activities.r-adams.co.uk

Get yourself ready!

Every Day Security Testing

Richard Adams

Plan for today!

- Introduction
- Security concepts
- Basic techniques with activities
  - Headers
  - Fun with developer tools
  - IDOR (aka URL manipulation)
  - XSS
  - SQL Injection
- Tools
- Wrap up
- Game (if we have time)

Who am I?

Testing & Quality Professional

- Started testing in 2008
- Had a variety of roles:
  - From Games Designer
  - To SW Developer
  - Mainly test, QA or QE variants
- Was a “Cyber Champion”
- Ministry of Testing Edinburgh Chapter Lead

"The epic tale of my pilgrimage to obtain these relics should be written on a scroll or something!"

LinkedIn

Who am I?

Am I an expert?

However I:

- Got some training in security
- Extended my testing toolkit
- Unlocked new issues to find
- Had fun with it
- Want to share that with my peers

Why does security matter?

A thought from Flick

"Will this world meet its reckoning, beheld only by the lonely eyes of the security bug?"

Security for testers

Security is part of **quality**	Quality is **important to us** as testers	**Security is important for testers!**
Risks are a huge part of testing & quality	Security can be a source of **major risks** with **critical** impacts	**Security is important for testers!**
People talk of **testing throughout the SDLC**	And we typically defer **security** testing until **the end** (or never)	**Security is important for testers!**

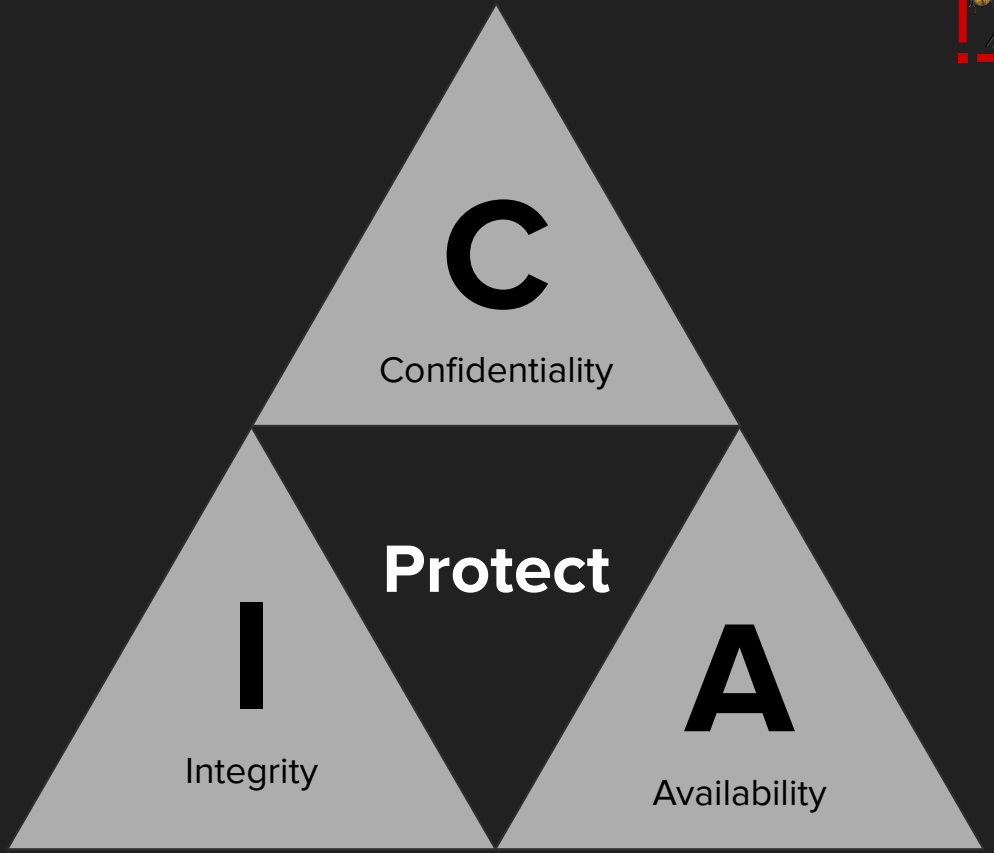
Security Concepts

CIA Triad

Popular model in cybersecurity.

- Confidentiality
- Integrity
- Availability

Can be a useful heuristic for risk based test strategies!



Authentication Vs Authorization

Authentication

“Proving that I am who I say I am.”

Logins / Keys / Tokens / Certificates etc

Authorization

“Having permission to perform actions or see the data I want.”

