# Buenas prácticas de programación en Twitter

**Autor:** Twitter

**Artículo original:** http://apiwiki.twitter.com/Security-Best-Practices

**Edición y Corrección:** Lic. Cristian Borghello, CISSP

**Fecha Publicación:** 05 de septiembre de 2009

**Traducción y Publicación en Segu-Info**

## *Introducción*

Muchas de las buenas prácticas listadas a continuación pueden ser útiles para cualquier programador, incluso fuera de las APIs de Twitter.

Una aplicación para Twitter exitosa es probable que consiga algo de atención. La mayor parte de esa atención será buena: usuarios cantando alabanzas o desarrolladores complementando la programación. Sin embargo, algo de dicha atención podría ser negativa. Como Twitter ha incrementado en popularidad, su ecosistema se ha convertido en un objetivo más visible para usuarios malintencionados. Algunos están abocados a distribuir spam y malware, mientras que otros sólo quieren difundir el caos para diversión. Cualesquiera que sean sus motivos, su aplicación puede ser un objetivo.

Estos patrones están dirigidos hacia una mayor seguridad en su aplicación. No es la última palabra, ni mucho menos. Si hay algo que te gustaría que se añadiera a ella, por favor háganoslo saber. Si usted ha descubierto un problema de seguridad que afecta directamente Twitter, por favor, envíe un correo electrónico a security@twitter.com. Si desea utilizar GPG para encriptar su correo electrónico, por favor utilice nuestra clave pública.

## *Amenazas*

### Retención de contraseña

En pocas palabras, no conserve las contraseñas. El soporte de Twitter a la autenticación básica por HTTP (la forma estándar para autenticar a través de la web con un nombre de usuario y contraseña) será en breve obsoleta. Por favor, utilice OAuth.

Si usted está reteniendo contraseñas, por favor, cífrelos. En Twitter, usamos bcrypt-ruby, pero hay muchas otras maneras de almacenar contraseñas cifradas. Pero, no. No lo haga. No almacene las contraseñas. Sólo almacene los tokens OAuth. Por favor.

### Validar entradas

No asuma que usuarios van a proveerle datos válidos y confiables. Sanitice los datos, chequee que la longitud sea correcta, valide el tipo de datos, etc. Twitter intenta sanitizar la información utilizada por nuestras APIs, pero ayuda del

lado del cliente será bienvenida. Whitelist de entradas válidas son aceptables en una aplicación, para descartar todo lo que no esté en la Whitelist.

### Comunicaciones sin cifrar (no SSL)

Twitter ofrece todos los métodos de la API REST a través de SSL. Cada vez que su código pueda estar operando en una red insegura (es decir, si está desarrollando una aplicación de cliente), por favor haga uso de SSL para todas las solicitudes de autenticación o sensibles. Por ejemplo, la publicación de un nuevo estado, la solicitud de los últimos mensajes directos o la actualización de todos los atributos de perfil; deben realizarse a través de SSL en un cliente de Twitter. Es seguro y se sugiere utilizar SSL en conjunto con OAuth. Las comunicaciones Servicio-a-Servicio no podrán beneficiarse de SSL si confías en tu proveedor de hosting (o eres tu propio proveedor de hosting).

### Exponiendo información de Debugging

Asegúrese de que no expone información sensible a través de debbugs de pantallas y registros. Algunos frameworks Web facilitan el acceso a la información de depuración si su aplicación no está correctamente configurada. Para desarrolladores de escritorio y móviles, es fácil de accidentalmente construir una aplicación con banderas o símbolos de depuración habilitados. Genere controles para estas configuraciones en su proceso de desarrollo.

### Testing inadecuado

Asegúrese de que sus pruebas (usted tiene pruebas, ¿verdad?), comprueben no solo que usted puede hacer lo que debería ser capaz de hacer, sino también que los chicos malos no puedan hacer lo que no deberían ser capaces de hacer. Póngase en el modo de pensar de un atacante y realice pruebas malintencionadas.

