DEVSTAFF

MEET UP

“whoami”

-root

Who Am I?

- Security Engineer
- Vulnerability Researcher
- Bug Bounty Hunter
- CEO (?)

OWASP Top 10

the grugq
@thegrugq

Follow

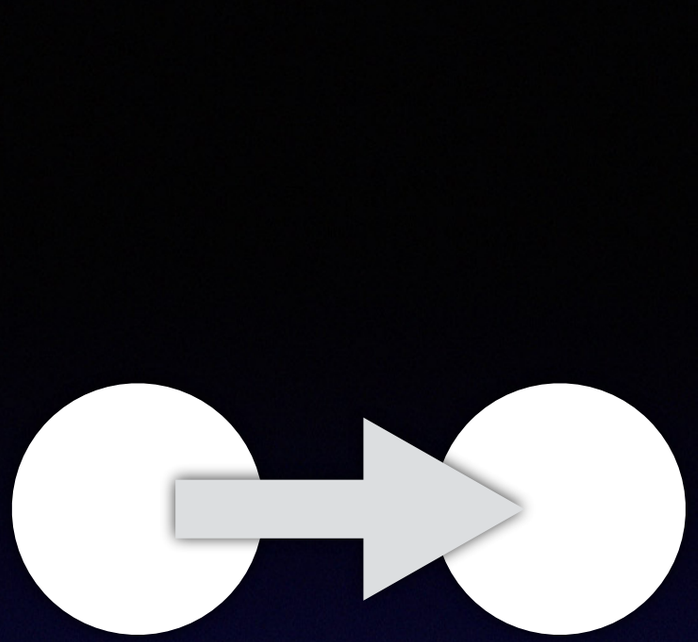
New rule: if you are hacked via OWASP Top 10, you're not allowed to call it "advanced" or "sophisticated."

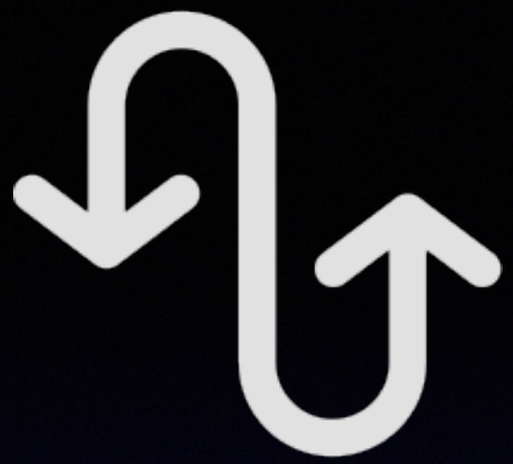
RETWEETS	LIKES
1,378	1,047

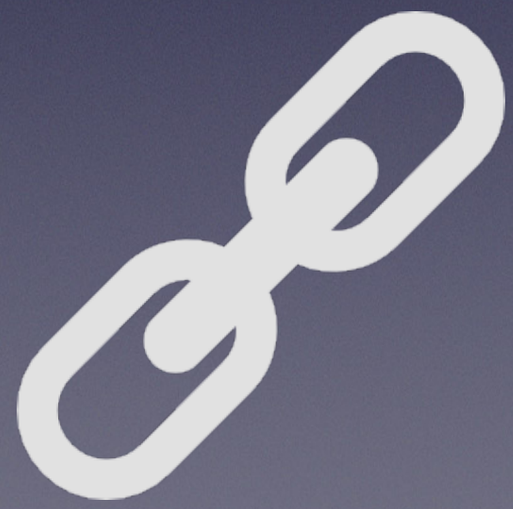
5:58 AM - 27 Oct 2015

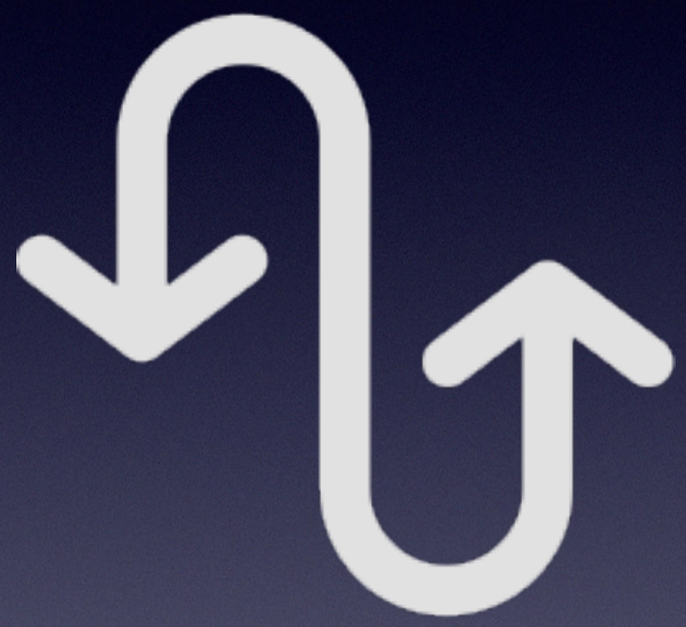
Q: What do you call software written with the sole purpose of downloading and executing arbitrary code on a user's system, without their explicit consent?

