Dip Your Toes in the Sea of Security

James Titcumb
PHP Dorset
2nd June 2014

Who is this guy?

James Titcumb

www.jamestitcumb.com

www.protected.co.uk

www.phphants.co.uk

@asgrim

protected.co.uk

php hampshire

Who are you?

https://www.flickr.com/photos/akrabat/10168019755/

Some simple code...

```php
<?php


$a = (int)$_GET['a'];
$b = (int)$_GET['b'];
$result = $a + $b;


printf('The answer is %d', $result);
```

The Golden Rules

The Golden Rules

1. Keep it simple

The Golden Rules

1. Keep it simple
2. Know the risks

The Golden Rules

1. Keep it simple
2. Know the risks
3. Fail securely

The Golden Rules

1. Keep it simple
2. Know the risks
3. Fail securely
4. Don’t reinvent the wheel

The Golden Rules

1. Keep it simple
2. Know the risks
3. Fail securely
4. Don’t reinvent the wheel
5. Never trust anything / anyone

OWASP
& the OWASP Top 10

https://www.owasp.org/

Application Security

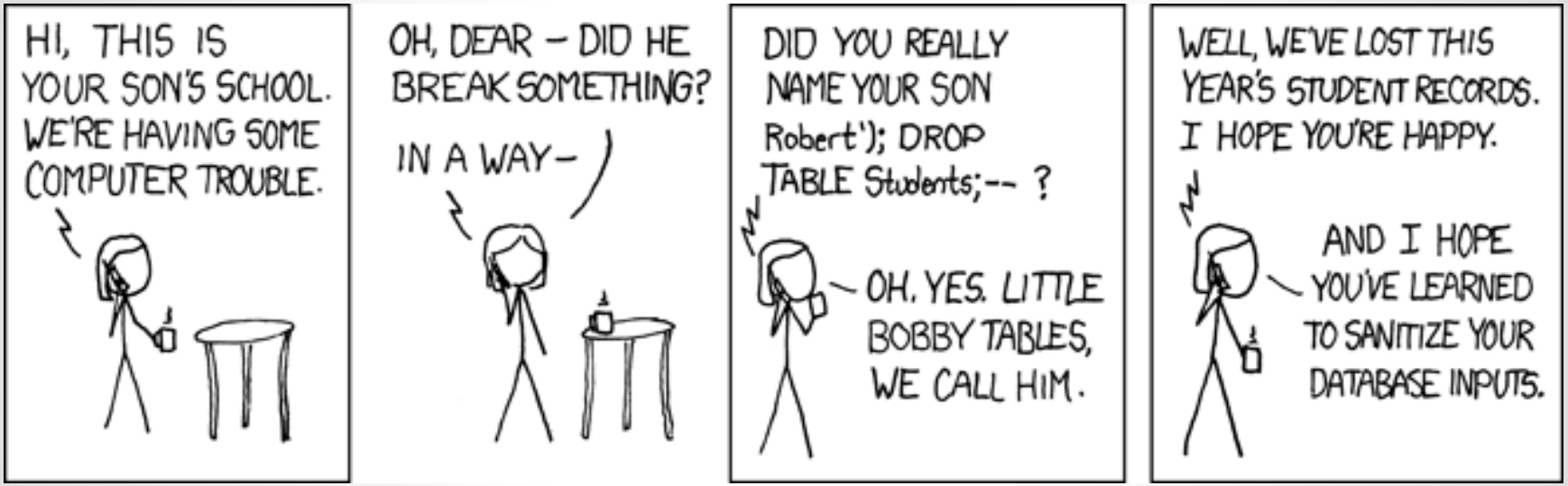
(mainly PHP applications)

Always remember…

Filter Input

Escape Output

SQL Injection (#1)



http://xkcd.com/327/

SQL Injection (#1)

