School of Computing
FACULTY OF ENGINEERING & PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

COMP3911 Secure Computing

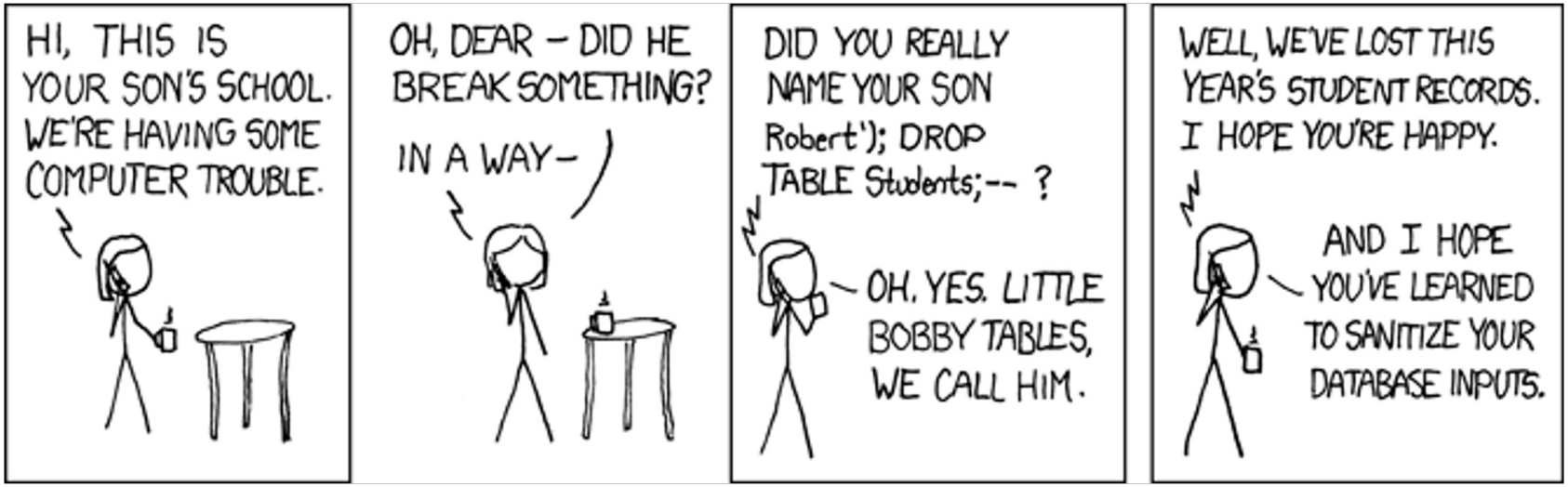
15: Web Vulnerabilities

Nick Efford

https://comp3911.info

Already Covered: SQL Injection

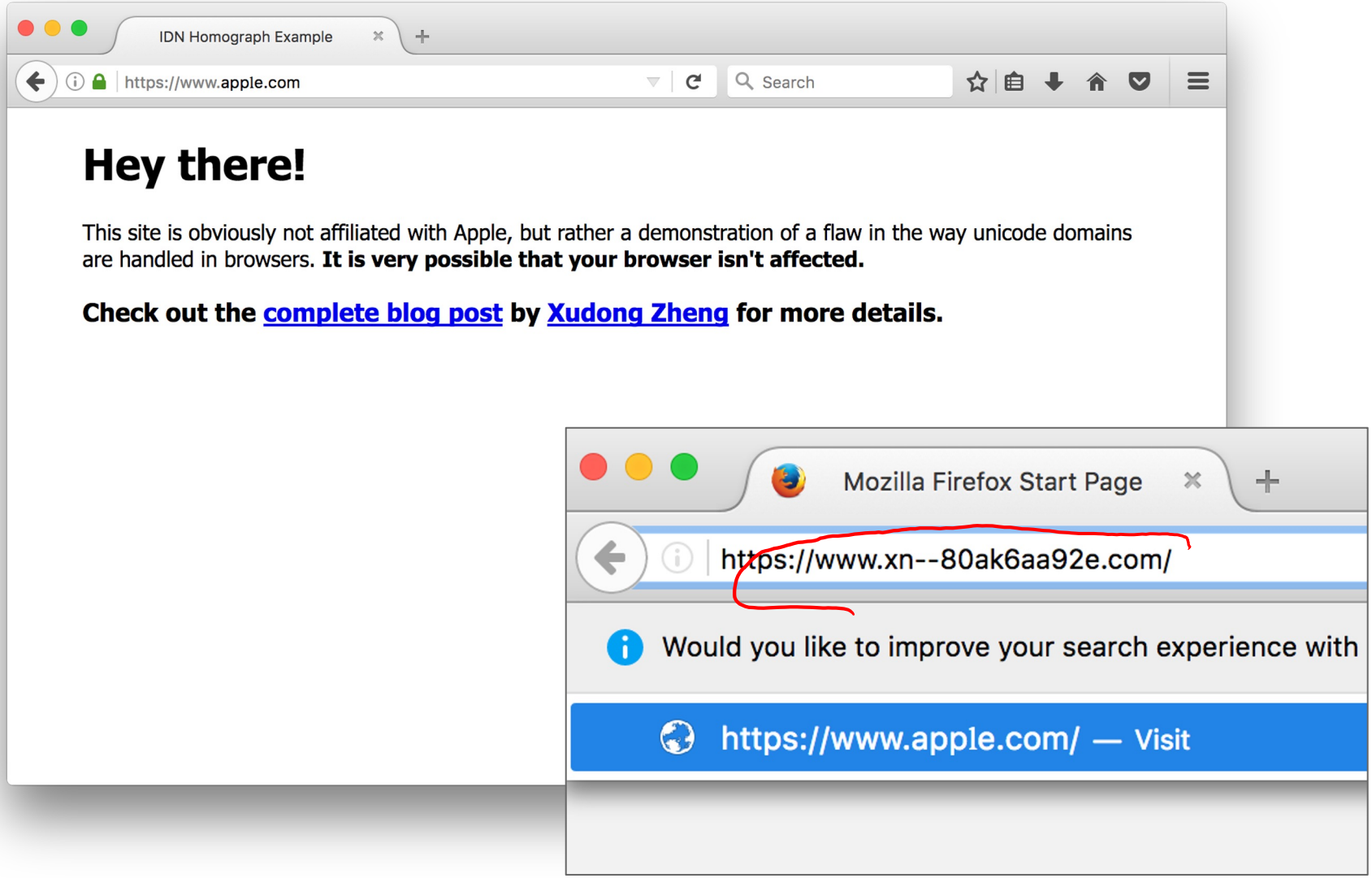
UNIVERSITY OF LEEDS



https://xkcd.com/327/

Already Covered: Homograph Attacks

UNIVERSITY OF LEEDS



Already Covered: Directory Traversal

UNIVERSITY OF LEEDS

Security

192

Dishwasher has directory traversal bug

Thanks a Miele-on for making everything dangerous, Internet of Things firmware slackers

By Richard Chirgwin 26 Mar 2017 at 23:08 SHARE ▼

Repeated use of `../` to step out of directory tree used by a web server

Possibly concealed by encoding characters (e.g., `%2e%2e%f`)

Objectives

UNIVERSITY OF LEEDS

- To review web architecture briefly
- To explore some of the ways in which web applications can be attacked (beyond those covered last time)
- To consider how we defend against these attacks

Web Vulnerabilities

UNIVERSITY OF LEEDS

- Web applications are typically implemented in Java, C#, PHP, Python, Ruby, etc
- This protects us to a large degree from buffer overruns and similar low-level vulnerabilities (in the app, at least)
- … but there are many other ways of attacking such systems at a higher level!
- Ubiquity of web-based systems compounds the problem; the **attack surface** is very large
- Again, the issue is often one of **input validation**