A: Web Browser





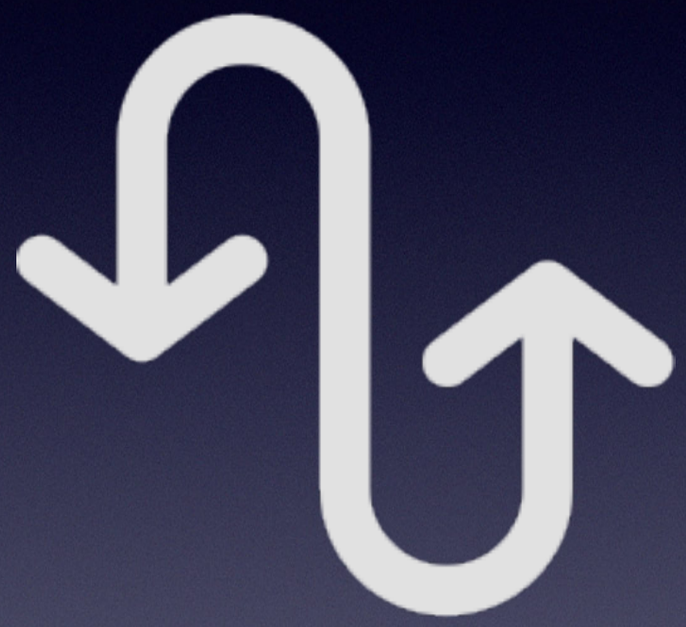




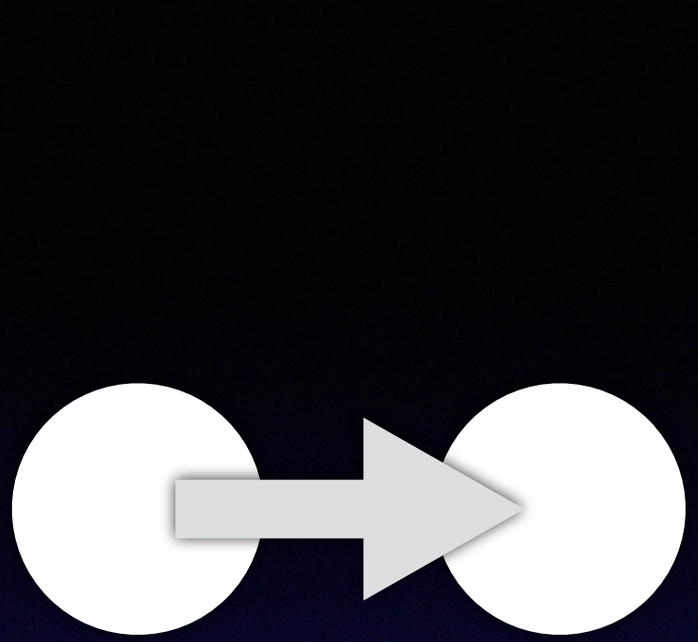
Unvalidated Redirects

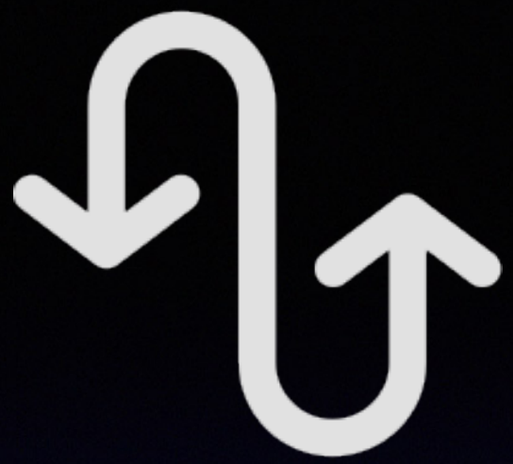
Unvalidated Redirects

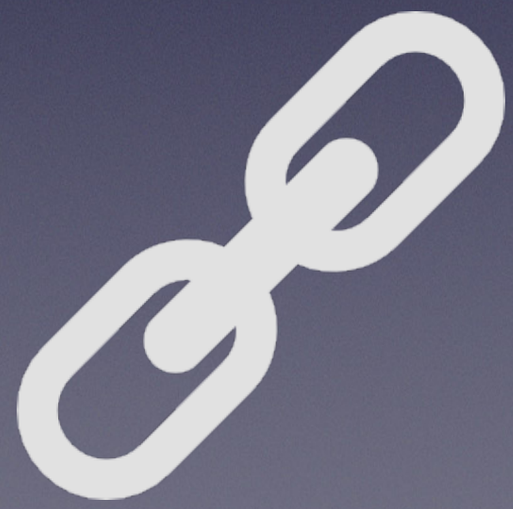
- Link to external website within the page
- Not trusted / Added by Owner
- Can lead to phishing site
- Can allow unauthorized page access