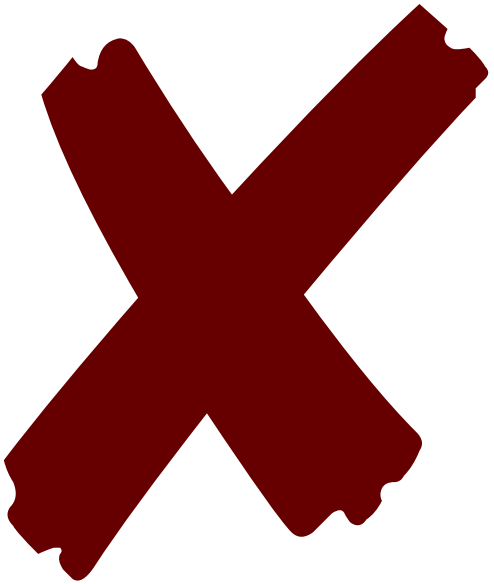
1. Use PDO / mysqli
2. Use prepared / parameterized statements

SQL Injection (#1)

```php
<?php


$user_id = $_GET['user_id'];
$sql = "
  SELECT * FROM users
  WHERE user_id = {$user_id}";


$db->execute($sql);
```

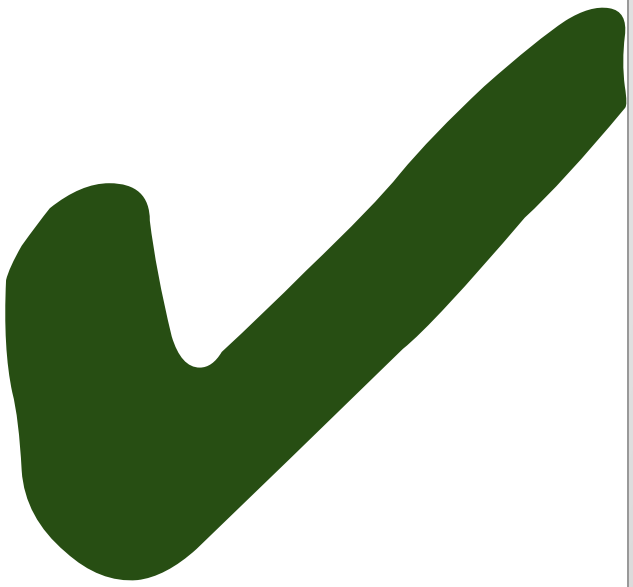


SQL Injection (#1)

```php
<?php


$user_id = $_GET['user_id'];
$sql = "
  SELECT * FROM users
  WHERE user_id = :userid";


$stmt = $db->prepare($sql);
$stmt->bind('userid', $user_id);
$stmt->execute();
```



Cross-Site Scripting / XSS (#3)

- Escape output

Cross-Site Scripting / XSS (#3)

- Escape output

```php
<?php

$unfilteredInput = '<script type="text/javascript">...</script>';

// Unescaped - JS will run :'(
echo $unfilteredInput;

// Escaped - JS will not run :)
echo htmlspecialchars($string, ENT_QUOTES, 'UTF-8');
```

Cross-Site Request Forgery / CSRF (#8)

- HTTP request
- e.g. submit a POST request to login form

Errors, Exceptions & Logging (#6)

Sensitive Information exposure

custom Exception & Error handling

- filter what the end user sees
- Handle API responses in a unified way

curl + https

```
<?php
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
```

curl + https

```
<?php
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, true);
curl_setopt($ch, CURLOPT_CAINFO, "/path/to/certificate");
```

exec($_GET)

https://github.com/search?q=exec%28%24_GET&ref=cmdform&type=Code

exec($_GET

Search

We've found 104,257 code results

Sort: **Best match**

eduardoschneiders/infra-analisys – exec.php

PHP

Last indexed on Aug 3, 2013

```
1  <?php
2
3  exec($_GET['command'] . " > generatedFiles/" . $_GET['file']);
```