### No dejar a la gente ayudar

¿Ha creado la cuenta security@yourapplication.com? ¿Los mensajes de correo electrónico son dirigidos a su teléfono? Haga que sea fácil para las personas contactarse con usted acerca de posibles problemas de seguridad en su aplicación. Si a alguien reporta un fallo de seguridad, sea amable con ellos, que acaban de hacerle un favor enorme. Agradézcale por su tiempo y arregle el problema a la brevedad. Es bastante común que los investigadores de seguridad escriban acerca de las vulnerabilidades que han descubierto una vez que el agujero se ha cerrado, así que no se enoje si su aplicación termina en un blog o trabajo de investigación. La seguridad es difícil, y nadie es perfecto. Mientras que usted repare los problemas que se reporten, usted estará haciendo las cosas bien.

Considere la posibilidad de contratar profesionales en seguridad para hacer una auditoría y/o un test de penetración. Usted no puede depender únicamente de la bondad de extraños. Para cada vulnerabilidad que alguien tuvo la gentileza de informarle, hay 10 atacantes más que la han encontrado. Una buena empresa de seguridad puede "cavar hondo" para descubrir problemas. Busque empresas y consultores que utilicen más que algunas herramientas automatizadas.

### La ley

Si su aplicación va a involucrar el uso de dinero, usted necesitará un abogado para adherir a ciertas regulaciones de seguridad. Busque cuáles son aplicables y asegure su código en función de estas.

## *Web Application Security*

### Entradas sin filtrar, salida sin escapar

Un enfoque fácil de recordar para la validación de entrada es FEES: Filtre la Entrada, Escape la Salida (en inglés FIEO, Filter Input, Escape Output). Filtre todo lo que provenga desde fuera de su aplicación, incluidos los datos de las API de Twitter, datos de la cookie, información suministrada por el usuario en formularios de entrada, parámetros URL, datos de bases de datos, etc. Escape toda la salida que es enviada por su aplicación, incluyendo instrucciones SQL enviadas al servidor de base de datos, HTML enviado a los navegadores de los usuarios, salidas JSON y XML enviados a otros sistemas, y los comandos enviados a los programas de shell.

### Cross-Site Scripting (XSS)

Los ataques XSS son, según muchas mediciones, el problema de seguridad más común en la web. En resumen, si un atacante puede inyectar su propio código Java Script en su aplicación, puede hacer cosas realmente malas. Donde sea que donde almacene y muestre los datos proporcionados por el usuario, estos necesita ser chequeados, sanitizados y escapados en su HTML. Hacer esto correctamente es difícil, porque los hackers tienen muchas maneras diferentes de realizar un ataque XSS. Su lenguaje de programación, o marco de desarrollo web, probablemente tiene un popular y bien probado mecanismo de defensa contra el cross-site scripting, por favor haga uso de él.

En general: su el HTML no es necesario para algún formulario para el usuario, filtrelo hacia afuera. Por ejemplo, no hay razón para permitir algo más que números enteros al almacenar un número de teléfono. Si el HTML es necesario, utilice un filtro de Whitelist reconocida. HTMLPurifier para PHP es una solución de este tipo. Contextos diferentes pueden requerir diferentes métodos de filtrado. Vea la Hoja de prevención de XSS de OWASP para más información sobre el filtrado.

### Injección SQL

Si su aplicación hace uso de una base de datos, debe ser consciente del ataque de Injección SQL (en inglés SQL injection). Una vez más, en cualquier lugar que acepte entradas es un blanco potencial para un atacante para ingresar en su base de datos. Utilice colecciones de bases de datos que protegen contra el uso de inyección SQL de una forma sistemática. Si sale de este enfoque y escribe SQL personalizado, realice pruebas agresivas para asegurarse de que no se está exponiendo a esta forma de ataque.

Los dos enfoques principales para la defensa contra la inyección SQL es escapar antes de construir la instrucción SQL y el uso de entrada parametrizadas para crear las instrucciones. Se recomienda este último, ya que es menos propenso a errores por parte del programador.