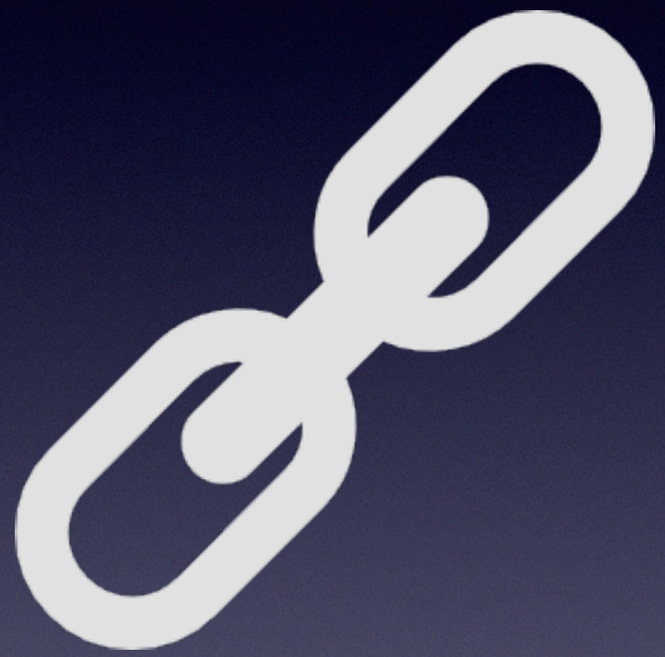


Unvalidated Redirects





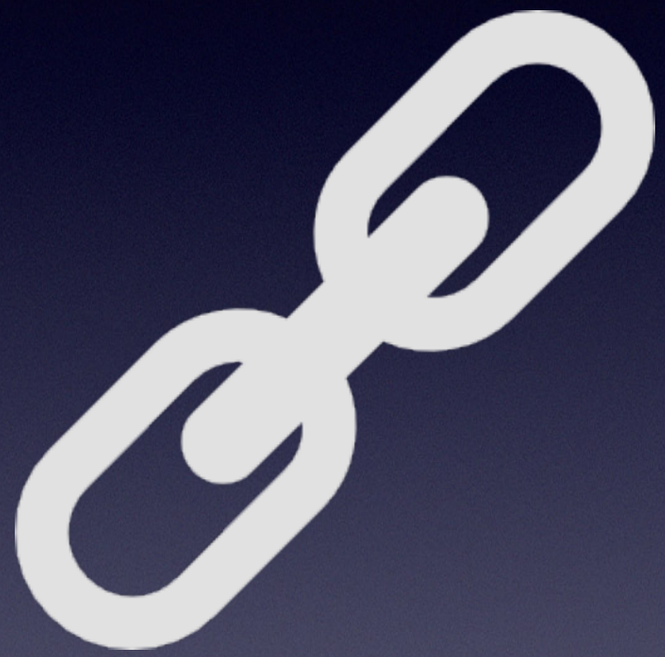




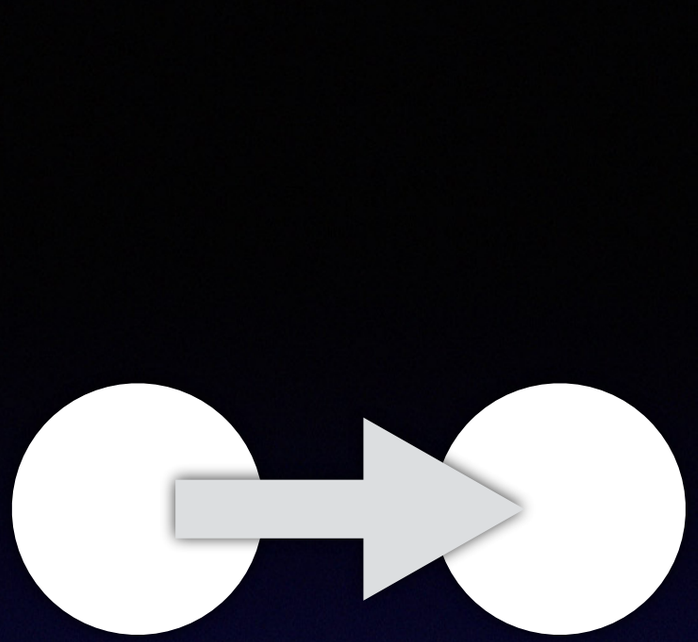
Using Vulnerable Components

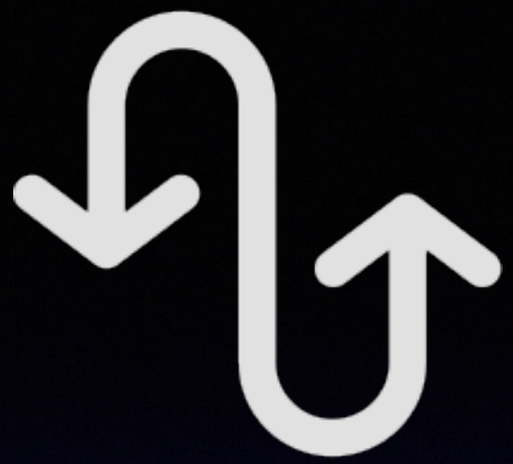
Vulnerable Component Use

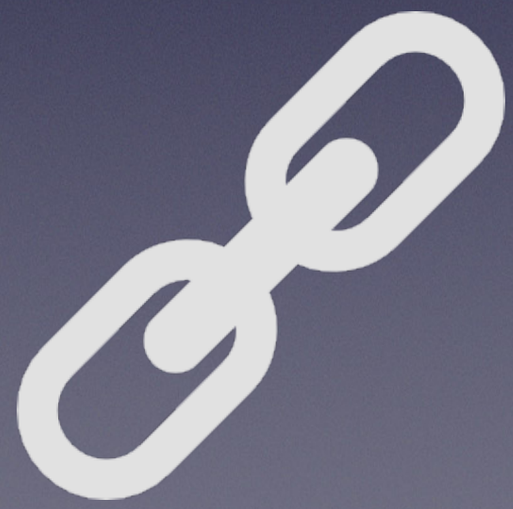
- Use Framework / Library / Software with known vulnerabilities
- Use backdoored modules in application
- Use old versions / Never maintain dependencies



Using Vulnerable Components







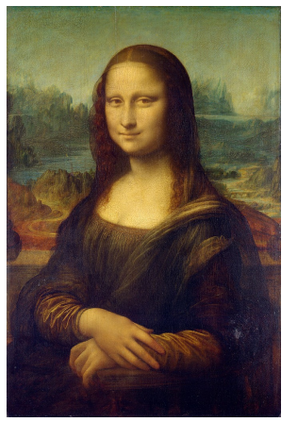
Cross Site Request Forgery (CSRF / Sea Surf)

CSRF

- Send HTTP Request from 3rd Party Website
- Use browser cookies to trigger action
  - Simple: Logout
  - Advanced: Account Takeover


```
<a href="/logout.php" class="button">Log Out</a>
```

Here is a nice picture!

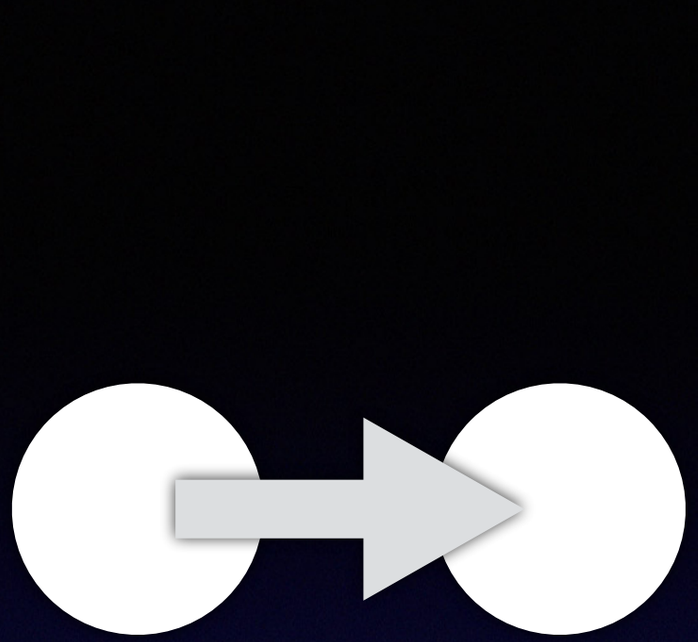


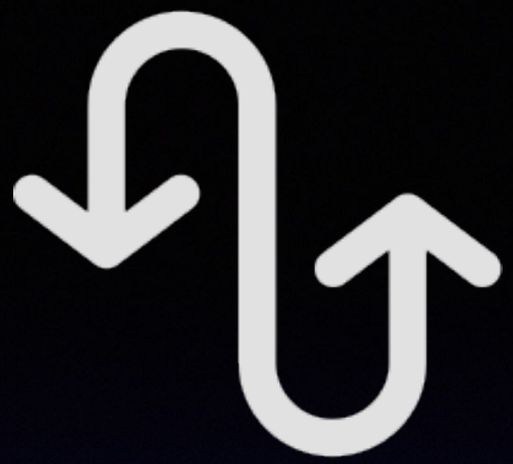
Tweet

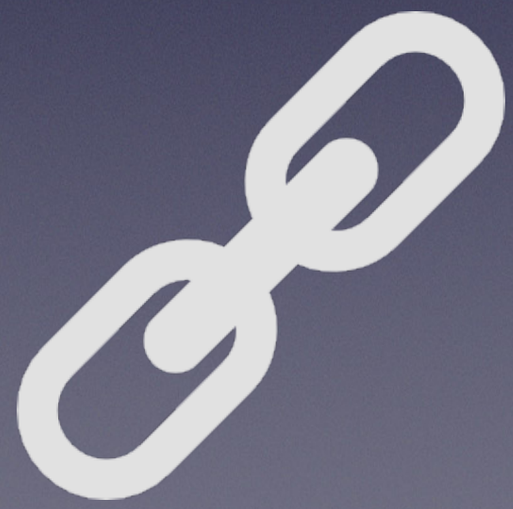
```
<form action=“tweet.php” method=“post”>
   <input type=“text” name=“tweet-content”>
   <input type=“submit” value=“Tweet”>
</form>
```

```
<body onload="$('#form').submit();">
  <form action="https://webapp.com/tweet.php"
method="post" id="form">
    <input name="tweet-content" value="Hacked">
  </form>
</body>
```

Cross Site Request Forgery (CSRF / Sea Surf)