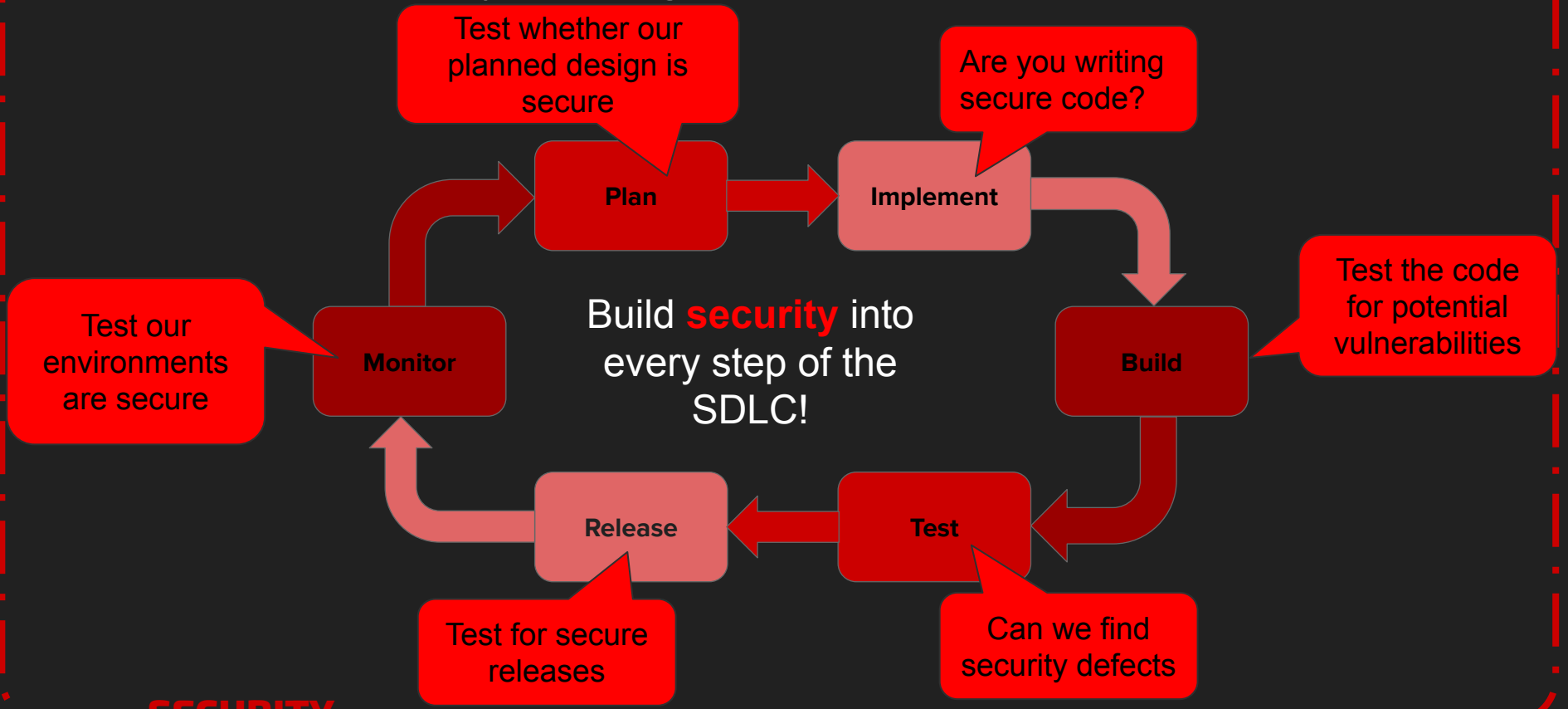
Admin / User / Viewer / Editor etc

Vulnerabilities, Exploits and Attacks

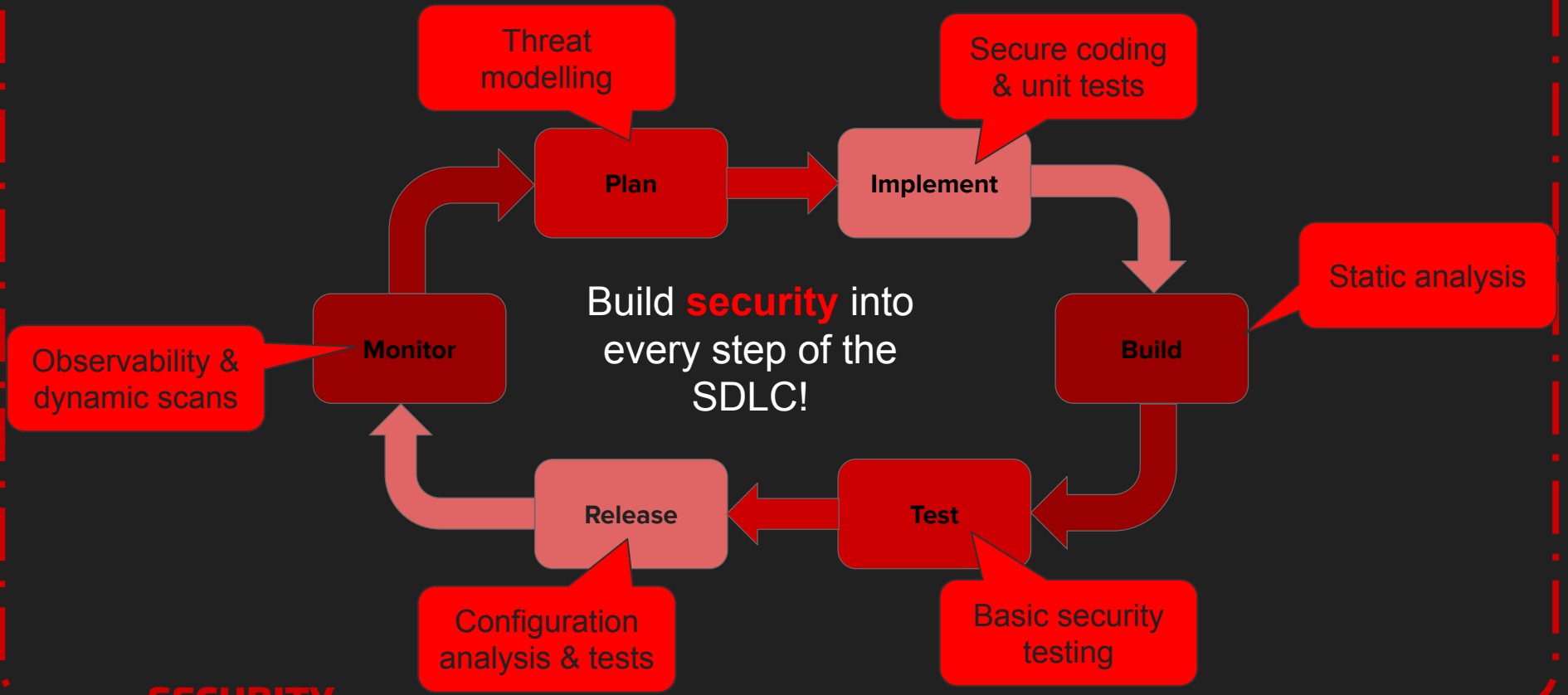
Vulnerability	A mistake is made within code or the environment, creating a vulnerability.	**The Flaw**	**Defect**
Exploit	A tool, code, or method used to take advantage of that vulnerability.	**The Means**	**Test Technique**
