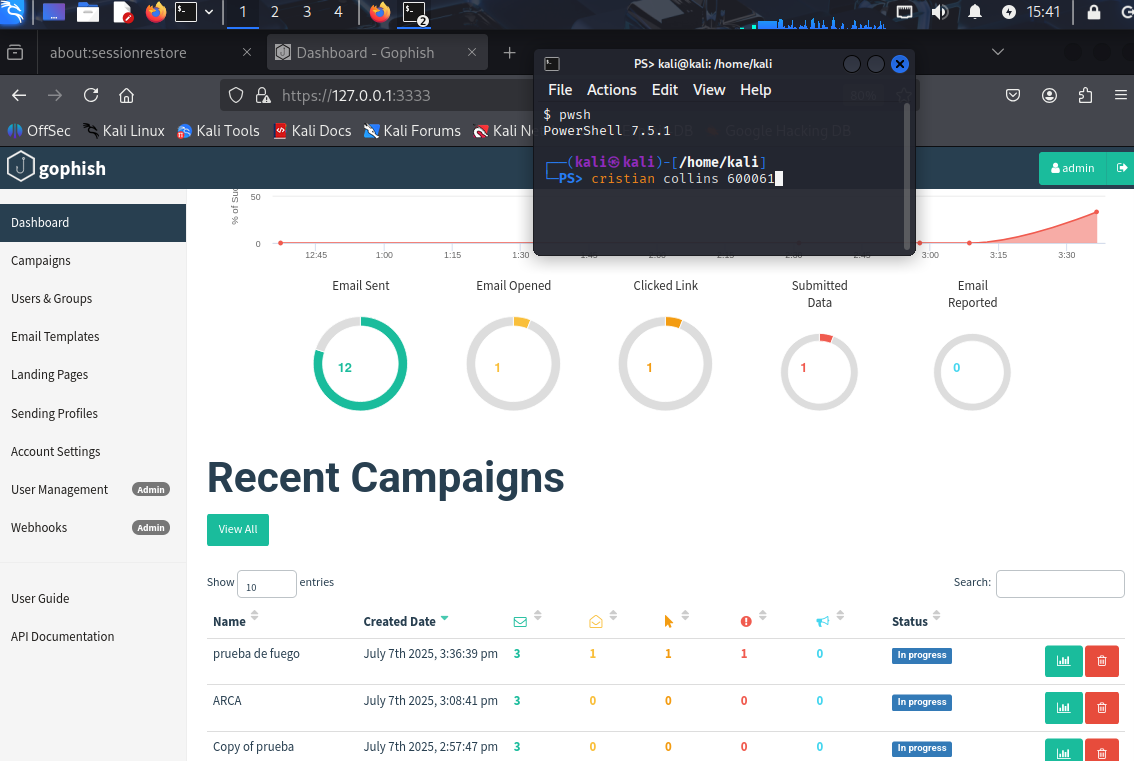
- Captura del dashboard de GoPhish

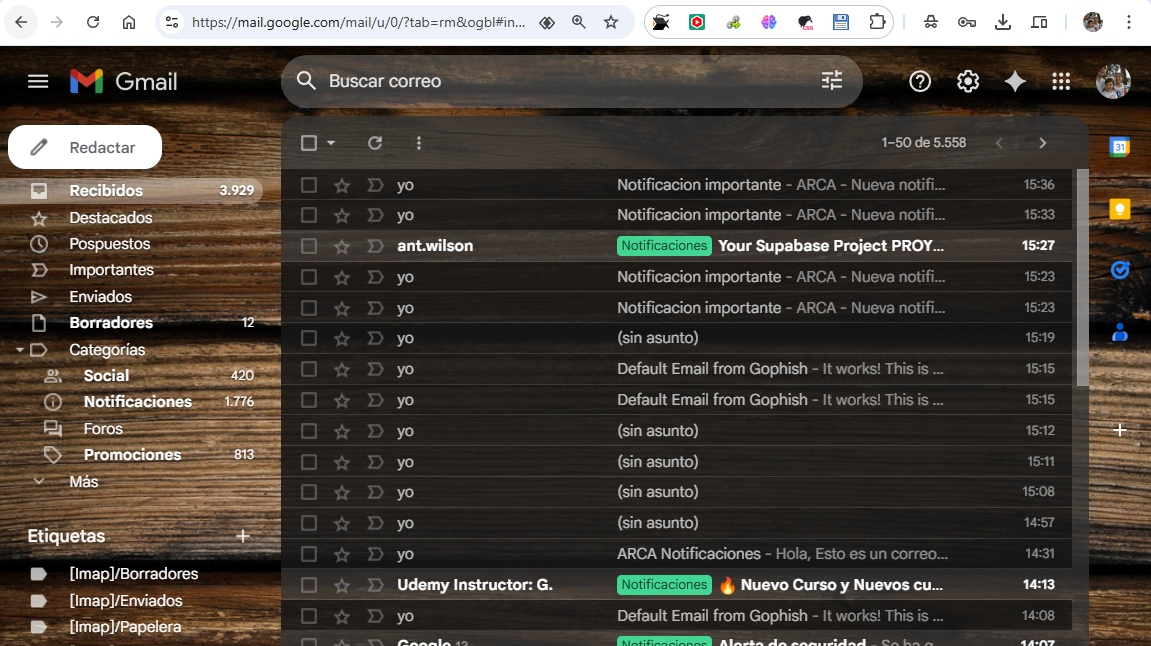
- Envío del mail falso a hvillalbaseg@gmail.com

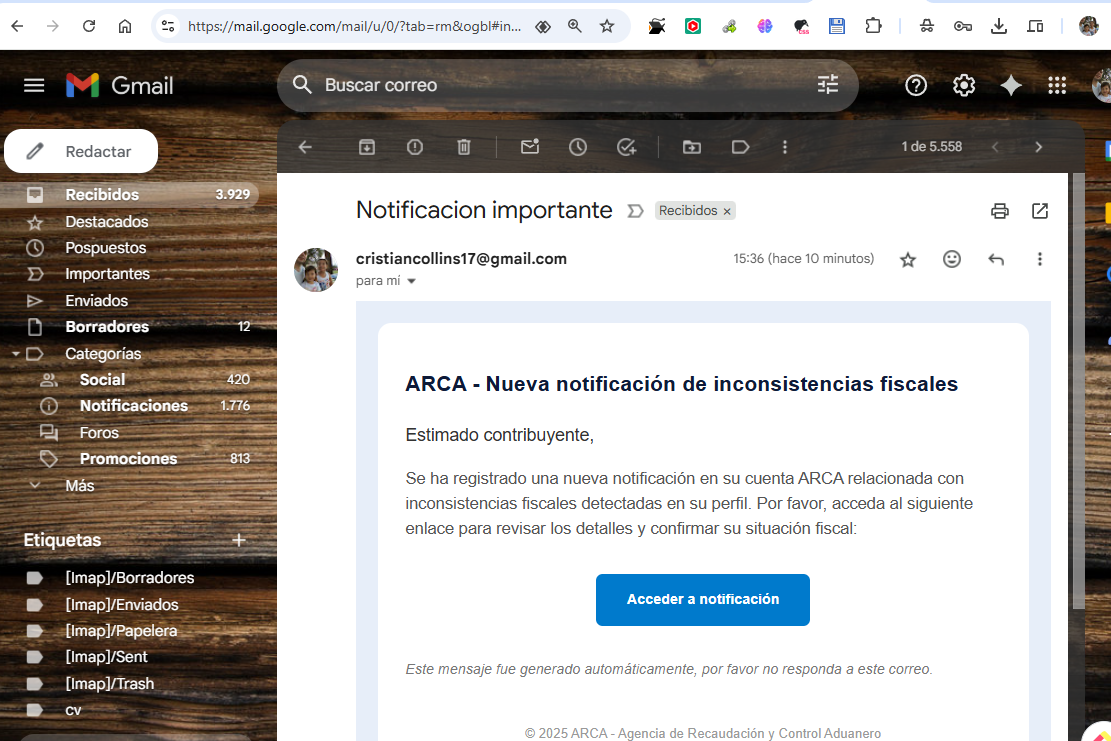
**Análisis técnico y social:**

* **Técnicas utilizadas:**
  + Spoofing de remitente (simulación de cuenta @arca.afip.gob.ar).
  + Clonación visual de landing legítima (AFIP).
  + Inserción de variables GoPhish ({{.URL}}) para rastreo personalizado.
* **Vector de ataque:**
  + Ingeniería social basada en urgencia: “Nueva notificación en ARCA”.
* **Impacto simulado:**
  + Posible obtención de credenciales fiscales.
  + Alto nivel de confianza visual por el uso de logos y estilo institucional.
* **Medidas preventivas sugeridas:**
  + Capacitación al personal.
  + Verificación de dominios y remitentes.
  + Uso de segundo factor de autenticación.









# SQL Injection, XSS y Fuerza Bruta en DVWA

**Explotación de Vulnerabilidades Web en DVWA**

**Objetivo:**  
Ejecutar pruebas de concepto sobre vulnerabilidades conocidas en aplicaciones web utilizando Damn Vulnerable Web Application (DVWA) y demostrar cómo se explotan ataques del tipo **SQL Injection, XSS (Reflected y Stored)** y **Fuerza Bruta**, incluyendo evidencias, payloads utilizados y medidas de remediación.

**Requisitos previos**

* Máquina virtual **Kali Linux** y **DVWA** configurados en red interna.
* **Nivel de seguridad DVWA:** LOW y MEDIUM (dependiendo del desafío).
* Herramientas utilizadas: Firefox, Burp Suite, DVWA, comandos de Kali.

**Pruebas Realizadas**

**3.1. SQL Injection (CWE-89)**

**Descripción:**  
Se explotan formularios vulnerables a inyecciones SQL. Se prueban consultas que permiten obtener información de la base de datos completa.

**Pasos:**

* Acceder al módulo SQL Injection.
* Probar consultas simples (1' OR '1'='1) y avanzadas (UNION SELECT, database()).
* Ejecutar al menos **3 consultas diferentes**.