Missing Function Level AC

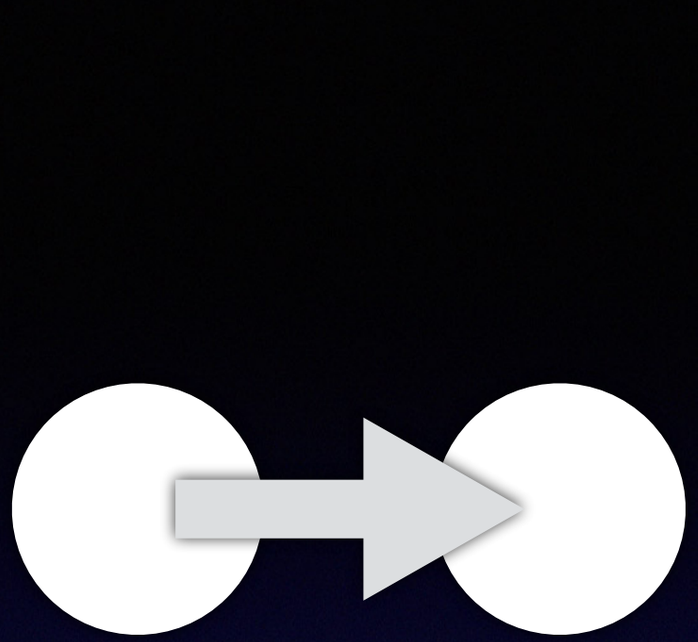
Missing Access Control

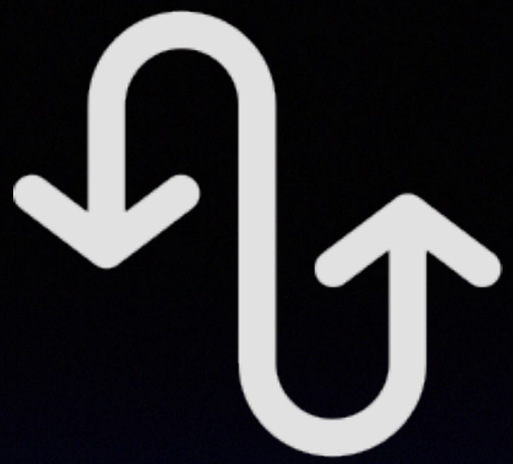
- Client-Side Checks
- Lack of multi-stage verification / authentication server-side
- Excessive trust in internal components

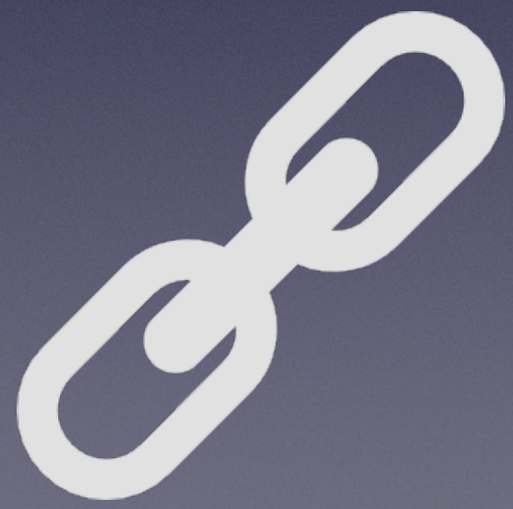
```
<script type="text/javascript">
    if ($("#password").val() == "password1234"){
        window.location = "/logged-in.php";
    }
</script>
```

```
function changePassword($User, $NewPassword){
    $PassHash = md5($NewPassword);
    $mysql->query(“UPDATE users SET password =
‘$PassHash’ WHERE username = ‘$User’;”);
}
```

Missing Function Level AC



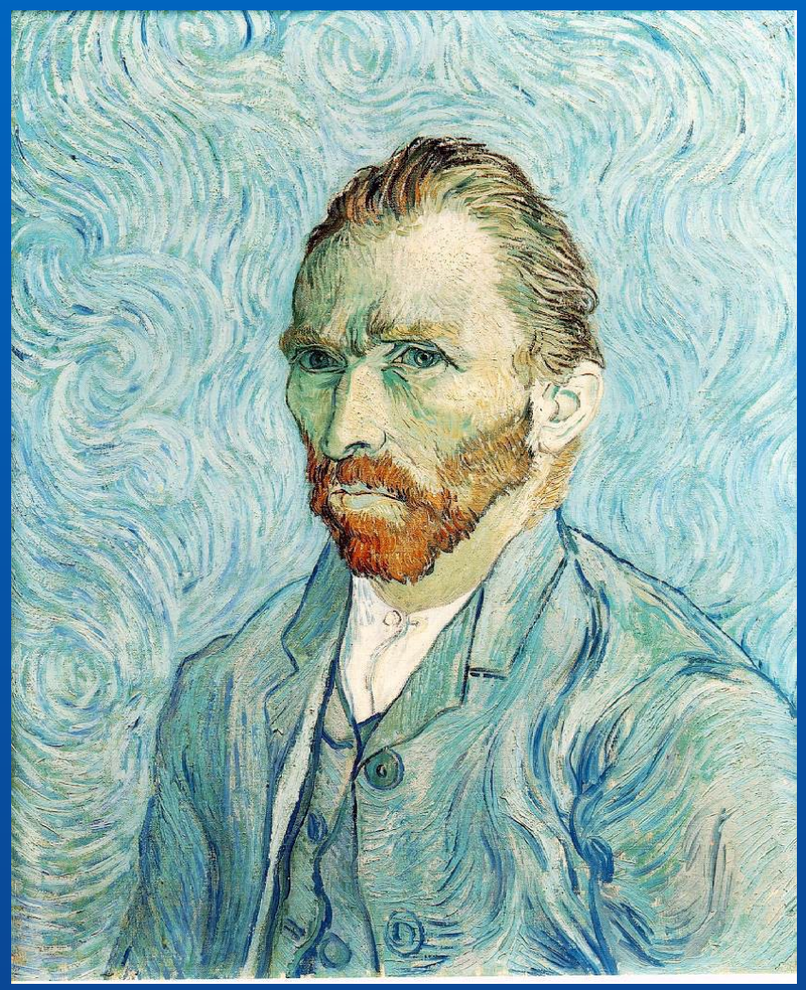




Sensitive Data Exposure

Sensitive Data Exposure

- Unauthenticated Data Access
- Weak / Lack of Access Control
- Data not protected at rest / transit
- Configuration File Access

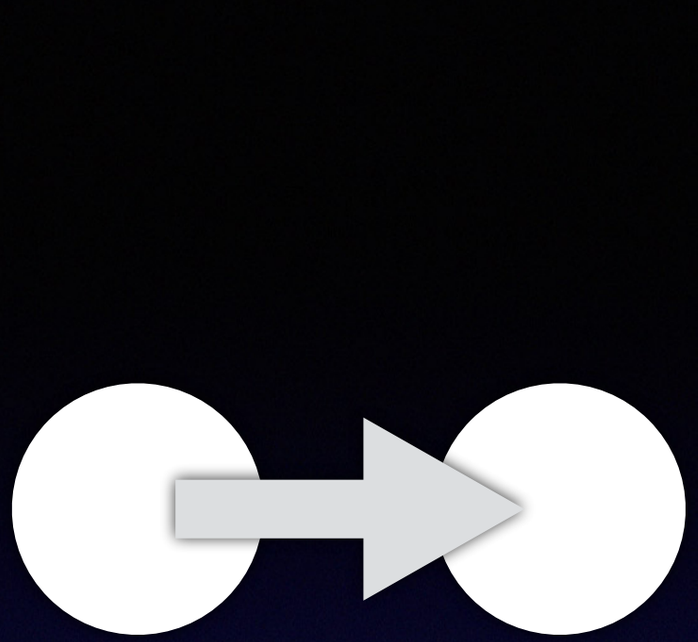


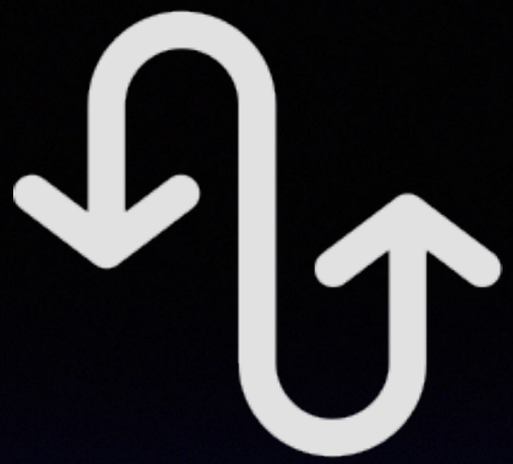
Picture:
Selfie_00183.JPG
Tags:
Me
Privacy Settings:
Only Me 🔒

