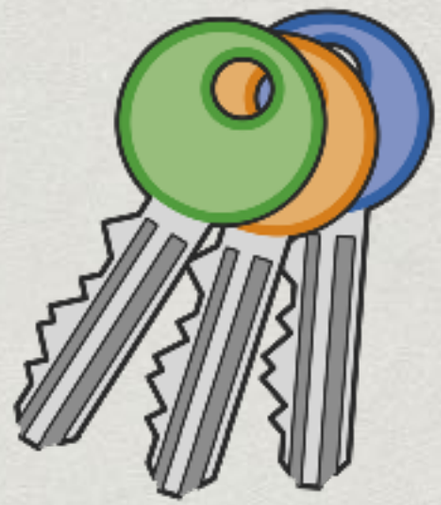
SEGURIDAD WEB

REFLEXIONES Y PRINCIPIOS

Aspectos claves

- Usar procedimientos almacenados para evitar problemas de seguridad como SQL injections
- Usar requests POST cuando la información sea confidencial
- Usar HTTPS para la comunicación



ACTIVIDAD

HTTPS mejora la seguridad de la comunicación

Request POST ayudan

¿SQL Injection?

¿QUÉ ES SEGURIDAD?

¿QUÉ SIGNIFICA QUE UNA APLICACIÓN SEA SEGURA?

https://xkcd.com/327/

https://xkcd.com/327/

https://xkcd.com/327/

https://xkcd.com/327/

```
custId = Request.QueryString("id")
query = "SELECT * FROM Customer WHERE CustId=" & custId
```

```
1; DELETE FROM Customer
```

```
SELECT * FROM Customer WHERE CustId=1; DELETE FROM Customer
```

```
address = request.getParameter(“address");

userId = (Integer) session.getValue(“usrid");

query = "UPDATE Usr SET Address=’" + address + "’ "
             + "WHERE Id=" + userId;
```

```
UPDATE Usr
SET Address=’’ || (SELECT Password FROM Usr
                          WHERE UserName=’john’) || ’’

WHERE Id=1234
```

```
UPDATE Usr
SET Address='' || (SELECT Password FROM Usr
                   WHERE UserName='john') || ''
WHERE Id=1234
```

'SQL'

```
char(83)+char(81)+char(76)
```
(MS SQL)

```
chr(83)||chr(81)||chr(76)
```
(PostgreSQL)

```
chr(83,81,76)
```
(MySQL)

```
0x53514C
```
(MySQL)

¿QUÉ PASA?

¿POR QUÉ NO PODEMOS PREVER CADA FORMA POSIBLE DE ATAQUE?

PRINCIPIO

DESARROLLADORES → **PENSAMIENTO CREATIVO**

ATACANTES → **PENSAMIENTO DESTRUCTIVO**

¿CÓMO DETENER SQL INJECTIONS?

Escapar meta-caracteres

Usar “prepared statements”

```
Customer.where("ref_id = ?", params[:id])
Customer.where(ref_id: params[:id])
```

Ejemplo en Rails de “prepared statements”

http://guides.rubyonrails.org/security.html

GUÍA DE SEGURIDAD DE RUBY ON RAILS

BREAK;

“Accepting input from the client is probably the greatest threat to the security of a web application”

–Sverre H. Huseby

¿Qué es *input*?

- Parámetros en la URL
- Información enviada (POST) de inputs, check boxes, selects, inputs de tipo “hidden”, etc.
- Cookies y cualquier otro encabezado HTTP

INPUT

PRINCIPIO

The invisble security barrier

CLIENTE

SERVIDOR

NUNCA CONFIAR EN UN INPUT

ACTIVIDAD

HTTPS://GITHUB.COM/JOHNOWENATALA/RAILS-SECURITY-TOY

SEGURIDAD WEB

REFLEXIONES Y PRINCIPIOS

Referencias

- Innocent code - a security wake-up call for web programmers. Sverre H. Huseby, 2004.
- The Web Application Hackers Handbook. Dafydd Stuttard and Marcus Pinto, 2008.
- Rails Security Guide, http://guides.rubyonrails.org/security.html