**Payloads utilizados:**

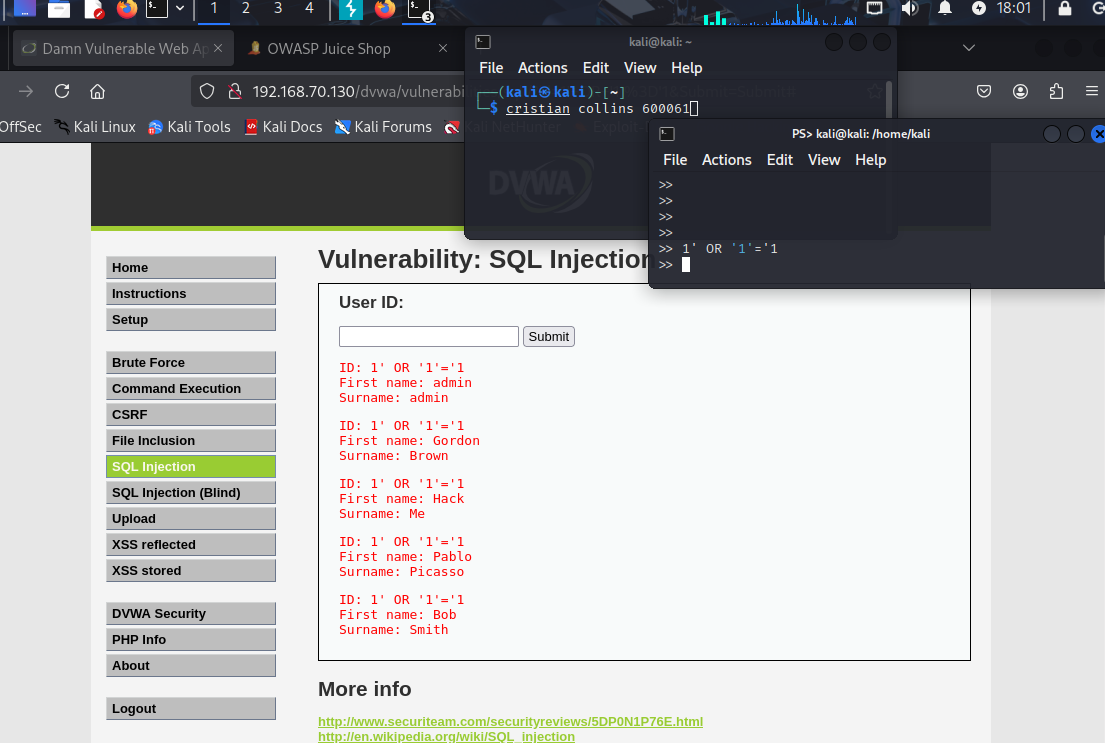
1' OR '1'='1

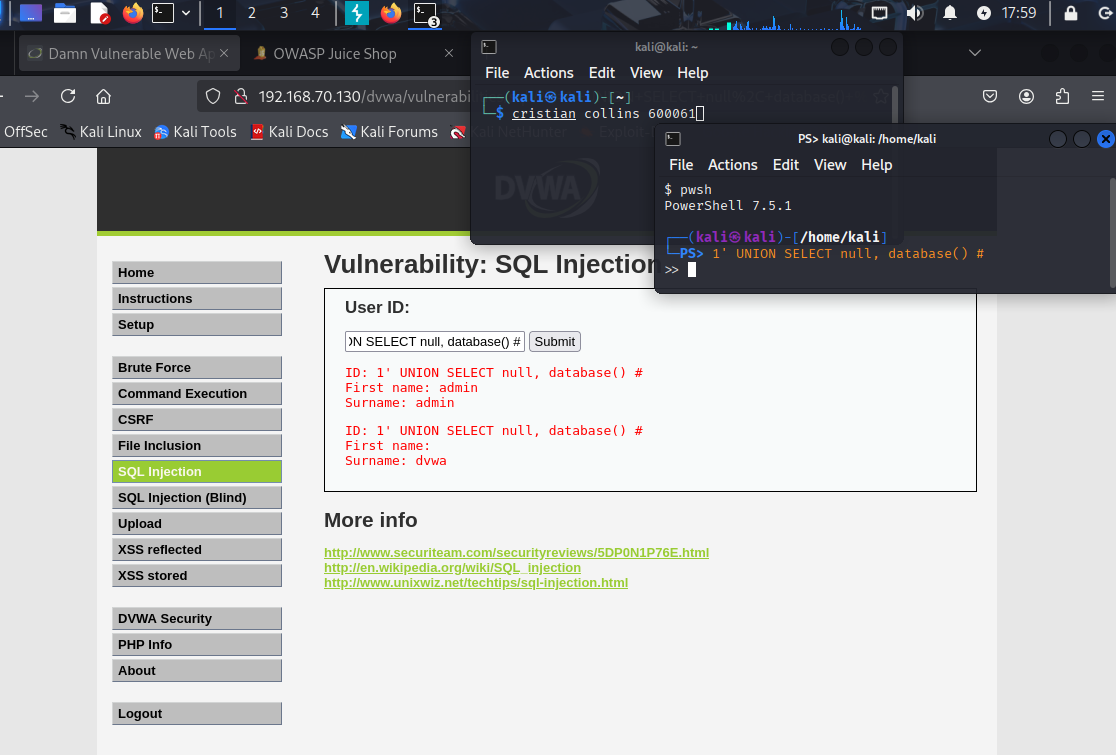
' UNION SELECT null, database() #

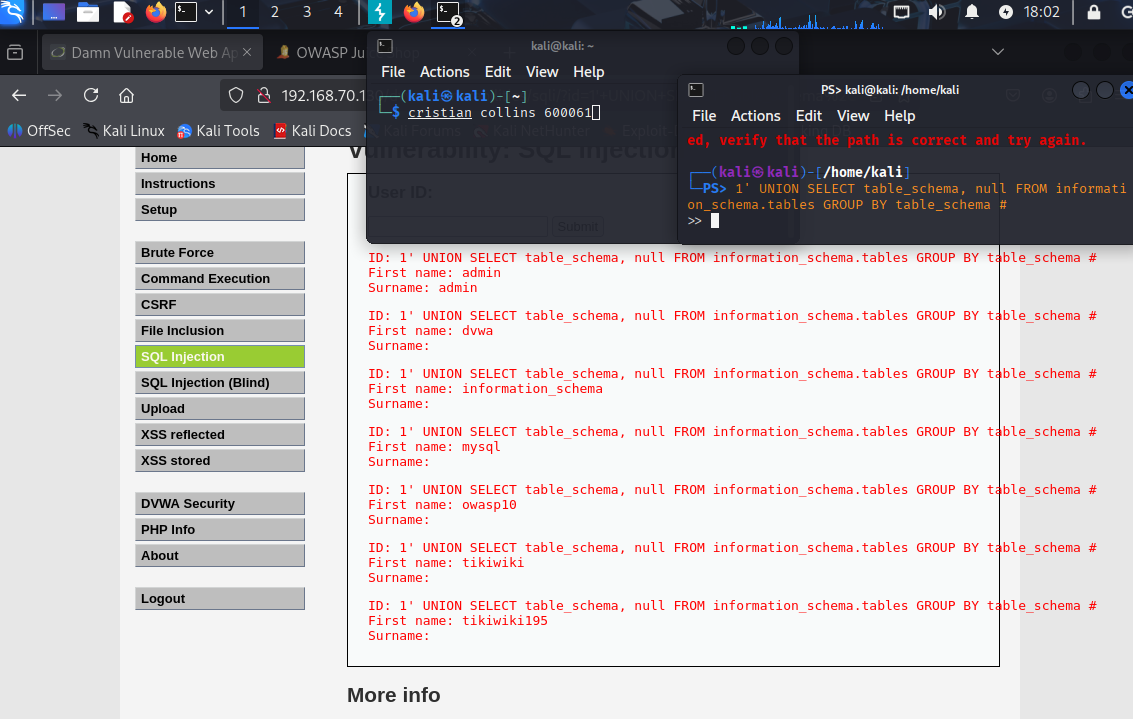
' UNION SELECT table\_name, null FROM information\_schema.tables --

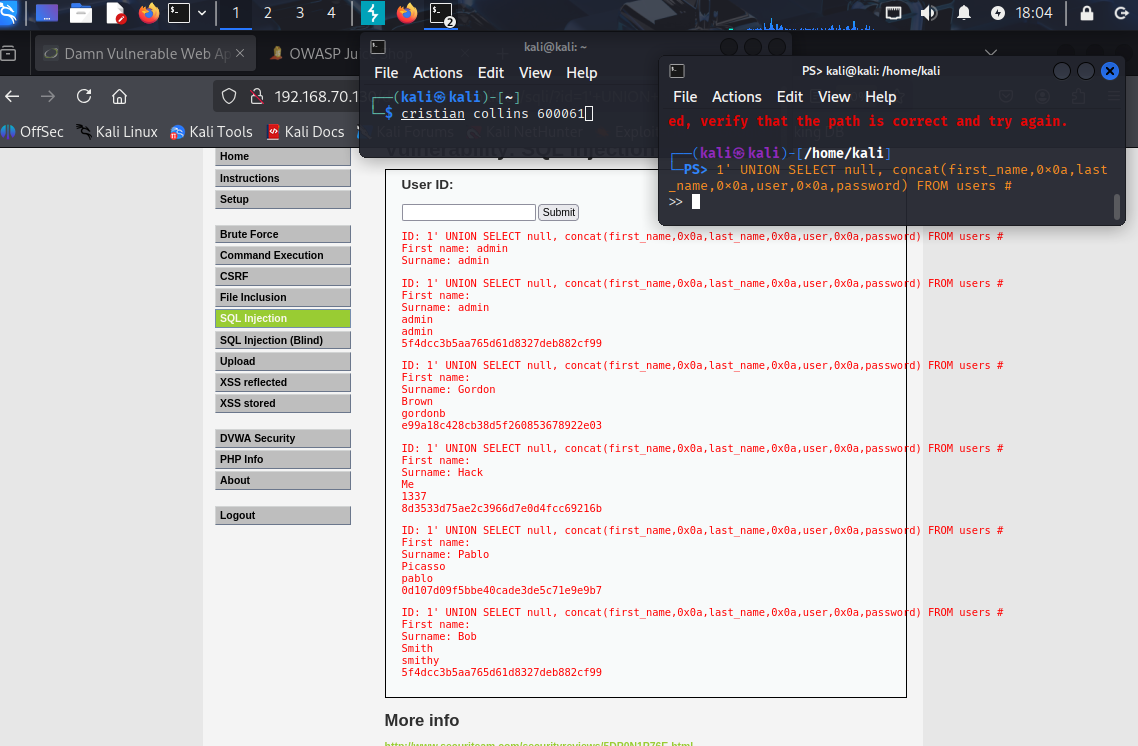
**Evidencias:**

* Capturas de resultados con múltiples usuarios.
* Evidencia de extracción de información sensible.