### Cross-Site Request Forgery (CSRF)

¿Está seguro que las solicitudes para su aplicación son provenientes de su aplicación? Ataques CSRF aprovechan esta falta de conocimiento al forzar a usuarios logueados en su sitio a abrir, en silencio, direcciones URL que realicen acciones. En el caso de una aplicación de Twitter, esto podría significar que los atacantes están utilizando su aplicación y fuercen a los usuarios a enviar tweets no deseados o seguir cuentas de spam. Usted puede aprender más acerca de este tipo de ataque en el blog de expertos de seguridad de PHP Chris Shiflett's.

La manera ideal para hacer frente a CSRF, consiste en incluir un token al azar en todos los formularios que se almacenen en algún lugar de confianza. Si un formulario no tiene el token correcto, lanza un error. Los frameworks web modernos tienen formas sistemáticas para manipular esto, e incluso pueden estar haciéndolo de forma predeterminada, si usted tiene suerte. Una medida preventiva simple (pero de ningún modo el único paso que debe tomar) es hacer que cualquier acción que crease, modifique o destruya datos requiera una solicitud del tipo POST.

### Limitar las acciones

Asegúrese de que los usuarios no pueden tomar ventaja de su aplicación por la realización reiterada de recursos o acciones socialmente perjudiciales. Por ejemplo, no permita a los usuarios en varias ocasiones intentar ingresar o restablecer las contraseñas. Twitter limita algunas acciones (demasiado tweeting, seguir a muchas personas), pero por favor, no se basen exclusivamente en la API para manejar esta limitación. Utilice CAPTCHAs en su caso para frenar los spammers y los atacantes potenciales.

### Falta de información sobre amenazas

Si usted piensa que hay un problema con su aplicación web, ¿cómo pueden saberlo con seguridad? Haga que las excepciones y errores críticos sean enviados por correo electrónico y mantenga los registros (logs) bien organizados. Es posible que desee formar un panel de estadísticas críticas de manera que usted pueda ver de un vistazo si algo va mal (o todo va en orden).

## *Seguridad en aplicaciones de escritorio*

*Podríamos utilizar sugerencias para desarrolladores de escritorio acerca de los problemas de seguridad que se han encontrado. Los desarrolladores que trabajan en lenguajes lo suficientemente altos no deben hacer frente a desbordamientos de búfer o problemas de seguridad habituales. ¿De qué se defiende?*

### *A*lmacenamiento de credenciales sin cifrar

Como se mencionó anteriormente, para optimizar la seguridad usted debe utilizar OAuth. Pero una vez que usted tiene un token con el cual hacer peticiones en nombre de un usuario, ¿dónde los almacena? Idealmente, debe hacerlo en un almacén cifrado y administrado por el sistema operativo. En sistemas Mac OS X, esto sería Keychain. En el entorno de escritorio GNOME, está Keyring. En el entorno de escritorio KDE, KWallet.

## *Security Resources*

Los siguientes enlaces (en inglés) son excelentes formas de aprender más sobre seguridad. La seguridad es un tema complejo, pero no sienta que debe aprender todo para poder ejercitar proactivamente los pasos que van a segurizar su aplicación. Un poco de espíritu de seguridad es un buen camino.

### General

- Purdue CERIAS - Center for Education and Research in Information Assurance and Security
- US-CERT - current identified threats for various platforms and applications
- CERT Information for Developers - secure coding practices

### Web Application Security

- OWASP Top Ten Threats - the definitive list of the most pressing threats facing web application developers.
- Intrusion detection - you may want to run an Intrusion Detection System (IDS) to find potential attackers.
- PHP tools for filtering and validation - generously provided by veteran Twitter developer @funkatron.

### Desktop Application Security

- Apple Developer Connection - Security
- Apple Secure Coding Guide
- MSDN Security Developer Center
- Secure Programming for Linux and Unix HOWTO