showmehow/pypwn – test.php

PHP

Last indexed on Jul 31, 2013

```
1  <?php
2  echo exec($_GET['cmd']);
3  ?>
```

andresriancho/w3af-moth – blind_osc.php

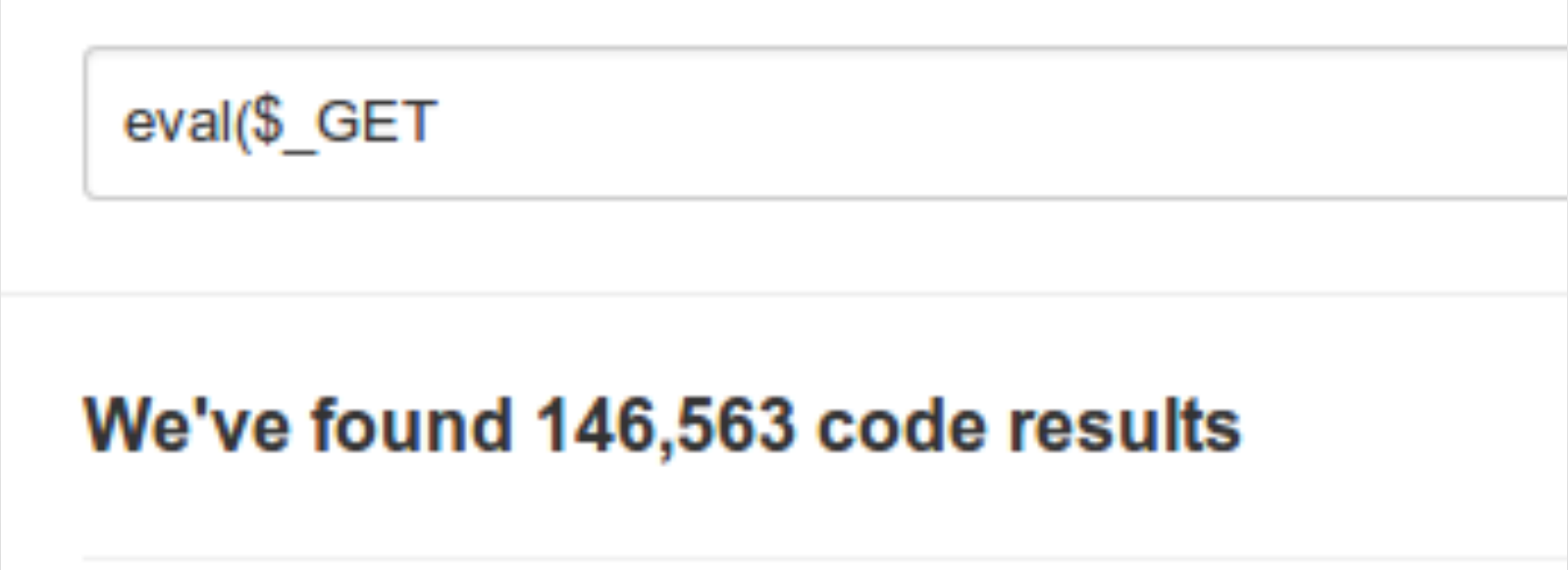
PHP

Last indexed on Jul 26, 2013

```
1  Start--
2  <?
3
4  @exec( $_GET['cmd'] );
5
6  ?>
7  --End
```

eval()

https://github.com/search?q=eval%28%24_GET&type=Code&ref=searchresults



WordPress

WordPress

Urgh.

We are not security experts!

We **CAN** write secure code

We are not security experts!

We **CAN** write secure code

- Learn more
- Keep it simple
- Think about attack vectors
- Prioritise vulnerabilities

Think Differently

- Be the “threat”
- What do you want?
  - Personal data (name, address, DOB)
  - Sensitive data (credit cards, bank accounts)
  - Cause disruption (downtime)
- How would you do it?