**Remediación:**

* Validación de entradas.
* Uso de consultas preparadas.
* ORM seguro (parametrización automática).

**Mitigación**

DVWA en nivel Medium o superior:

* Usa **mysqli\_real\_escape\_string()** para limpiar entradas.
* Puede usar consultas preparadas con ? para evitar concatenación directa.

**XSS Reflected y Stored en DVWA (nivel Medium)**

**3.2. Cross-Site Scripting (CWE-79)**

**Tipo:**

* Reflected XSS
* Stored XSS

**Pasos:**

* Reflected: Inyectar script en formularios de saludo o búsqueda.
* Stored: Ingresar JavaScript en formulario de comentarios.

**Payloads utilizados:**

<script>alert("XSS Reflected")</script>

<script>alert(document.cookie)</script>

<script>alert("Alumno: TU NOMBRE - Legajo: 12345")</script>

**Evidencias:**

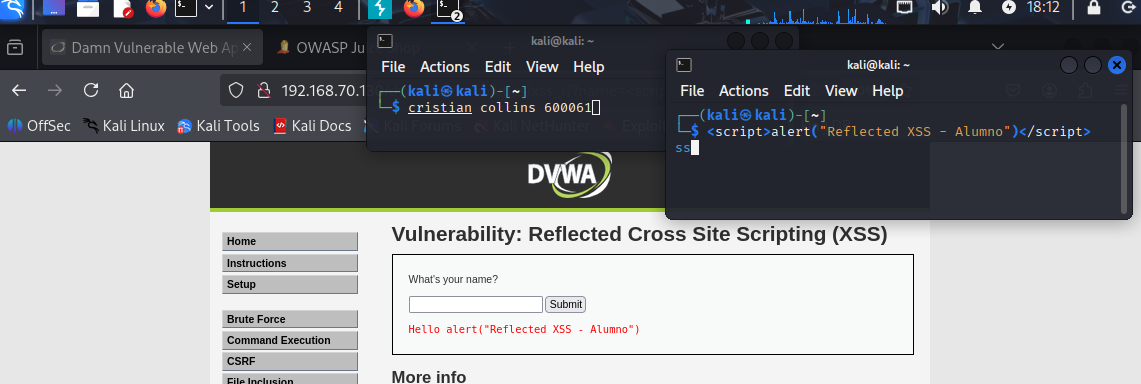
* Captura de alertas ejecutadas.
* Aparece document.cookie en ventana emergente.
* Mensajes almacenados con nombre y matrícula.

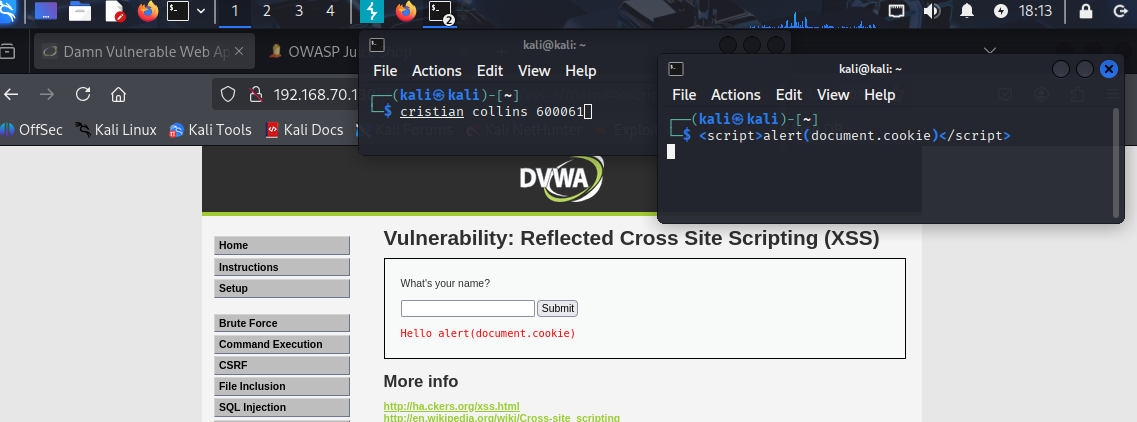
**Remediación:**

* Escapado de caracteres especiales (<, >, ", etc.)
* Validación de entradas en el backend.
* Uso de librerías de sanitización.

**Objetivo:**

Ejecutar un XSS Reflected para mostrar una alerta con la cookie de sesión u otro texto, en nivel Medium.





**Explicación:**

* El campo no sanitiza adecuadamente el input del usuario.
* El script se refleja en la misma página sin ser filtrado ni escapado.
* Esto permite la inyección directa de JavaScript, ejecutado en el navegador.

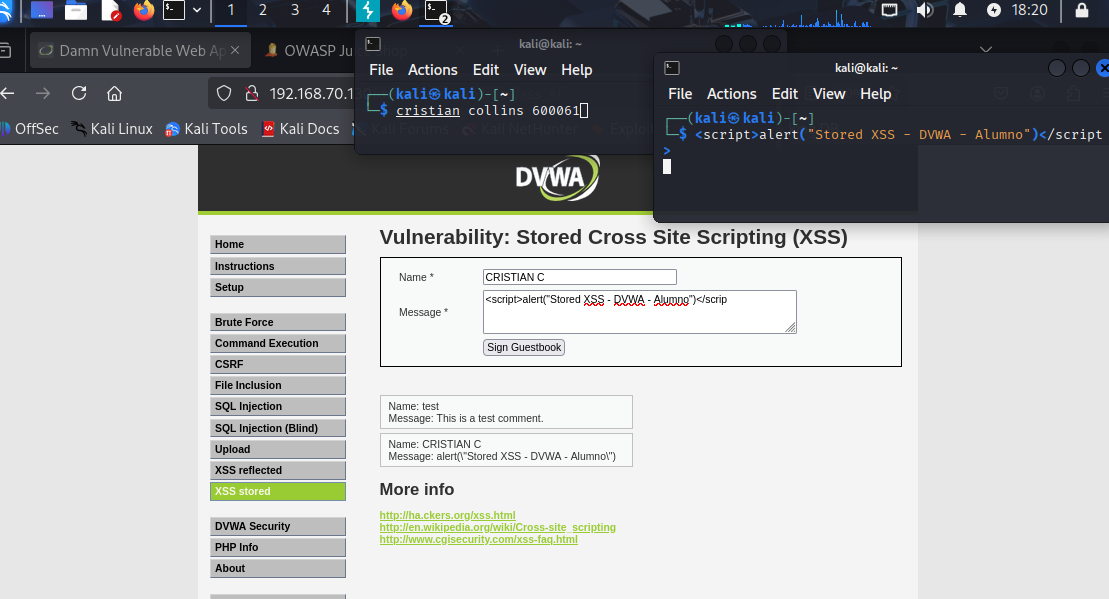
**Medida de remediación:**

* Usar **htmlspecialchars() o htmlentities()** en el servidor para escapar caracteres especiales.
* Implementar una **lista blanca** de entradas válidas.
* Evitar reflejar directamente cualquier input en el HTML.

**XSS Stored**

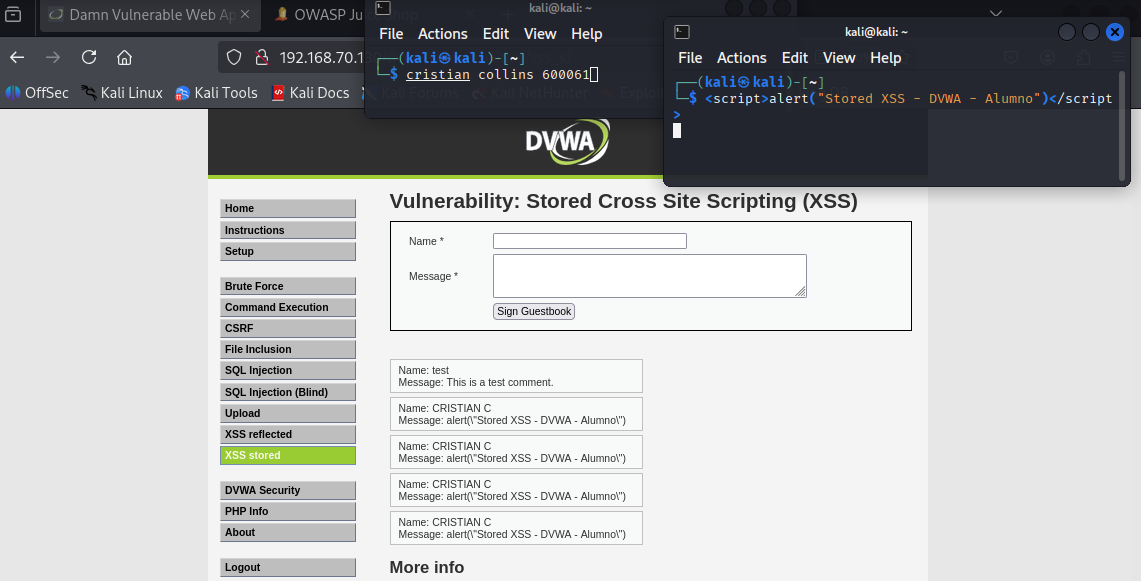
Objetivo:

Guardar un script malicioso que se ejecute cada vez que alguien vea el contenido.