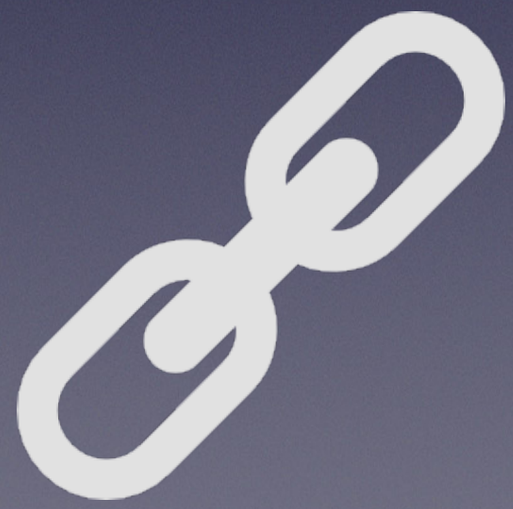
<img src="/images/vvangogh/selfie_00183.jpg">

```
$ curl https://webapp.com/config.txt

AppTitle=“Tweeter”
MySQL=“db.tweeter.com”
MySQLUser=“root”
MySQLPass=“6W3xZ945L05JZan7t69Po”
Photos=“/var/lib/tweeter/images”
```

Sensitive Data Exposure







Security Misconfigurations

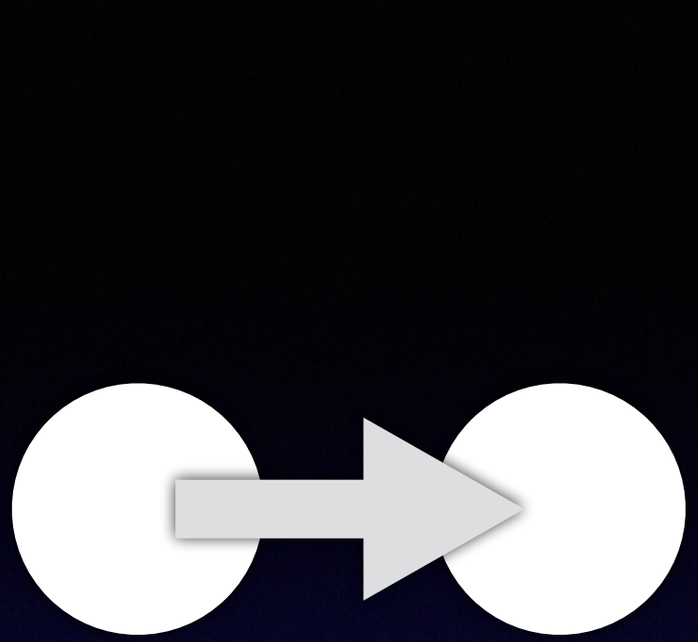
Security Misconfigurations

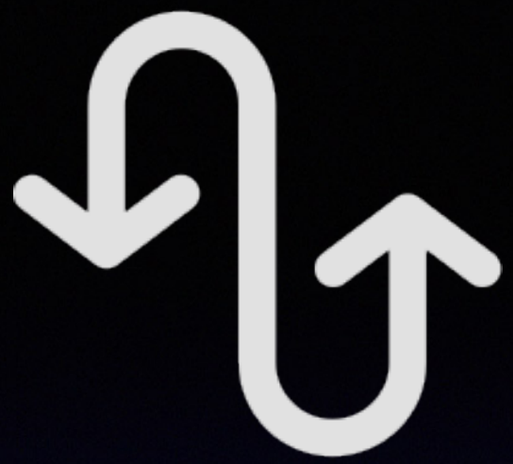
- Enable Debug / Development Mode in Production
- Fail to disable unused functions
- Fail to add language / web server protection mechanisms
- Misconfigure Security Components
- HTTP (DEADLY)
  - Mixed Content

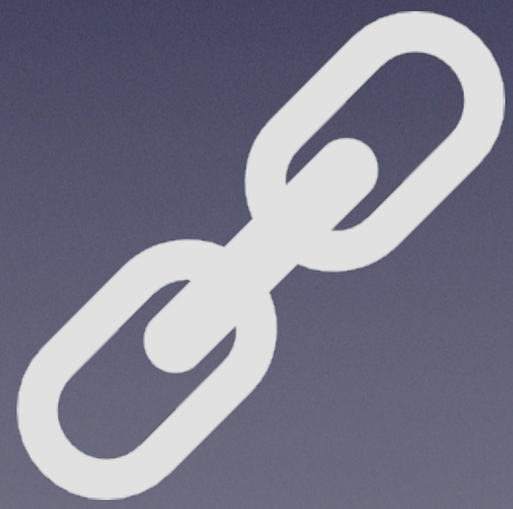
```
<a href="http://webapp.com/activate.php?[...]">
   Verify E-Mail Address
</a>
```

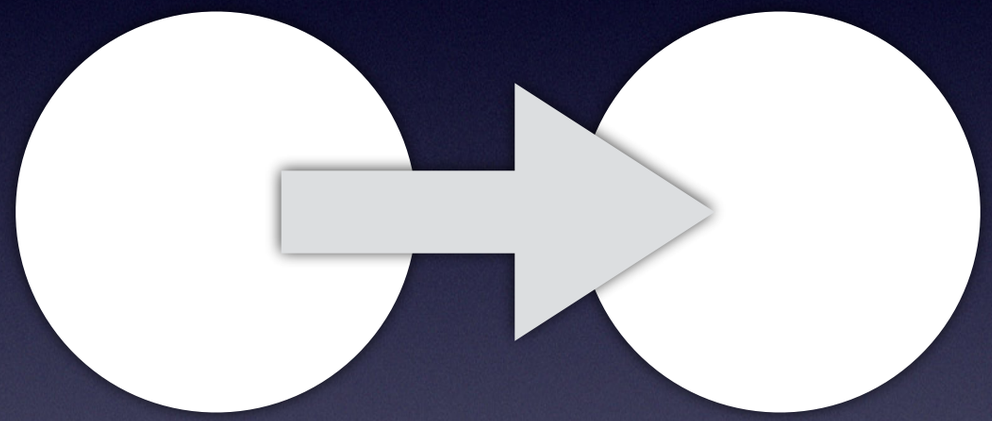
<script type="text/javascript" src="http://examplecdn.com/heelstrap.js"></script>