Attack	An attacker uses the exploit to perform a malicious action and make an attack.	**The Action**	**Test Execution**

Security Testing

How does security testing fit into SDLC?



Carrying out security testing throughout SDLC



Basic Techniques

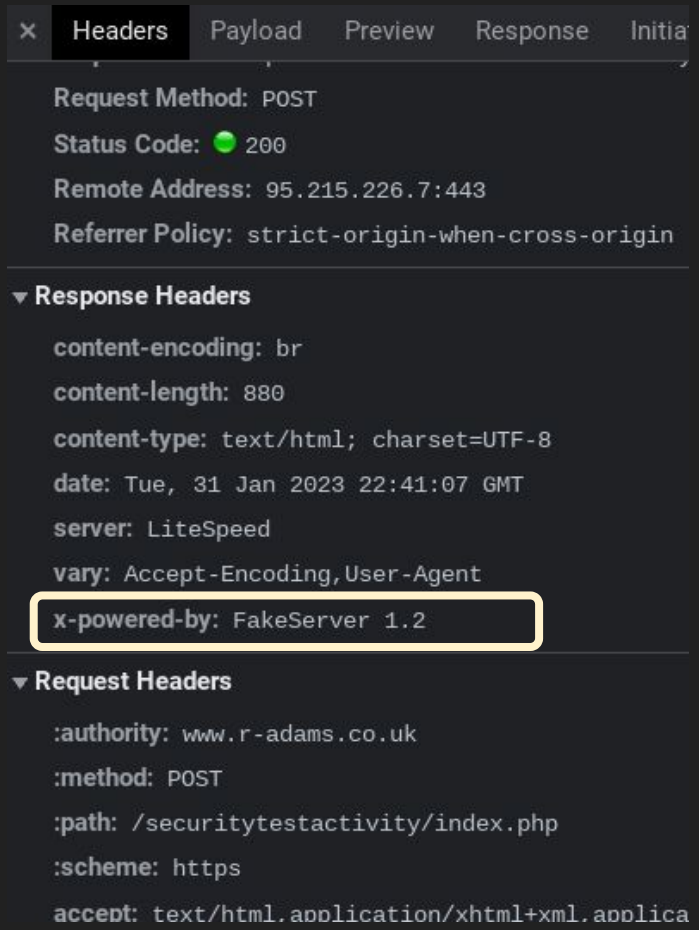
Inspecting Headers

Headers

What do the headers tell you about the site?