**Explicación:**

* El input del formulario **se guarda en la base de datos sin sanitización**.
* Cada vez que la página se renderiza, el navegador ejecuta los scripts que quedaron almacenados



**Medida de remediación:**

* Usar sanitización antes de guardar en la base.
* Escape de HTML en la visualización.
* Implementar una política de **Content Security Policy (CSP)** para prevenir ejecución de scripts externos.
  1. **Fuerza Bruta (CWE-287)**

**Descripción:  
Se intenta acceder a cuentas de usuario mediante el uso automatizado de múltiples combinaciones de nombre de usuario y contraseña, aprovechando la falta de mecanismos de bloqueo o detección de intentos repetidos.**

**Pasos:**

* Acceder al módulo Brute Force.
* Interceptar la petición con Burp Suite.
* Usar Intruder con lista de usuarios y contraseñas.
* Ejecutar el ataque con **Intruder** y analizar las respuestas.

**Payloads utilizados:**

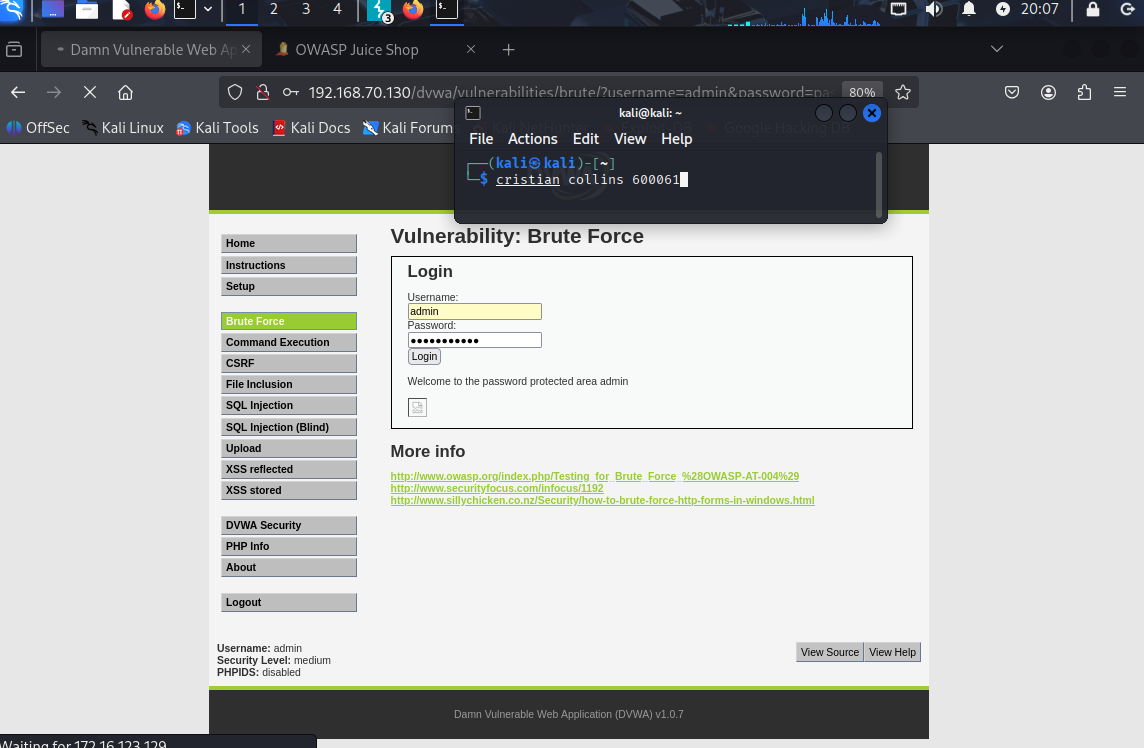
* Diccionario simple con combinaciones como:
* admin
* password
* 123456
* alumno123

**Evidencias:**

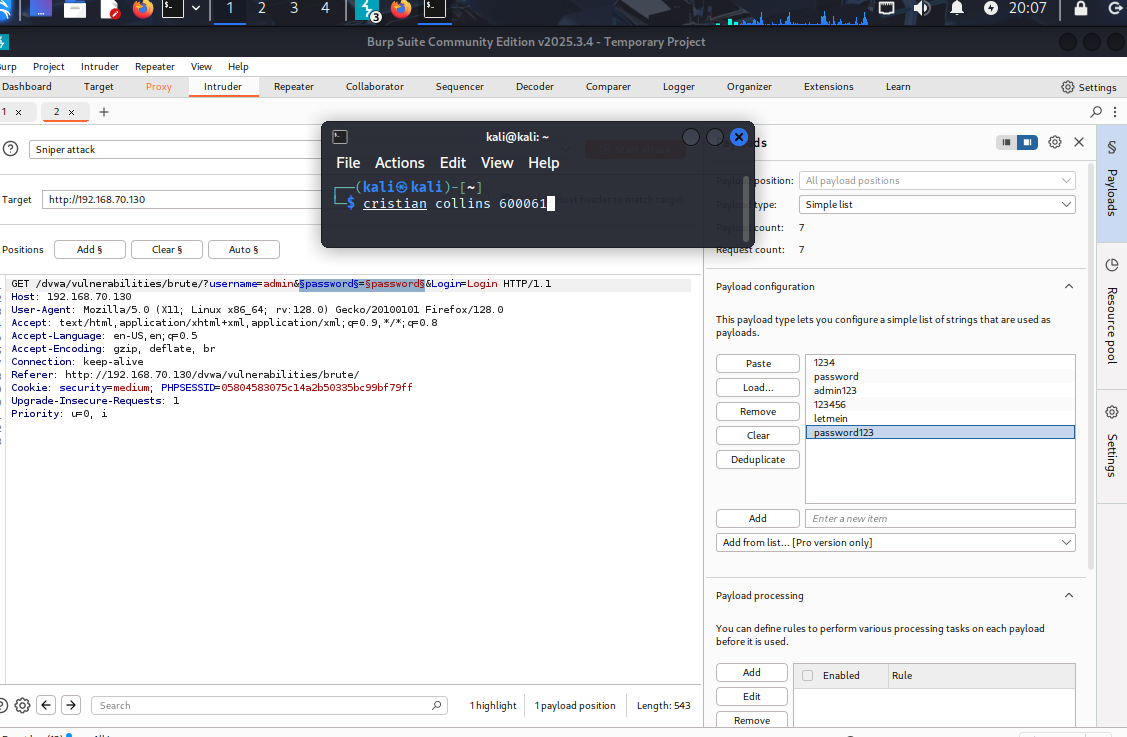
* Captura de pantalla del **login exitoso** mostrando el mensaje de bienvenida de DVWA.
* Petición interceptada y modificada en **Burp Suite**.
* Comparación de respuestas HTTP (código 200 y mensaje de acceso concedido).
* Evidencia de la credencial correcta resaltada.

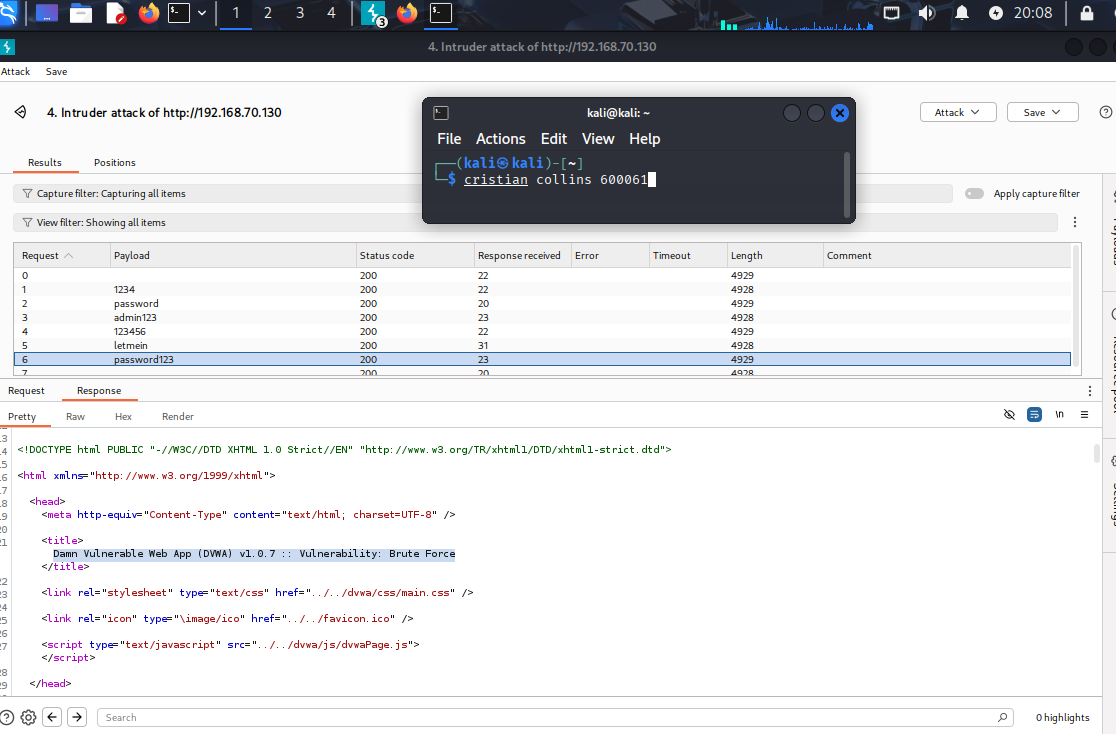
**Remediación:**

* Limitar intentos (lockout).
* Autenticación multifactor.
* Captchas.
* Hashing y almacenamiento seguro de contraseñas.



\*Hubo un error en la carga de la imagen de abajo del login, pero se puede leer la leyenda en la proxima captura (**“Damn Vulnerable Web App”** en el **Response (Pretty)**)





**Resumen Final**

| **Vulnerabilidad** | **Evidencia presentada** | **Payloads utilizados** | **Medidas de mitigación propuestas** |
| --- | --- | --- | --- |
| SQL Injection | Sí | 3 SQL diferentes | Validación, ORM, parametrización |
| XSS (Reflected + Stored) | Sí | Al menos 2 por tipo | Escapado, validación, sanitización |
| Fuerza Bruta | Sí | Diccionario usado | Lockout, MFA, CAPTCHA |

# 4. Explotación en OWASP Juice Shop

Evidencias:

- Captura del Score Board con desafíos completados

- Payloads o técnica utilizada

- Breve análisis de impacto, riesgo y mitigación

**Descripción:**

El formulario de comentarios de Juice Shop no permite seleccionar una calificación de 0 estrellas desde la interfaz. El control del slider y el botón Enviar están diseñados para forzar una calificación mínima de 1 estrella.