Security Misconfigurations







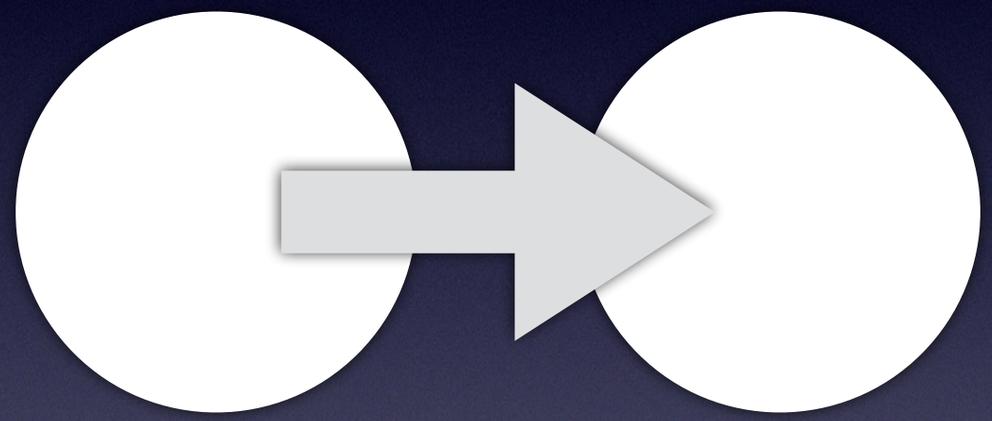


Direct Object Reference

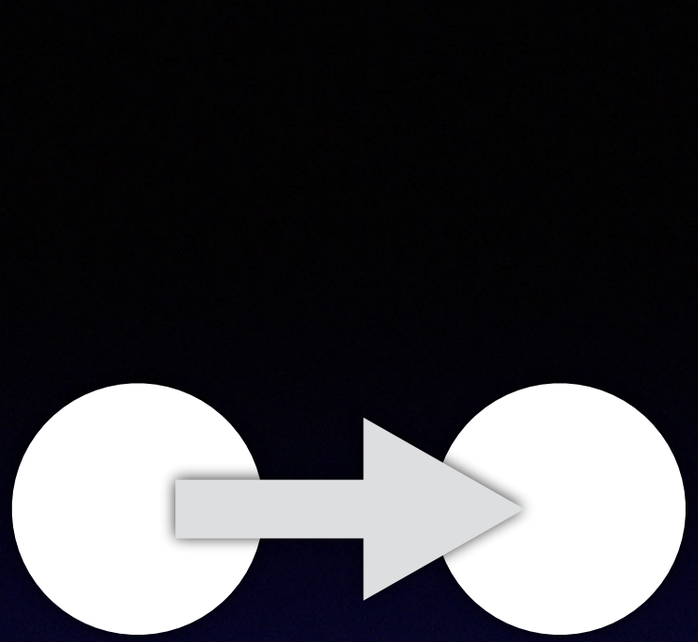
Direct Object Reference

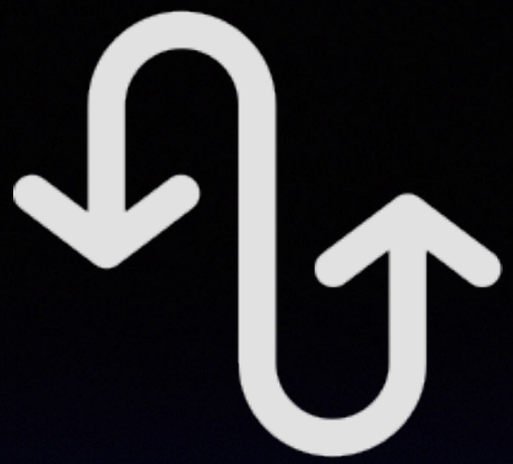
- Expose Internal Objects to User Interface
  - Structs / Classes
  - Databases and queries
  - File Structures / Contents

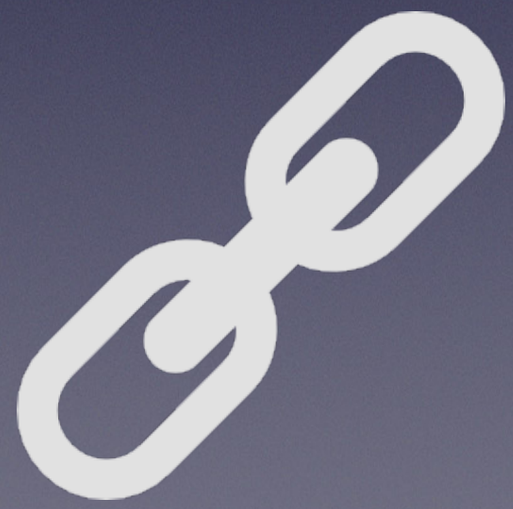
```
$UserName = $_SESSION[‘username’];
$Bio = $_GET[‘bio’];
fwrite($jsondb, ‘[“user”: “$Username”,
 “bio”: “$Bio”,
 “money”: “10.00”
]’);
```



Direct Object Reference







Cross Site Scripting (XSS)

Cross Site Scripting

- Persistent
- Reflected
- Inject Attacker Controlled JavaScript
- Hijack entire website content / functionality

Reflected

https://webapp.com/search.php?term=Hello

Reflected

```
if ( !searchFor ( $_GET[‘term’] ) ){
    print “No results for ‘“ . $_GET[‘term’] . “‘.”;
}
```

Reflected

No results for ‘Hello’.

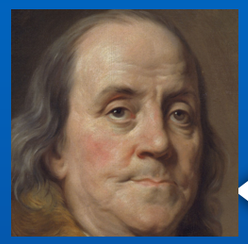
Reflected

https://webapp.com/search.php?term=
<script>alert(“XSS”);</script>

Reflected

No results for ‘
<script>alert(“XSS”);</script>
’.

Stored

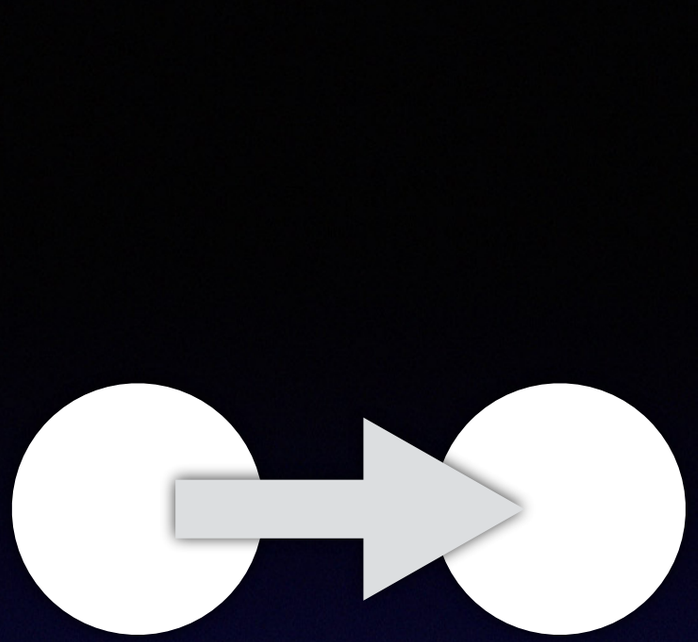


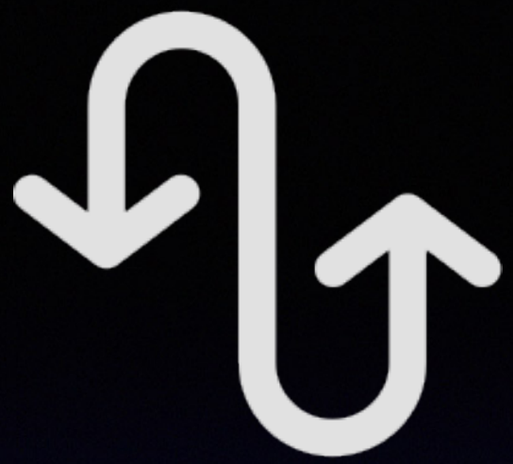
Good morning!