Do they tell you about the site's vulnerabilities?

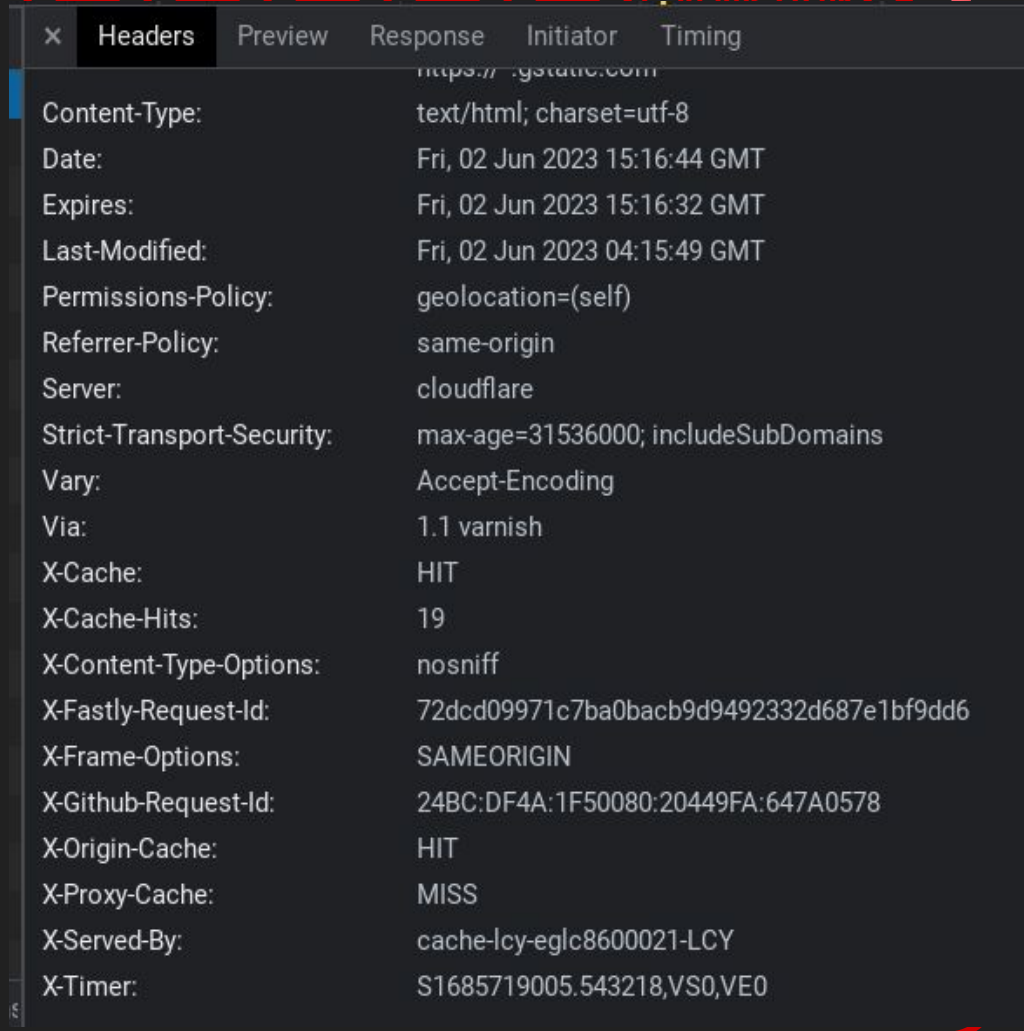
Example: Google search for "FakeServer 1.2 vulnerabilities"



Headers

What do the headers tell you about the site?