**Herramienta utilizada:**

**Burp Suite**, para interceptar y modificar la petición HTTP enviada por el navegador antes de que llegue al servidor.

**Acceso al formulario:**  
Ingresé al formulario de contacto de la tienda y completé un comentario normal, seleccionando 1 estrella (valor mínimo permitido).

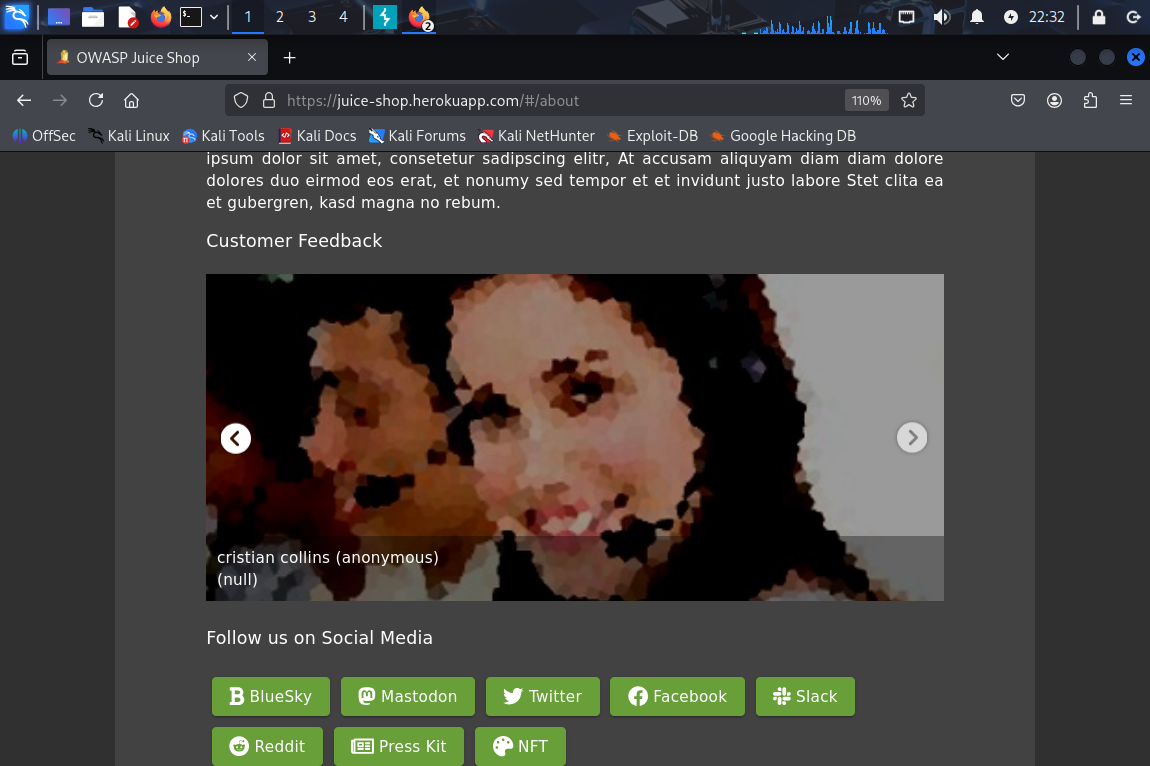
**Interceptar la solicitud:**  
Activé **Intercept is on** en Burp Suite. Luego, al enviar el formulario, Burp capturó la solicitud POST que incluía los datos del comentario.

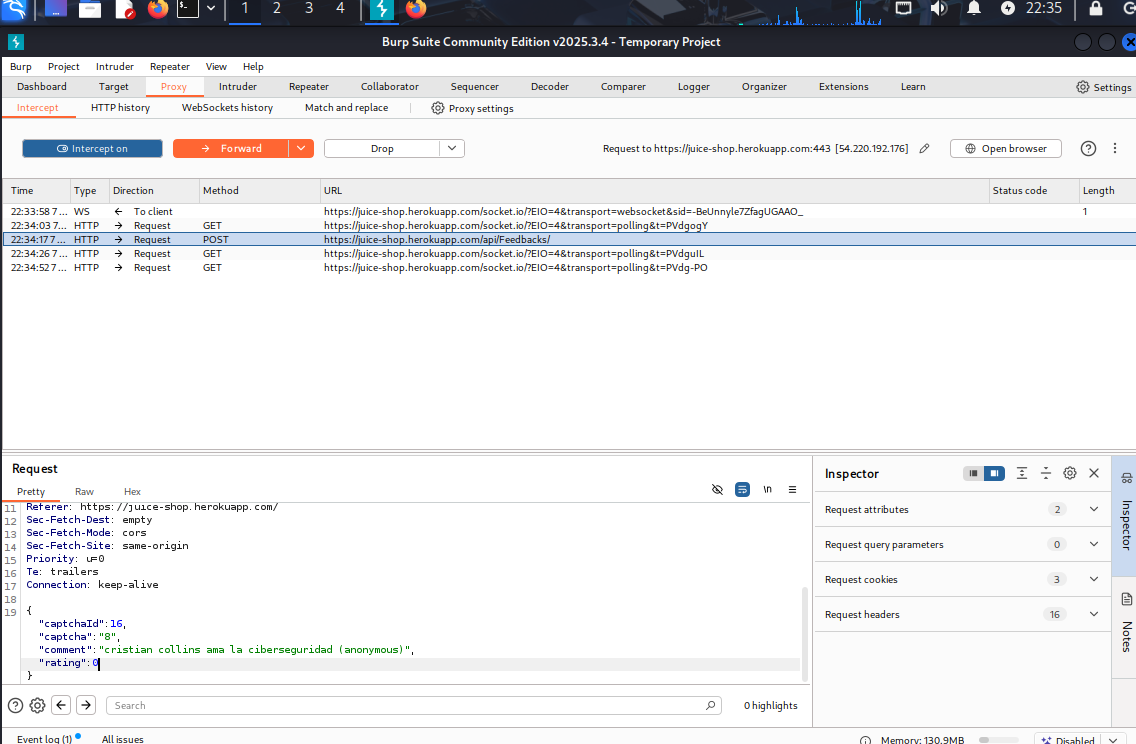
**Editar el valor rating:**  
En la solicitud interceptada busqué el campo:"rating": 1y lo cambié a :rating": 0

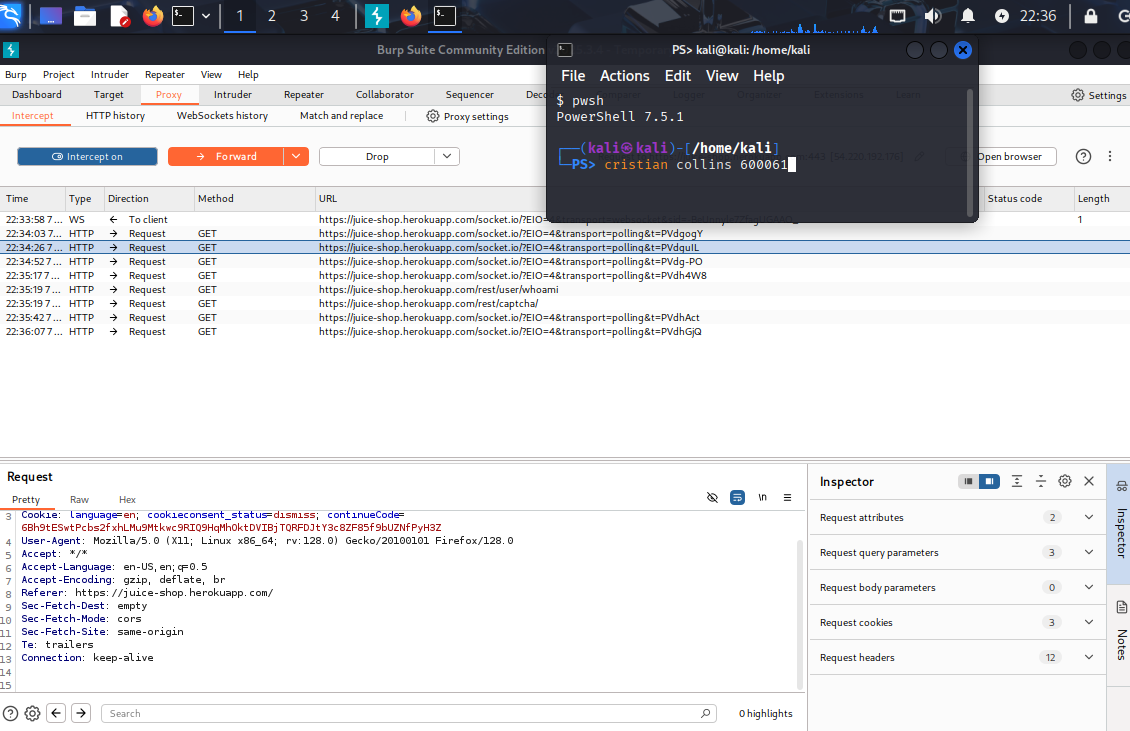
Esto manipula directamente el valor enviado al backend, forzando una calificación no permitida por el frontend.

**Reenviar la solicitud:**  
Presioné **Forward** en Burp Suite para dejar pasar la solicitud modificada hacia el servidor.

**Verificación:**  
En el **Score Board** o sección de logros de Juice Shop, confirmé que el desafío se resolvió correctamente. El comentario quedó guardado con 0 estrellas, algo imposible de lograr desde la interfaz legítima.