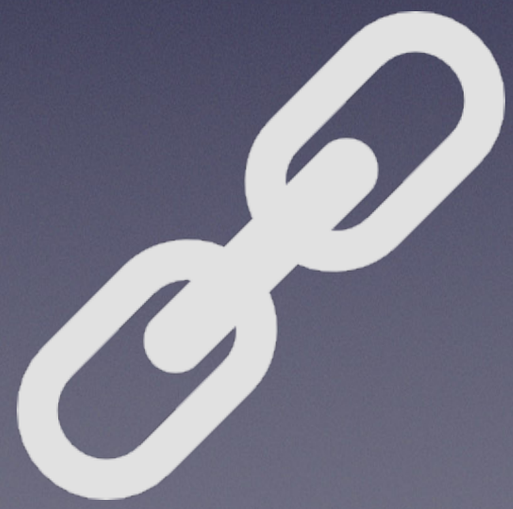
<script>alert(“XSS”);</script>

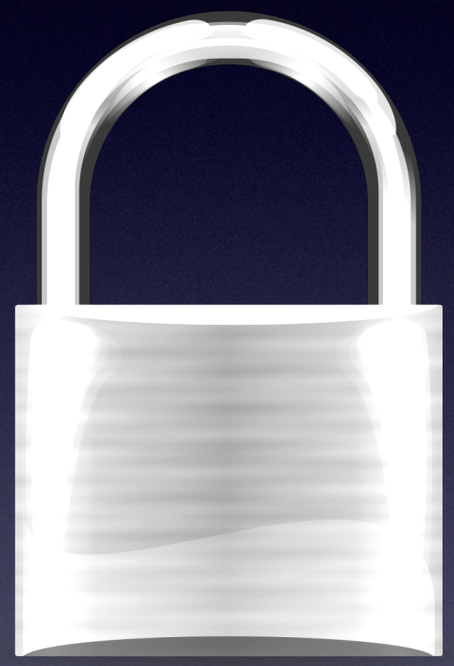
Send

Cross Site Scripting (XSS)









Broken Authentication
Broken Session Management

Broken Auth / Session Mgmt

- Inadequate checks in login, permissions, etc.
- Bad Session Handling
- Bad Cookie Handling

```
function checkPassword($Username, $Password){
    return true;
}
```

```
function checkPassword($Username, $Password){
    $dbHash = getPasswordFromDatabase($Username);
    $loginHash = passhash($Password);
    if ( $dbHash == $loginHash ) {
        return true;
    } else {
        return false;
    }
}
```

PHP Magic Hash

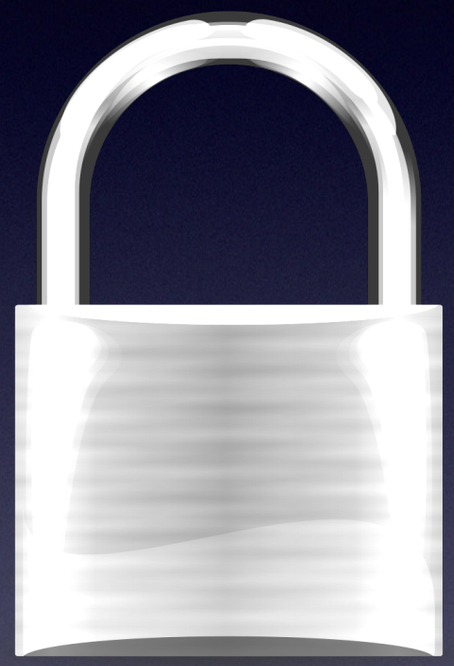
- If the user's password hash starts with "0e"
- If the attacker attempts a login with a password whose hash starts with "0e"
- PHP thinks you're comparing "0" with "0"
- md5(240610708)

Cookie Fixation

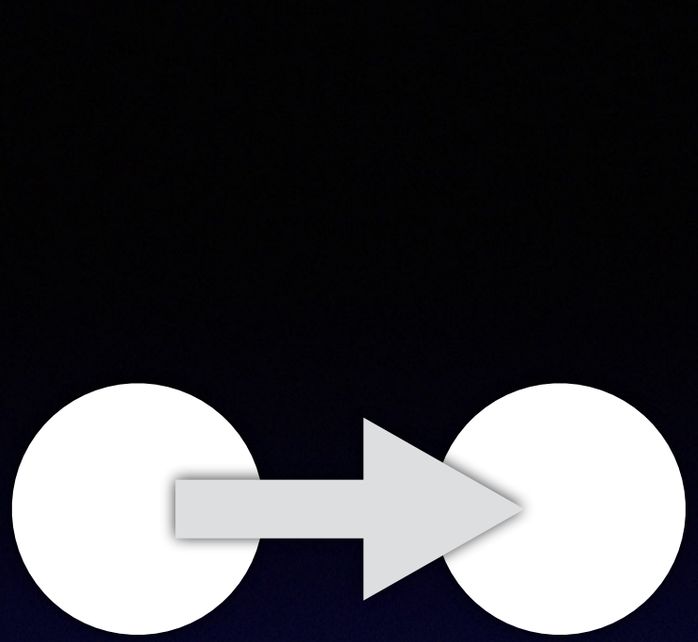
```
$ cat login.php
if ( userLoginCheck($Username, $Password) ) {
    setcookie(“username”, $Username);
    header(“/profile.php”);
}
```

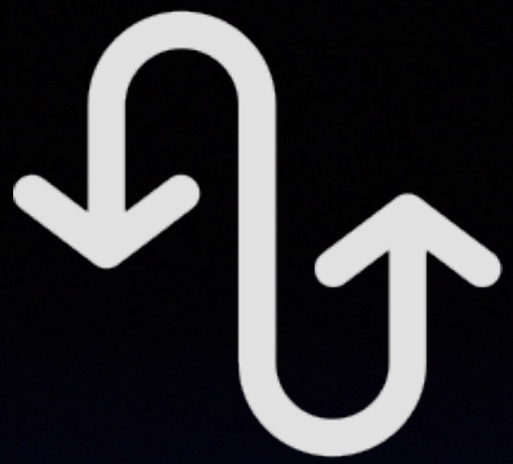

```
<a class="button" href="/delete.php?
username=ppicaso"></a>
```