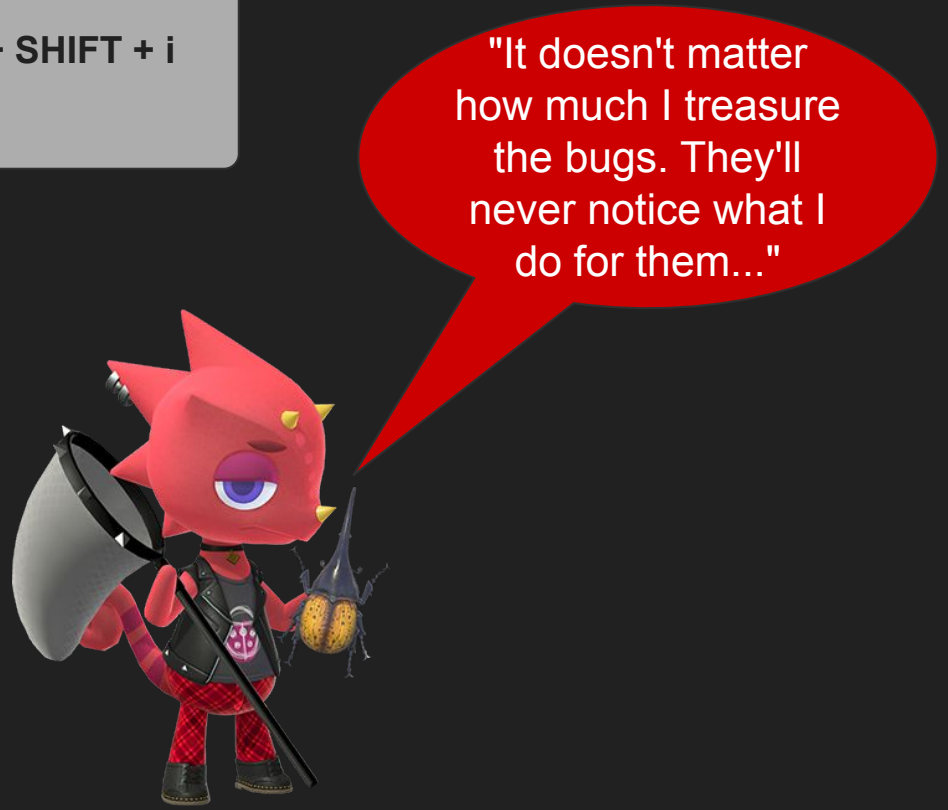
There are also headers that are good to set to make your application secure.



Exercise: Headers

1. Visit activities.r-adams.co.uk
2. Open the Developer Tools
3. Use the network tab
4. Access “Advertising My Insecurities”
5. Provide two answers:
   a. The framework being used.
   b. A vulnerability with a CVSS score of 9.8

CTRL + SHIFT + i
F12



Hint: NIST have a National Vulnerability Database with loads of issues called CVEs.

Insecure Direct Object References

Testing the URL

What is a URL?

In the browser address bar you will see the URL for your webpage. It is the links that we share.

Within it contains lots of key parts:

The protocol
Ideally HTTPS for security!

Port
Omitted if standard (80 or 443)

Parameters
Sometimes param=value, sometimes /order/value/

`http://mysite.com:8080/userPanel/order.php?id=1234`

Domain name
May have a subdomain, like sales.mysite.com

Path

Testing a URL: Example

If there's a "userPanel", does that mean there's "adminPanel" ?

"375" is an order number/ID. Can I change it?

Out of scope

https://mysite.com/userPanel/orders/375/view

"view" is an action. Can I think of other actions? "/edit" or "/delete"

Can I go a step bolder and mix a different ID & a different action?

Can I do something I shouldn't?

Exercise: IDOR & Secrets of a Tester

1. Visit **activities.r-adams.co.uk**
2. Go to “**Secrets of a Tester**”
3. Log in with the default credentials
4. Explore the site for a couple of minutes
5. Modify the URL to solve the challenges

Session based fun

Even without experimenting with the URL itself, we can play with accessing URL across different “sessions”.

We can try directly accessing URLs:

- Save a bookmark then access it later.
- Copy-paste URLs between browsers when using different user accounts.

Experiment with different sessions and account types:

- Admin vs user account
- User account vs not logged in

API Requests

Don’t be limited to just your browser!

1. Use the “Developer Tools” to see all the requests.
   - Tools like Wireshark work for desktop applications or services
2. Use a REST client to send the same requests and see what you get back.
   - Ask a colleague for help if unsure!
3. Look at their parameters and the responses. Use your tester spidey-senses.
4. Start to tinker with these requests!
5. Try switching authentication to use the same request with different users

IDOR Recap

- Insecure Direct Object Reference is a vulnerability where you can access objects you shouldn’t be able to.
- We can test for it using URL Manipulation (testing the URL).
- Use multiple browsers to test with different sessions.
- Can also be performed using direct API requests.

Question

Is what you see on a web page fixed or can you modify it?

MINISTRY OF TESTING

Learn Events Trends Radar Club Pro Search...

Software testing news – Issue 485: Testers meet to praise Richard

Read a roundup of comments, love and adoration of Rich and his testing super powers. Discover how you too can be (almost) as amazing as Rich!

Newsletter Issue

How to manage your photos of Rich

Article

How much do you love security testing?