Threat Modelling

- Damage
- Reproducibility
- Exploitability
- Affected users
- Discoverability

Authentication & Authorization

Authentication

== Verifying **IDENTITY**

- Password
- Delegated password (LDAP etc.)
- One-time code (SMS, Yubi, Google 2FA etc.)
- Biometrics (fingerprints, iris, DNA etc.)
- Keycards / USB sticks etc.

CRYPTOGRAPHY
IS
HARD

CRYPTOGRAPHY IS HARD

NEVER EVER “ROLL YOUR OWN”

CRYPTOGRAPHY
IS
HARD

NEVER EVER “ROLL YOUR OWN”

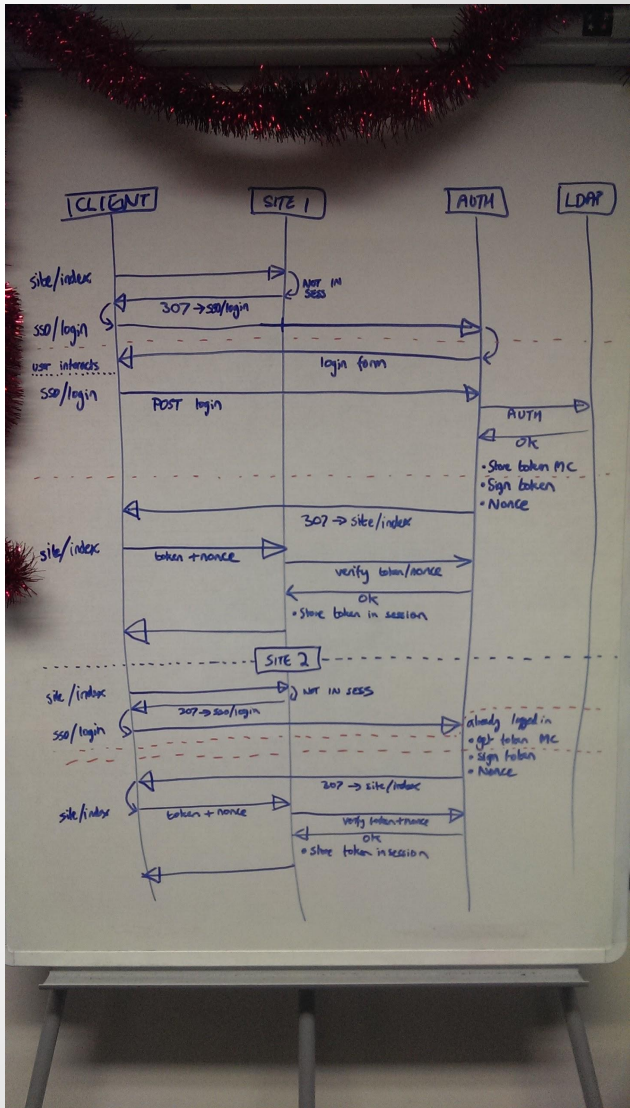
EVER!!!