**Acceder a la sección de administración**

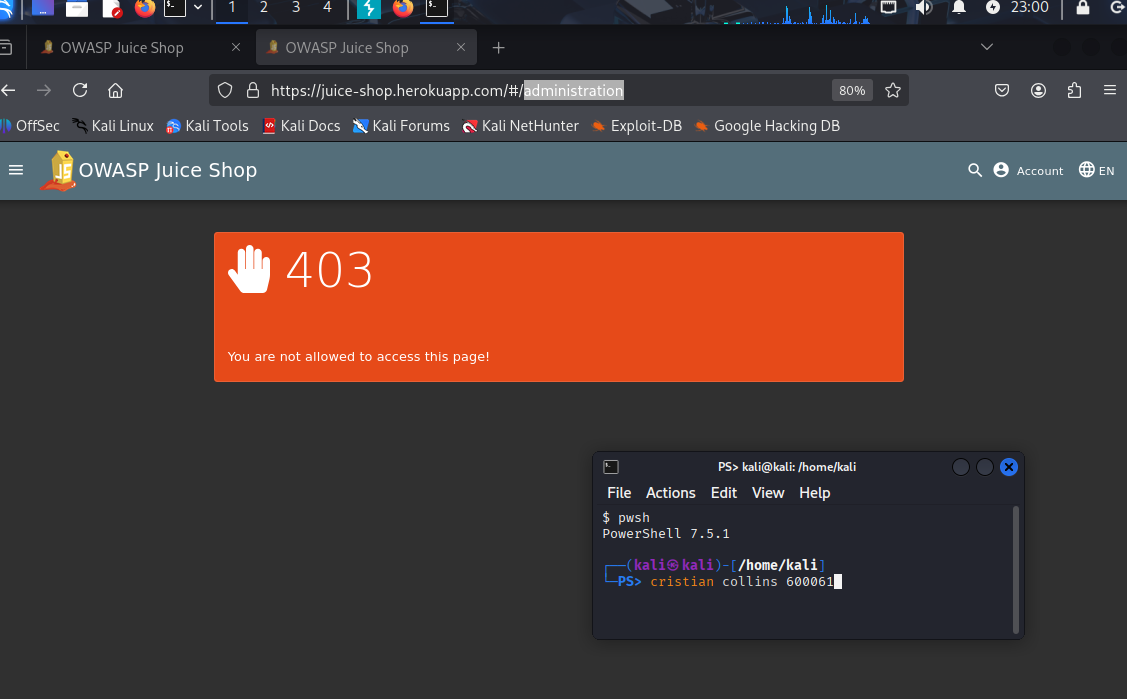
**Descripción:**  
Juice Shop expone una ruta de administración (#/administration) que debería estar protegida para usuarios sin privilegios. Sin embargo, es posible acceder directamente intentando rutas conocidas, sin autenticación adicional. Esto revela información sensible sobre la estructura interna y muestra un mensaje de error 403, confirmando que existe la ruta restringida.

**Herramienta utilizada:**  
Navegador web (Firefox en Kali Linux). No se utilizó Burp Suite porque no se requirió interceptar tráfico, solo modificar manualmente la URL.

**Acceso a la administración:**  
Abrí la aplicación Juice Shop en la ruta normal #/search. Luego, desde la barra de direcciones del navegador, modifiqué manualmente la URL a #/administration y presioné Enter.

**Resultado:**  
La aplicación cargó la página de administración y respondió con un mensaje de acceso denegado (403 Forbidden) junto a un ícono de advertencia. Esto demuestra que la ruta es accesible públicamente a nivel de cliente, sin validación robusta en el backend.

**Verificación:**  
Se verificó el acceso restringido observando la URL modificada (#/administration) y la respuesta de la aplicación mostrando la advertencia 403. Esta evidencia confirma que la ruta está expuesta pero no protegida adecuadamente.



**Medida de remediación:**  
Implementar controles de acceso robustos en el backend para validar los privilegios de cada usuario en cada solicitud. No basta con ocultar o bloquear botones en el frontend; el servidor debe verificar que el usuario esté autenticado y tenga permisos de administrador antes de permitir el acceso a rutas o funcionalidades sensibles. También se recomienda usar controles de ruta protegida y segmentación por roles (RBAC).

**Análisis breve:**  
**Impacto:** Alto. Un atacante podría descubrir rutas sensibles y, si encuentra otras debilidades, escalar privilegios o manipular configuraciones críticas.  
**Riesgo:** Crítico si se combina con otras vulnerabilidades como sesión insegura o bypass de autenticación.  
**Dificultad:** Baja. La prueba se realiza manualmente, sin herramientas avanzadas. El riesgo surge por falta de validación en el servidor.

**Ver la cesta de otro usuario**

**Descripción:**  
La funcionalidad de la tienda permite a un usuario autenticado ver su propia cesta de compras. Sin embargo, es posible modificar manualmente parámetros de la URL o de la solicitud HTTP para intentar acceder a la cesta de otro usuario, exponiendo información privada.

**Herramienta utilizada:**  
Burp Suite, para interceptar y modificar la solicitud GET/POST enviada cuando se consulta la cesta de compras.

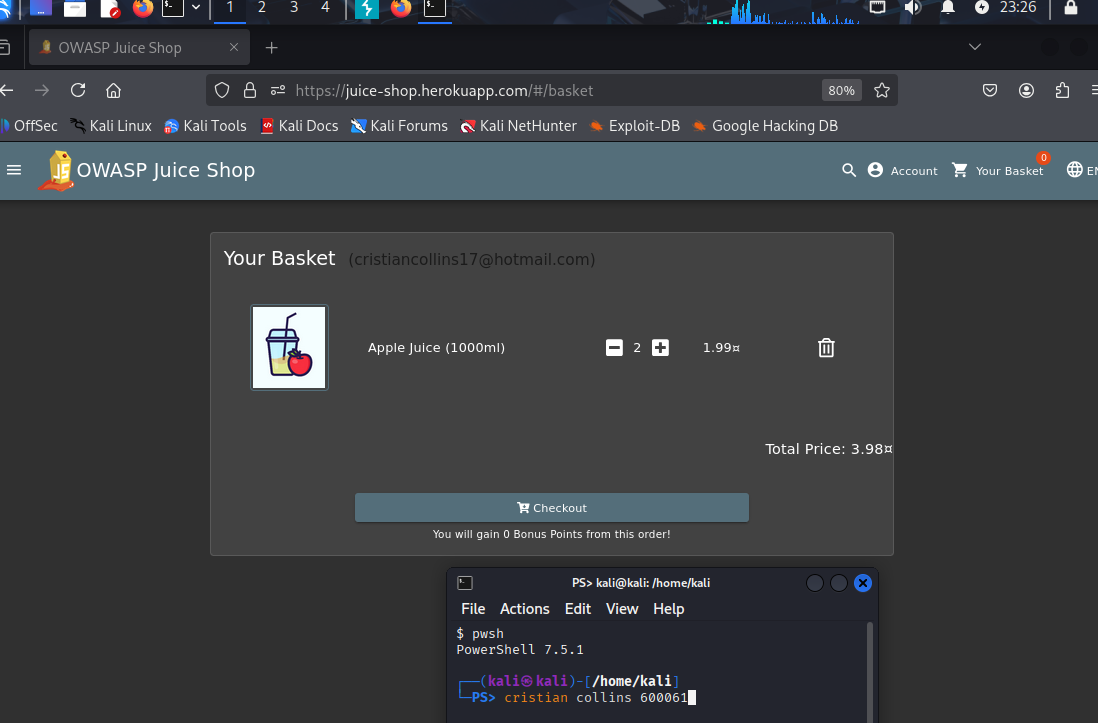
**Acceso a la cesta:**  
Desde la cuenta del usuario, se accede normalmente a la sección de la cesta (cart). Se identifica el parámetro cartId o userId en la URL o el cuerpo de la solicitud.

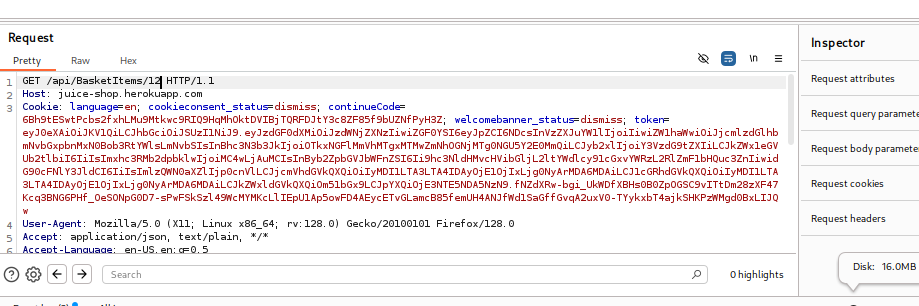
**Interceptar la solicitud:**  
Activé **Intercept is on** en Burp Suite. Al abrir la cesta, capturé la solicitud correspondiente que incluía el ID de usuario o cesta.

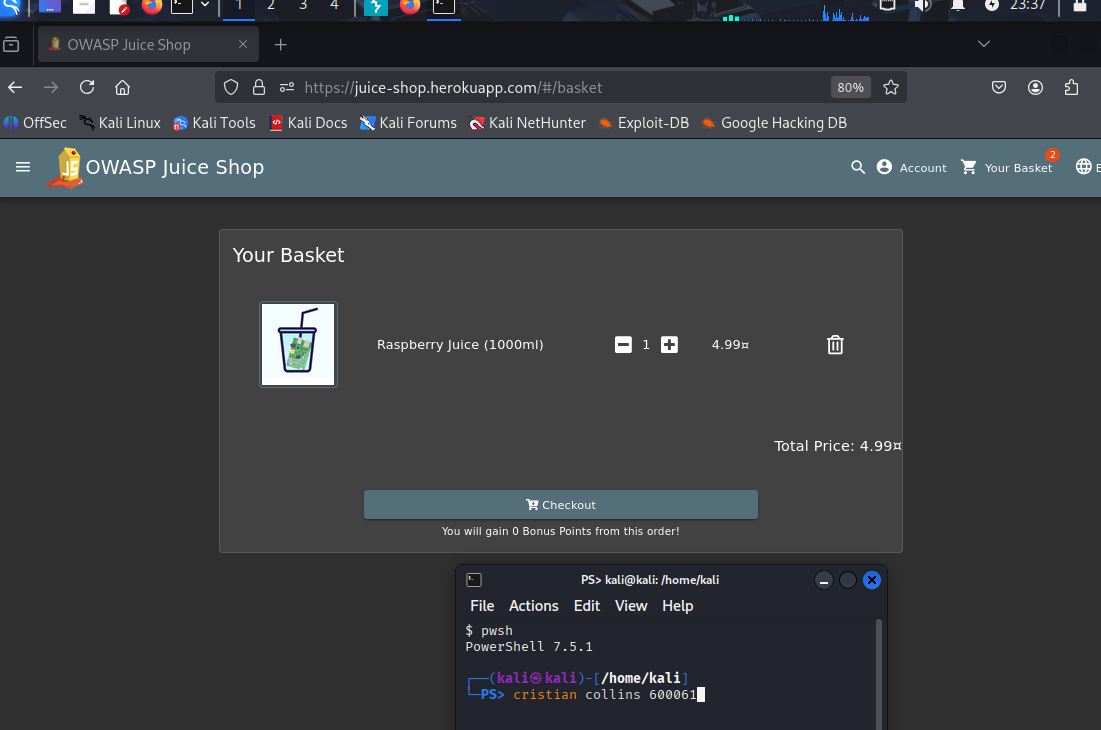
**Editar el parámetro:**  
Modifiqué el valor de cartId o userId en la solicitud interceptada, reemplazándolo por el ID de otro usuario (de 14 a 12).