Club Topic

This Week in Testing - Episode 55 - 09 August, 2024

TWiT Episode

🤖 New Course: Everyday security testing: A practical guide to getting started

News

What do software testers need to know about security?

Security is a number one priority

Tips for SQL injection

Sometimes you can't trust a screenshot

Elements Console Sources Network

```
Richard"
::after
</a>
</h2>
</div>
</div>
<div class="summary">
" Read a roundup of comments, love and adoration of Rich and his testing super powers. Discover how you too can be (almost) as amazing as Rich! "
</div>
</div>
<div class="card-footer py-1 z-index-1">…</div>
</div>
</div>
<div class="col-12 col-lg-6">
<div class="row pb-4">flex
<div class="col-6">…</div>
<div class="col-6">
<div class="card shadow h-100 summary-card">container flex
<div class="overflow-hidden position-relative">…</div>
<div class="card-body">
<div class="d-flex">flex == $0
<div class="flex-grow-1">
<h2 class="title mb-0">
<a class="stretched-link" target="_top" href="/club-topics/what-automated-testing-tools-will-dominate-in-the-near-future">
"How much do you love security testing? "
::after
</a>
</h2>
```

div.row.pb-4 div.col-6 div.card.shadow.h-100.summary-card div.card-body div.d-flex

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls

Console Issues What's new Search

Highlights from the Chrome 127 update

Enhanced 'Never pause here'

Browser Developer Tools

- Most modern browsers provide tools to modify the HTML, CSS & javascript etc in the displayed page.
- This doesn’t affect anyone else or persist - just your interaction.
- This lets developers try changes without rebuilding etc.
  - Alter styling or layout.
  - Change attributes on elements
  - Try and fix the javascript function being called.

Question

Is the validation in a web UI fixed?

Browser Developer Tools

Developers… and malicious users… and curious testers can:

- Change which fields are disabled or enabled
- Make a required field optional
- Turn a numeric field into a text field
- Increase the maximum length of fields
- Modify a hidden field

Demo

http://testingchallenges.thetestingmap.org/challenge2.php

Exercise: Banana Fan Group Meetup

1. Visit activities.r-adams.co.uk
2. Access Banana Fan Group Meetup
3. Use the browser developer tools to solve the challenges

Injection

20th Century Fox Television

Injection - How does it work?

Cross Site Scripting (XSS)

Cross Site Scripting

Simply put, try and get the website to run a script (Javascript) that you provide.

Usually targeting other users of the website.

Attackers will do clever things:

- Steal cookies
- Phishing
- Reading user data

Cross Site Scripting

Usually in testing this is getting an alert to show

This may be achieved with the following test data:

```
<script>alert('Woohoo I found a security bug!')</script>
```

OR: `<script>alert(1)</script>`

Exercise: Cross Site Social Part I

```
<script>alert(1)</script>
```

1. Visit activities.r-adams.co.uk
2. Access “Cross Site Social”
3. Update the Bio with the javascript test data.
4. Click View Profile and see if you get the alert!
5. Optionally try the same test data in the other fields!

Note that the **Resources** section contains handy test data.

Injecting XSS

Injecting XSS

Sometimes we won't be able to use such simple test data:

- Maybe there's protection against <script> tags
- Maybe it wouldn't be valid HTML

```
<img src="http://site.co/my.png<script>alert(1)</script>" />

<a href="http://site.co<script>alert(1)</script>">Link</a>
```

We need a different way to run our Javascript

Injecting XSS: Another method

Did you know that HTML elements have events that can run Javascript?

```
<img src="URL" onload="hideLoadingPlaceholder()" />

<img src="URL" onmouseover="greyOutOtherImage()" />

<img src="URL" onclick="showConfetti()" />
```

Injecting XSS: Another method

We can use events to inject XSS (i.e. an alert box):

```
<img src="URL" onload="alert(1)" />

<a href="URL" onmouseover="alert(2)" >My Website</a>

<input name="email" type="text" onchange="alert(3)" />
```

Injecting XSS: Another method

Let's say we're testing providing a URL for an image:

```
<img src="YOUR_INPUT" />
```

And we want the end result of:

```
<img src="VALID_URL" onload="alert(1)" />
```

Our test input is everything we want between the first and last quotes:

```
<img src="VALID_URL" onload="alert(1)" />
```

The attack here is:

```
https://r-adams.co.uk/Flick.png" onload="alert(1)
```

Exercise: Cross Site Social Part II

1. Revisit “Cross Site Social”
2. Try to make attacks for the image & website.
3. Optionally use the “publish” functionality and share your profile with your table.

SQL Injection

SQL Injection

A famous type of attack

- Inject bits of SQL into input.
- Change:
  - A filter / WHERE clause
  - Which fields are updated
  - And more…
- Can be combined with modifying the UI

Worked Example 1

SQL Injection

Let's say that I have an input box asking for your display name.