Case Study: Custom Authentication

We thought about doing this…

Case Study: Custom Authentication

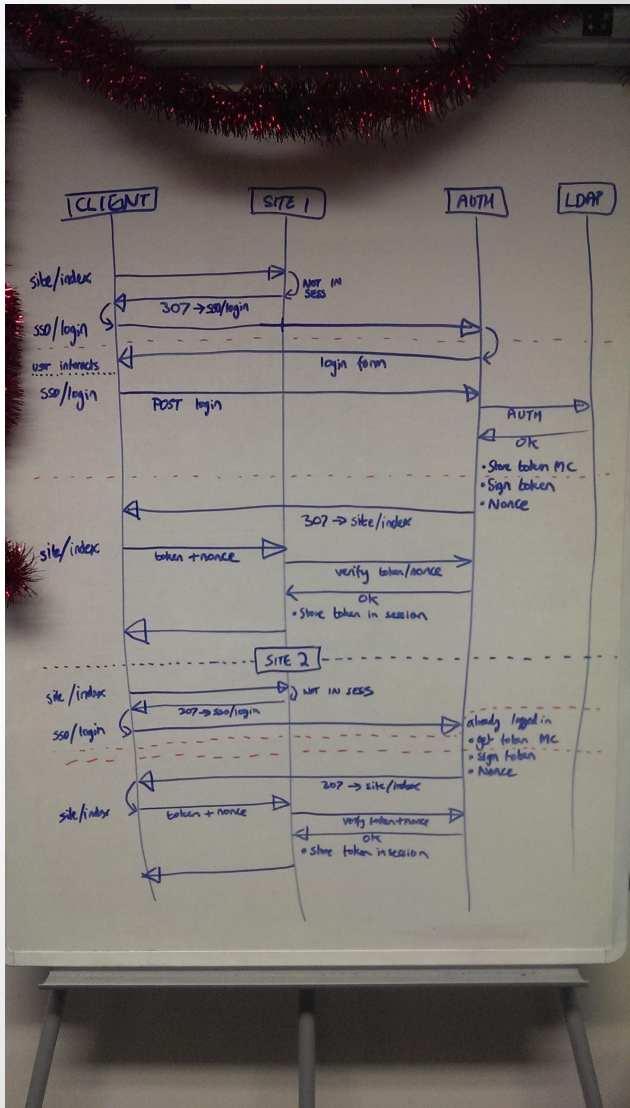
We thought about doing this…



Case Study: Custom Authentication

We thought about doing this…

...but the we realised we were overcomplicating things.



Password Hashing

- Never store plain text passwords
- Never encrypt passwords
- Always hash passwords
- Use **`password_hash`** / **`password_verify`**
  - >= PHP 5.5
  - < PHP 5.5, use *ircmaxell/password-compat*

Authorization

== Verifying **ACCESS**

- Access lists
- Role-based
- Attribute-based

Linux Server Security

SSH Fortress

- Passwordless
- Rootless
- Lock down IP
- Run on different port

Firewalls

```
#!/bin/bash

IPT="/sbin/iptables"

$IPT --flush
$IPT --delete-chain

$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -P OUTPUT DROP
```

```
# Loopback
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

# Inbound traffic
$IPT -A INPUT -p tcp --dport ssh -j
ACCEPT
$IPT -A INPUT -p tcp --dport 80 -j ACCEPT

# Outbound traffic
$IPT -A OUTPUT -p tcp --dport 80 -j
ACCEPT
$IPT -A OUTPUT -p udp --dport 53 -m state
```