**Reenviar la solicitud:**  
Presioné **Forward** para enviar la solicitud modificada al servidor.

**Verificación:**  
Si el servidor no valida correctamente la propiedad de la cesta, devuelve los datos de la cesta de otro usuario. Se evidencia con el contenido devuelto (productos, direcciones, datos de pedidos).





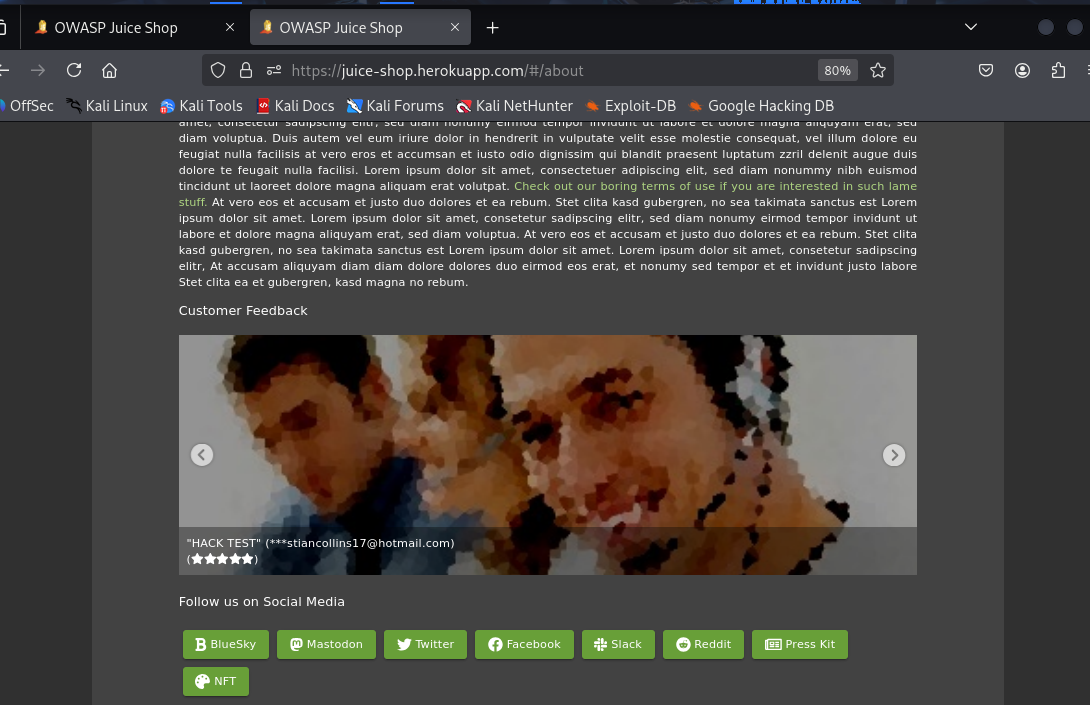


**Medida de remediación:**  
Implementar control de acceso estricto a nivel de servidor, validando que la identidad autenticada coincida con los datos solicitados. Cada recurso sensible debe estar protegido con verificaciones de propiedad y autorización, no solo confiando en parámetros enviados por el cliente.

**Análisis breve:**  
**Impacto:** Alto, porque permite exposición de datos personales y potencial fraude.  
**Riesgo:** Crítico si se combinan con escalación de privilegios.  
**Dificultad:** Media-baja. Requiere interceptar tráfico y conocer cómo modificar IDs.

**Publicar un comentario en nombre de otro usuario**

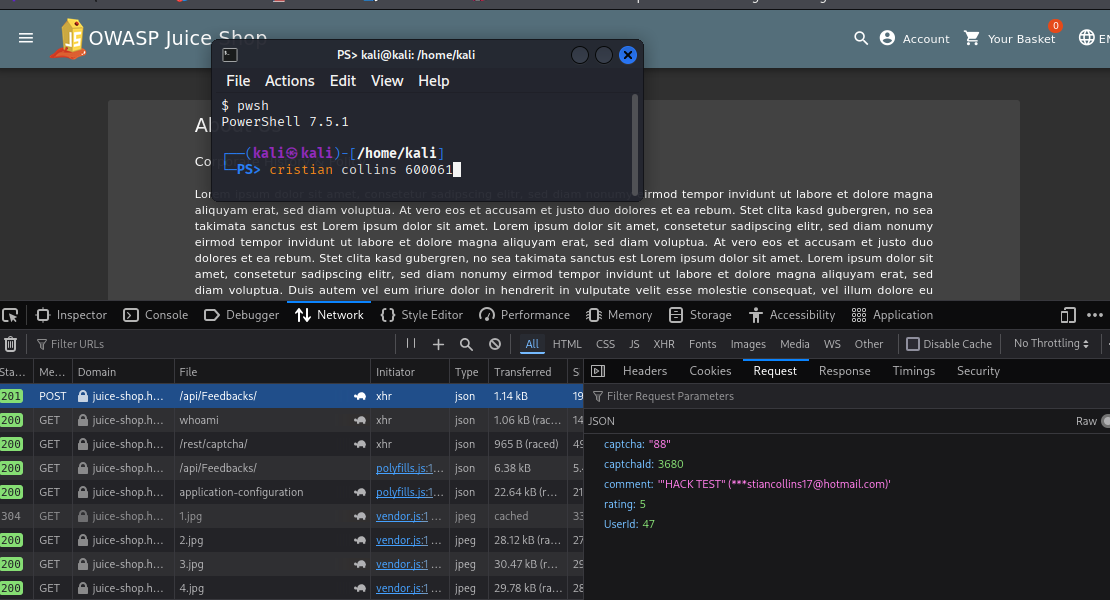
**Descripción del reto:**  
El objetivo era enviar un comentario (feedback) al sistema Juice Shop usando el UserId de otro usuario legítimo, simulando un ataque de *forged feedback* o *comment impersonation*.

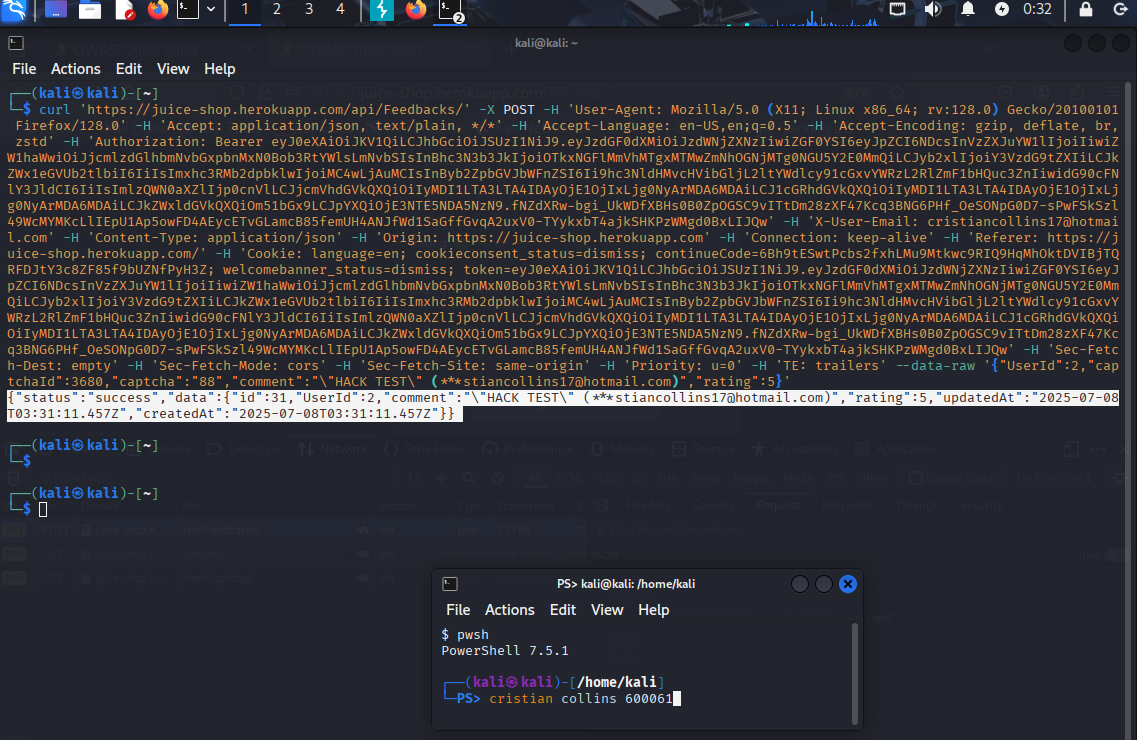


.**Técnica utilizada:**

1. Ingresé a la sección **Contact Us** de Juice Shop.
2. Llené un comentario normal y capturé la petición con las **Herramientas de Desarrollador** del navegador (Network → XHR).
3. Copié el cURL de la petición (Copy as cURL).
4. Modifiqué el JSON de envío para añadir **"UserId": 2**, forzando que el comentario se guardara a nombre de otro usuario.
5. Ejecuté el comando modificado en **Kali** usando la terminal.
6. Confirmé el éxito al recibir la respuesta JSON:

{"status":"success","data":{"id":31,"UserId":2,"comment":"\"HACK TEST\"","rating":5, ...}}





**Análisis:**

* **Impacto:** Permite a un atacante falsificar comentarios públicos, afectando la reputación de usuarios o la integridad del sistema.
* **Riesgo:** Medio-Alto. Un atacante puede realizar desinformación o spam.
* **Dificultad:** Media. Requiere interceptar/modificar peticiones HTTP.