Display Name: Rich Adams

The code to insert your value might look like this:

```
sql = "UPDATE users SET displayName='{name}' WHERE id={sessionId}"
```

The actual SQL query performed might be:

```
"UPDATE users SET displayName='Rich Adams' WHERE id=27"
```

SQL Injection

Now let's look at what happens with input with quotes (escape characters)

Display Name: Rich's Inner Voice

The actual SQL query performed might be:

```
"UPDATE users SET displayName='Rich's Inner Voice' WHERE id=27"
```

This would give an error as that single quote means my displayName is 'Rich' and the rest is invalid SQL.

SQL Injection

Now let's try being malicious

Display Name: Rich', role='admin

This will then change the behaviour of the SQL query:

```
"UPDATE users SET displayName='Rich', role='admin' WHERE id=27"
```

That is perfectly valid SQL! (assuming `role` is the name of a field).

Entering that value would make me an admin!

Worked Example 2

SQL Injection

In our next scenario we have a filter:

Category Filter: Book

SQL Injection

In our next scenario we have a filter:

Category Filter: Book

The code to run the filter might look like this:

```
sql = "SELECT * FROM products WHERE category='{category}' AND
released=true"
```

SQL Injection

In our next scenario we have a filter:

Category Filter: Book

The code to run the filter might look like this:

```
sql = "SELECT * FROM products WHERE category='{category}' AND released=true"
```

The actual SQL query performed might be:

```
"SELECT * FROM products WHERE category='Book' AND released=true"
```

SQL Injection

Now let’s try an injection

Category Filter: Book’ OR 1>0 -- comment

SQL Injection

Now let’s try an injection

Category Filter: Book’ OR 1>0 -- comment

This would make the query:

```
“SELECT * FROM products WHERE category=’Book’ OR 1>0 -- comment’ AND released=true”
```

SQL Injection

Now let's try an injection

Category Filter: Book' OR 1>0 -- comment

This would make the query:

```
"SELECT * FROM products WHERE category='Book' OR 1>0 -- comment' AND released=true"
```

The use of -- here is a comment, so let's remove the comment:

```
"SELECT * FROM products WHERE category='Book' OR 1>0"
```

Exercise: SQL Injection

1. Visit “Sea Quell”
2. Register (get your sea legs)
3. Try to perform the three different attacks.
4. Ask if you need help!

Tools

SAST & DAST

SAST & DAST

Static Application Security Tools

- Tests the source code or build output
- Some tools check dependencies for known vulnerabilities (SCA)
- Some tools check your code for design flaws
- Great for including in pipelines!

Dynamic Application Security Tools

- Tests the running application
- Some tools scan and look for configuration issues in your & application environment
- Some tools watch you, analysing your requests & the responses.
    - Can then make further requests (e.g. injection, directory traversal & more)

ZAP

Zed Attack Proxy

- Available for free
- https://www.zaproxy.org/
- Use it during standard testing.
- Monitor the pages to spot vulnerabilities.
- Attack mode will proactively make attacks.
- An alternative is Burp Suite
- Also OWASP Pen Test Kit plugin

WARNING: Don’t use attack mode in production!

WARNING: It is not user friendly.

File Edit View Analyse Report Tools Import Export Online Help

Demo Session - DemoVideo - ZAP 2.17.0

Standard Mode

Quick Start | Requ... | Respon... | Reques... | Client Details

Manual Explore

ZAP by Checkmarx

This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore: https://activities.r-adams.co.uk/playground/xss.php Select...

Enable HUD:

Explore your application: Launch Browser Firefox

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.

Hist... | Search | Alerts | Output | Active Scan

Filter: OFF Export

ID	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags

First we provide the website we're testing

OWASP Penetration Testing Kit Demo

XSS Playground Page

OWASP Penetration Testing Toolkit (Chrome Store)

Wrap Up

Test Data

As you'll have seen, security testing can begin with test data.

For SQL, which one depends on your database tech & SQL coding style:

- ' OR 1 > 0 -- comment
- " OR 1 > 0 -- comment
- " OR 1 > 0 # comment

I **LOVE** a cheat sheet and my favourites places are:

- OWASP Penetration Testing Kit
- Big List of Naughty Strings
- PortSwigger's Web Security Academy
- My cheat sheet of cheat sheets

Day-to-Day Security Testing

You could set aside time in a sprint dedicated to security testing.

Instead I would look at my next user story:

- Add happy path and known incorrect paths.
- Test with some problematic input: {0}, unicode, RTL, null char etc.
  - Including some injection from a cheat sheet.
- Add some exploration beyond the UI to my exploratory charters.