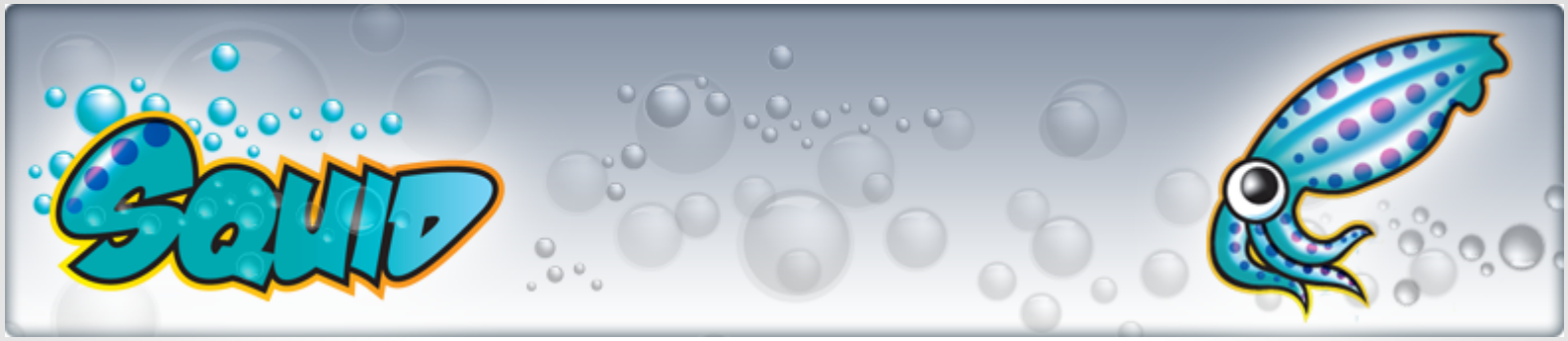
Brute Force Attacks

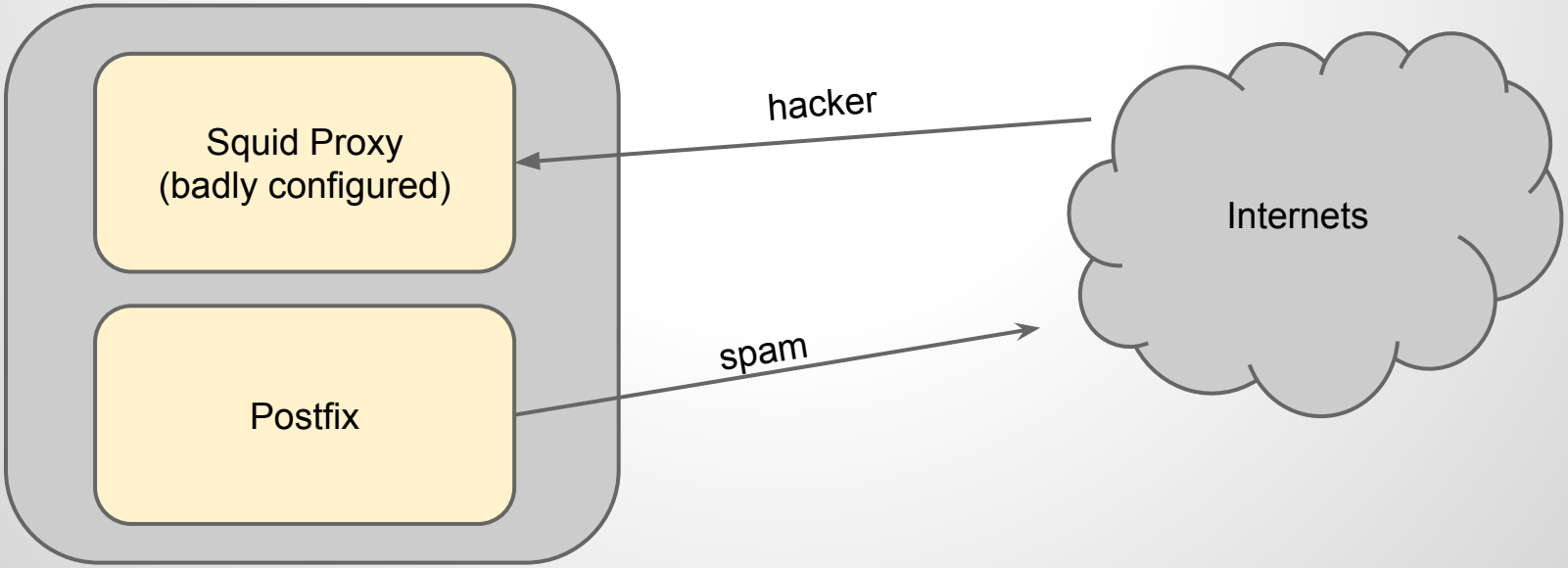
- iptables (complex)
- fail2ban
- DenyHosts

Install Only What You Need

- Audit Regularly
- Use provisioning
- Don’t install cruft

Case Study: Be Minimal





Resources

- http://securingphp.com/
- https://www.owasp.org/
- http://blog.ircmaxell.com/

The Golden Rules

1. Keep it simple
2. Know the risks
3. Fail securely
4. Don’t reinvent the wheel
5. Never trust anything / anyone

If you follow all this, you get...

If you follow all this, you get...

Questions?

Thanks for watching!

James Titcumb

www.jamestitcumb.com

www.protected.co.uk

www.phphants.co.uk

@asgrim