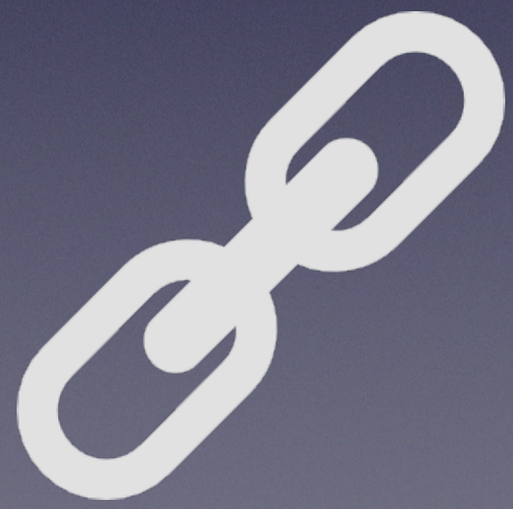
```
$ cat register.php
<?php
 $Username = $_POST[‘username’];
 $Password = $_POST[‘password’];

 if ( passwordIsStrong($Password) ) {
  $hash = passhash($Password);
  $db->query(“INSERT INTO users
  (username,password)
  VALUES
  (‘$Username’, ‘$hash’);”);
 }
?>
```



Broken Authentication
Broken Session Management







SQL Injection

SQL Injection

- User is able to inject content into database
- No input sanitization
- No database security checks

```
$user = $_GET[‘user’];
```

```
SELECT * FROM users WHERE username=“$user”;
```

$ curl https://webapp.com/profile.php?user=john

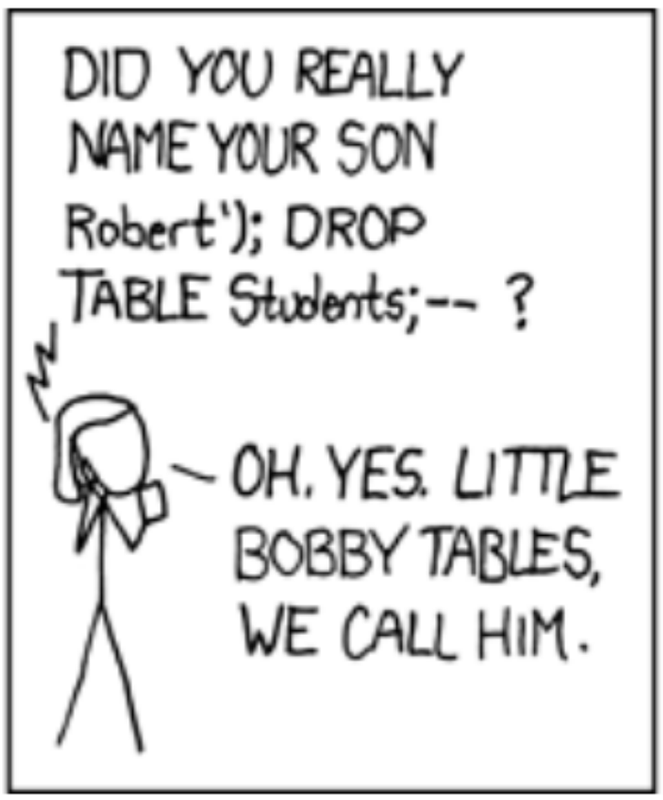
SELECT * FROM users WHERE username=“john”;

$ curl https://webapp.com/profile.php?user=“;
DROP TABLE users; - - -

SELECT * FROM users WHERE username=““;
DROP TABLE users; - - -”;


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.


OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY-


DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.


WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Code Execution

- SQL Databases can write files
  - Like /var/www/shell.php
  - Or /etc/passwd
  - Or /root/.ssh/authorized_keys
- Just sayin'…

Data Manipulation

- SQL Databases can change their data
- To whatever the query asks it to
- And that’s normal

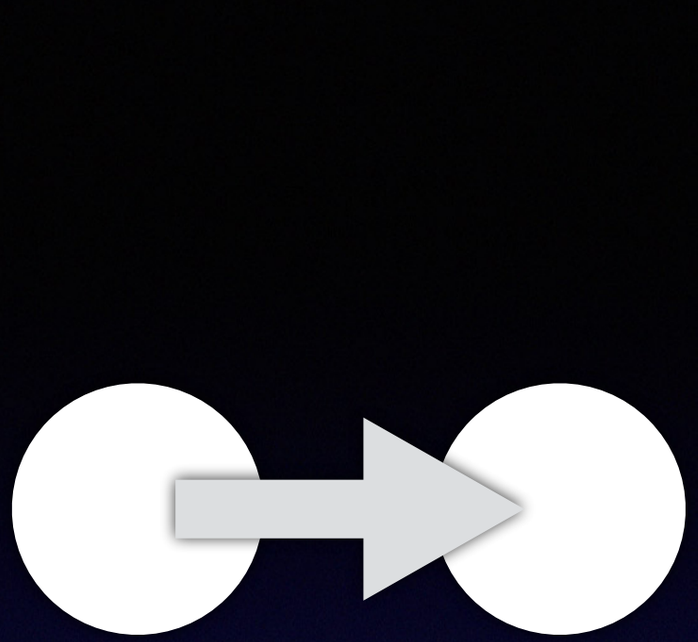
Data Exfiltration

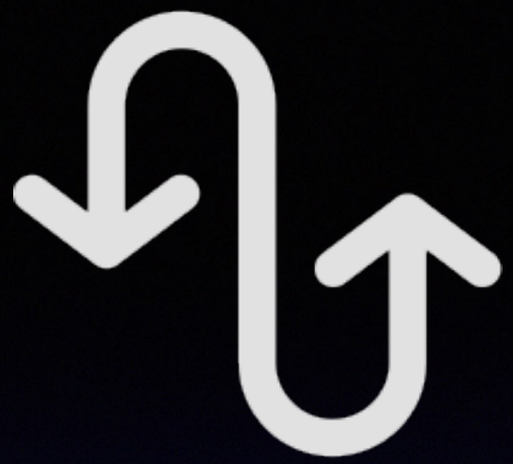
- SQL Databases can read their content
- All of it
- Really fast

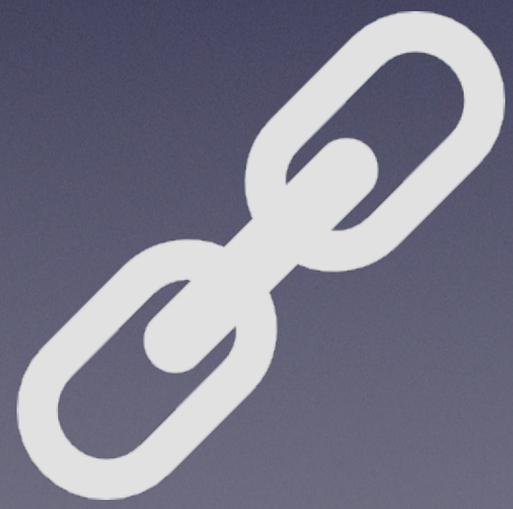
Data Loss

- SQL Databases can delete their data
- It only takes a query
- I hope you keep backups

SQL Injection







Stay Safe