And if I find a security bug, I log it.

If it is critical, as is often the case with security, I notify “someone”.

Resources

Learn

Ministry of Testing Security Course
ministryoftesting.com/courses/everyday-security-testing-a-practical-guide-to-getting-started

Ministry of Testing Dev Tools Course
ministryoftesting.com/courses/a-software-tester-s-guide-to-chrome-devtools

OWASP Web Security Testing Guide
owasp.org/www-project-web-security-testing-guide

Practice

OWASP Juice Shop
owasp.org/www-project-juice-shop

Google Gruyere
google-gruyere.appspot.com

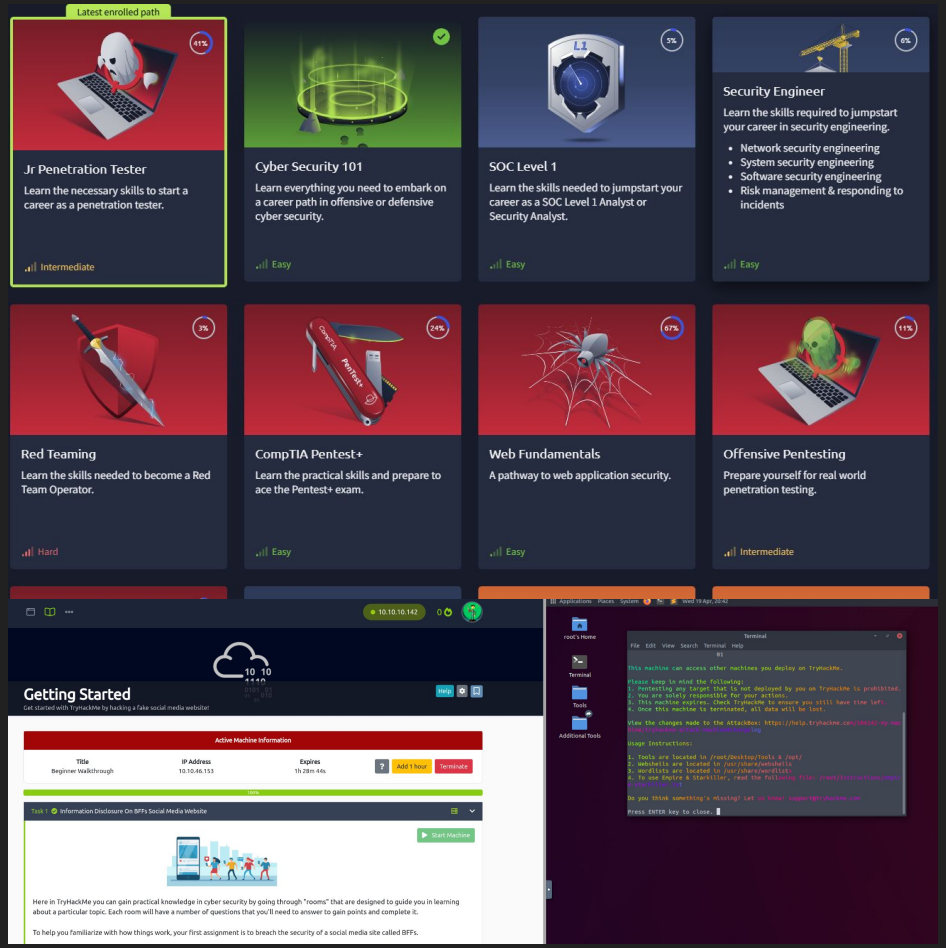
More on the Ministry of Testing site:
ministryoftesting.com/articles/websites-to-practice-testing

Resources

Learn & Practice

TryHackMe
tryhackme.com

Port Swigger Academy
portswigger.net/web-security



Keep Learning

It is sheer hubris to think that we could ever learn everything about security bugs.

Final Activity

Tea Virus

- Story:
  - The Tea Virus is set to be unleashed upon the population.
  - An agent was sent to destroy it.
  - We received confirmation that they installed the malware but nothing since.
  - Your mission is to find what the agent has done and execute the malware.

- Uses a mix of techniques:
  - Developer tools
  - SQL injection
  - XSS

Capture The Flag: We Are Angry

1. Access the admin panel without the admin login
2. Update the site name as a user
3. Update the biography of another user
4. View people's bios without logging in
5. Bypass the restriction on only admins posting different categories.
6. Post a message with a category not intended to be available.
7. As a user, see any messages in the "secret" category.
8. Submit a message without any actual message.
9. Perform a reflected XSS attack on the messages page.
10. Successfully add a new user when you aren't logged in as an administrator level user.

Thanks!


How did you enjoy this Bug-Off, our regular celebration of bugs? Were you spiritually enriched?