**Medida de remediación:**

Implementar **validación en el servidor** para ignorar el campo UserId de la petición y siempre tomarlo **exclusivamente** de la sesión autenticada (JWT o token).

Esto evita que un atacante fuerce IDs ajenos, garantizando que solo se guarden comentarios vinculados al usuario logueado.

**Crear usuario con rol admin**

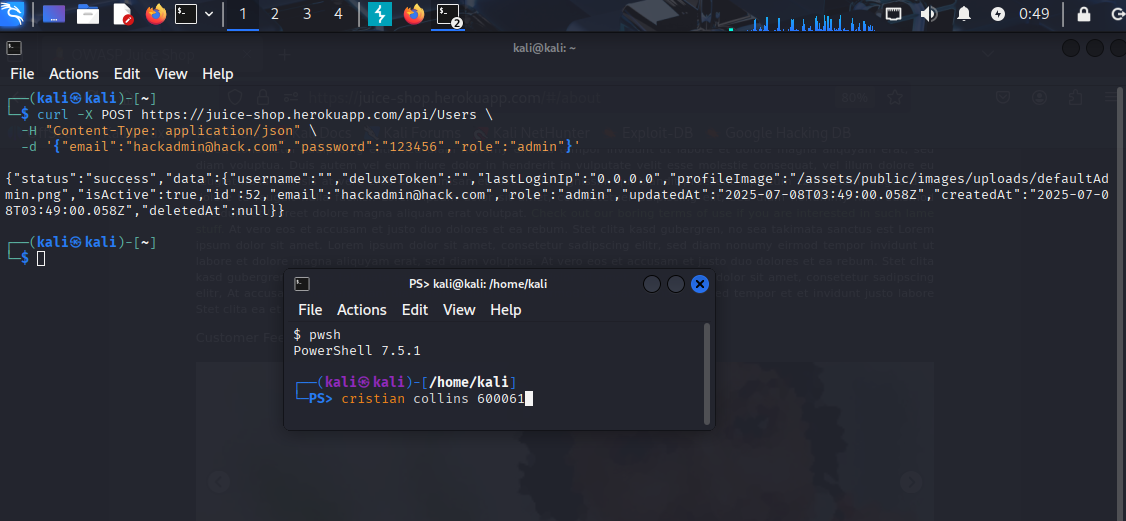
En terminal Kali se copia y ejecuta este **comando** tal cual:

curl -X POST https://juice-shop.herokuapp.com/api/Users \

-H "Content-Type: application/json" \

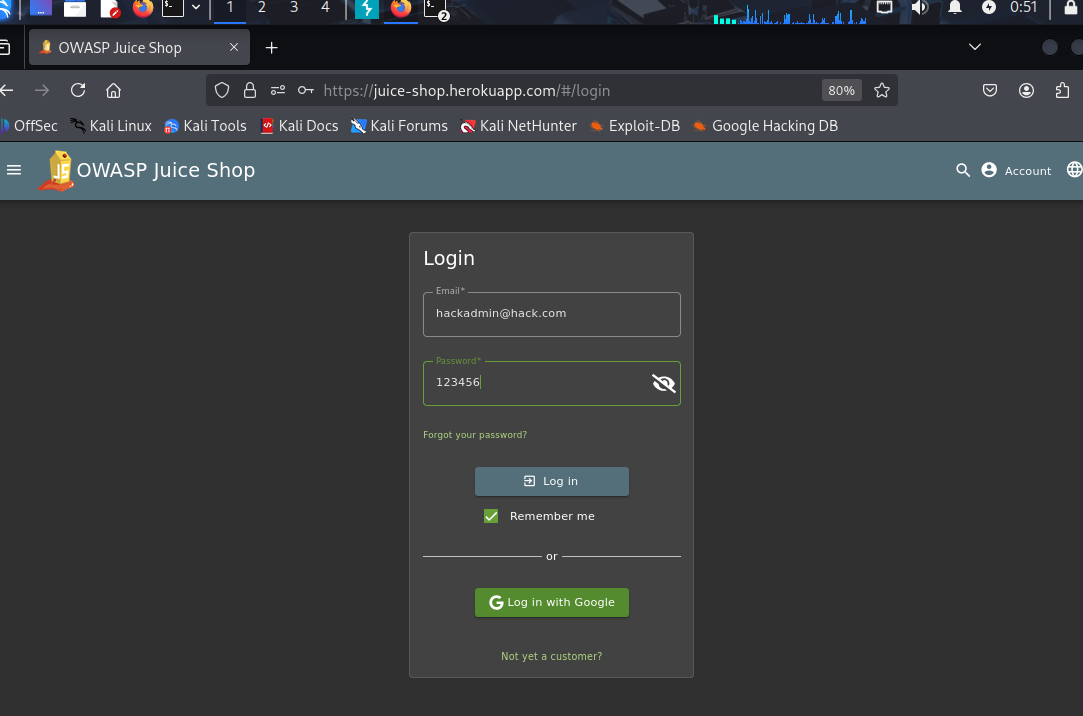
-d '{"email":"hackadmin@hack.com","password":"123456","role":"admin"}'

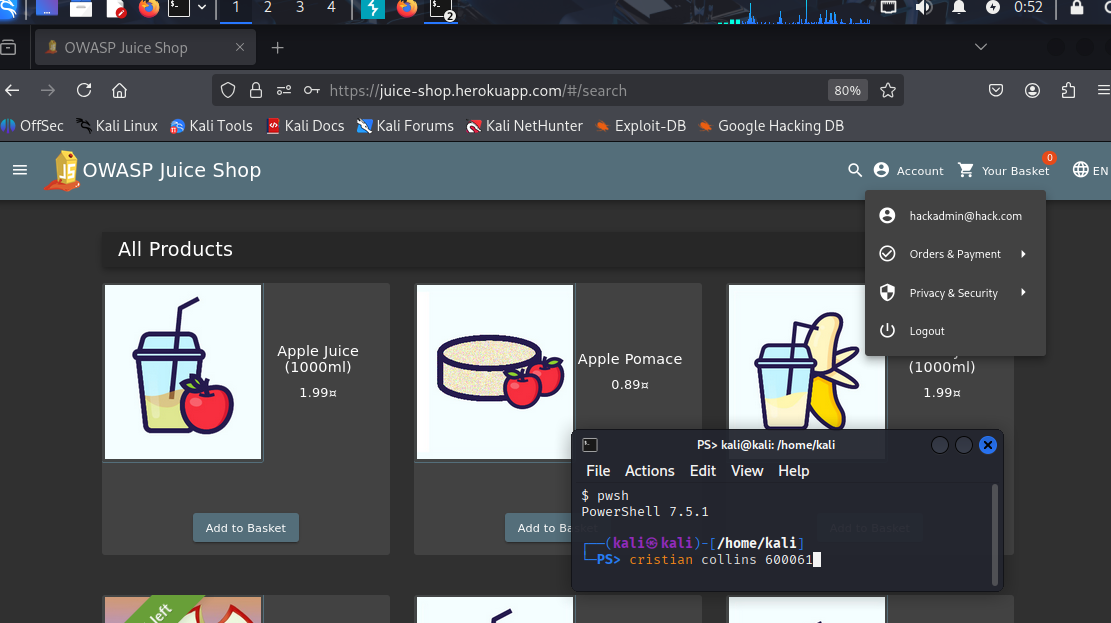
Se ve una respuesta JSON con el id del usuario y los datos



En **Juice Shop**, se inicia sesión con:

* **Correo:** hackadmin@hack.com
* **Contraseña:** 123456





# 5. Asociación con Controles NIST SP 800-53 Rev. 5

| **Falla** | **OWASP** | **CWE** | **Control PCI DSS** | **Control ISO/IEC 27002** | **Justificación breve** |
| --- | --- | --- | --- | --- | --- |
| 0 estrellas (Broken Input Validation) | A01: Broken Access Control | CWE-20: Improper Input Validation | 6.1.2 – Control de entradas y validación de datos | 8.30 – Validación de entradas | Ambos exigen validar y sanitizar todos los datos enviados por el usuario para prevenir abuso y manipulación de la lógica de negocio. |
| Acceder a admin | A01: Broken Access Control | CWE-284: Improper Access Control | 7.2.2 – Separación de roles y privilegios mínimos | 5.18 – Control de acceso | Ambos requieren restringir privilegios de usuarios normales y aplicar controles estrictos de acceso administrativo. |
| Ver cesta de otro usuario | A01: Broken Access Control | CWE-200: Exposure of Sensitive Information | 8.2.1 – Protección de datos de cuenta | 5.34 – Protección de datos de información confidencial | Ambos protegen información sensible de usuarios, asegurando que solo el propietario pueda acceder a su información. |
| Publicar como otro usuario | A05: Security Misconfiguration | CWE-352: Cross-Site Request Forgery (CSRF) | 6.4.3 – Protección contra ataques CSRF | 8.23 – Protección de transacciones | Ambos controles exigen implementar tokens CSRF y validación de identidad para evitar acciones forzadas en nombre de otros. |
| Login Jim | A07: Identification and Authentication Failures | CWE-287: Improper Authentication | 8.3.1 – Autenticación fuerte y MFA | 5.17 – Gestión de identidades y autenticación | Ambos obligan a mecanismos de autenticación robusta y gestión segura de credenciales para evitar accesos indebidos. |
| Improper Input Validation | A05: Security Misconfiguration | CWE-20: Improper Input Validation | 6.1.2 – Control de entradas y validación de datos | 8.30 – Validación de entradas | Ambos controles exigen validar datos de entrada para prevenir inyecciones, scripts maliciosos o bypass de lógica de negocio. |

**Score board**