Typical Goals of Web Attacks

UNIVERSITY OF LEEDS

- Tampering with data
  - Web page defacement
  - Alteration of data in backend DB
- Information disclosure
  - Active authentication tokens
  - Stored user credentials
  - Credit card details or other sensitive info
- Denial of service
- Elevation of privilege

Web Architecture

UNIVERSITY OF LEEDS

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en
User-Agent: Mozilla/5.0 ...
```

HTTP or HTTPS

client

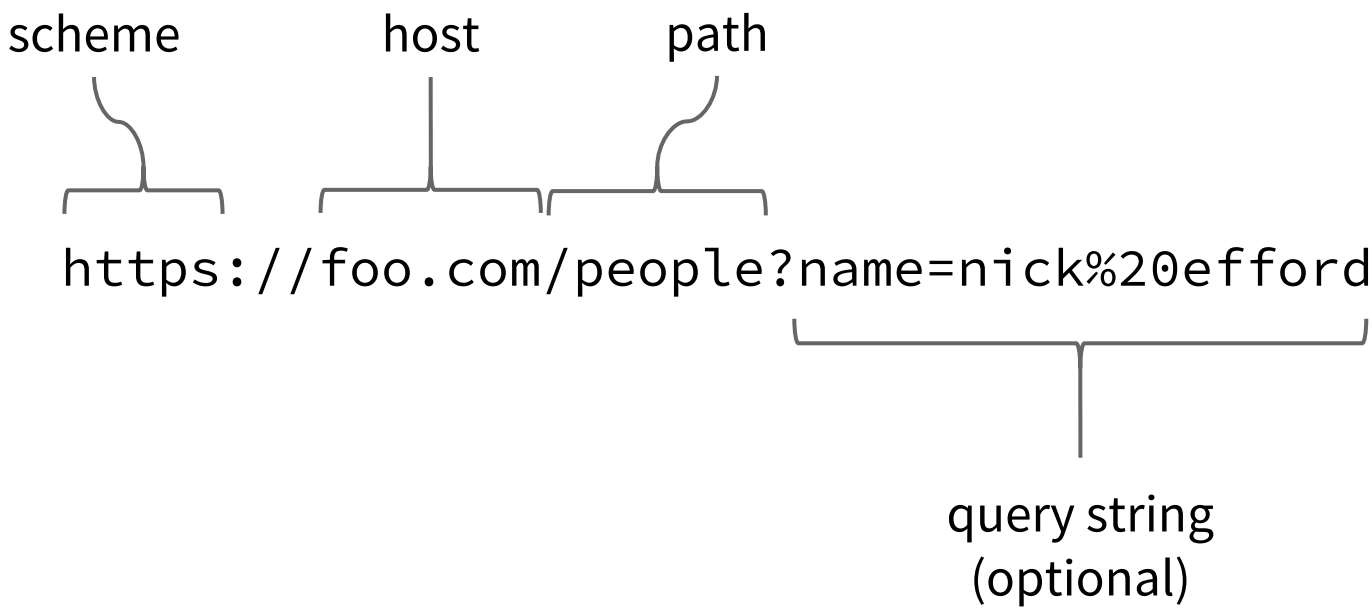
needn't be a browser!

server

```
HTTP/1.1 200 OK
Date: Mon, 2 Sep 2019 13:05:34 GMT
Content-Type: text/html; charset=UTF-8
...

<html>
...
```

URLs



Query string uses `key=value` format, concatenates multiple parameters with `&` and uses **URL encoding** for special characters (so spaces become `%20`, etc)

GET vs POST

UNIVERSITY OF LEEDS

- Confusingly named; both can send data to server! (e.g., from a form on a web page)
- GET sends data as a query string, POST sends it as payload of the request (URL-encoded in both cases)
- POST is intended for operations that have a side-effect, changing application state on the server
  - Browsers will allow resubmission of GET requests via page reload, but will warn if you do this for POST

Sessions

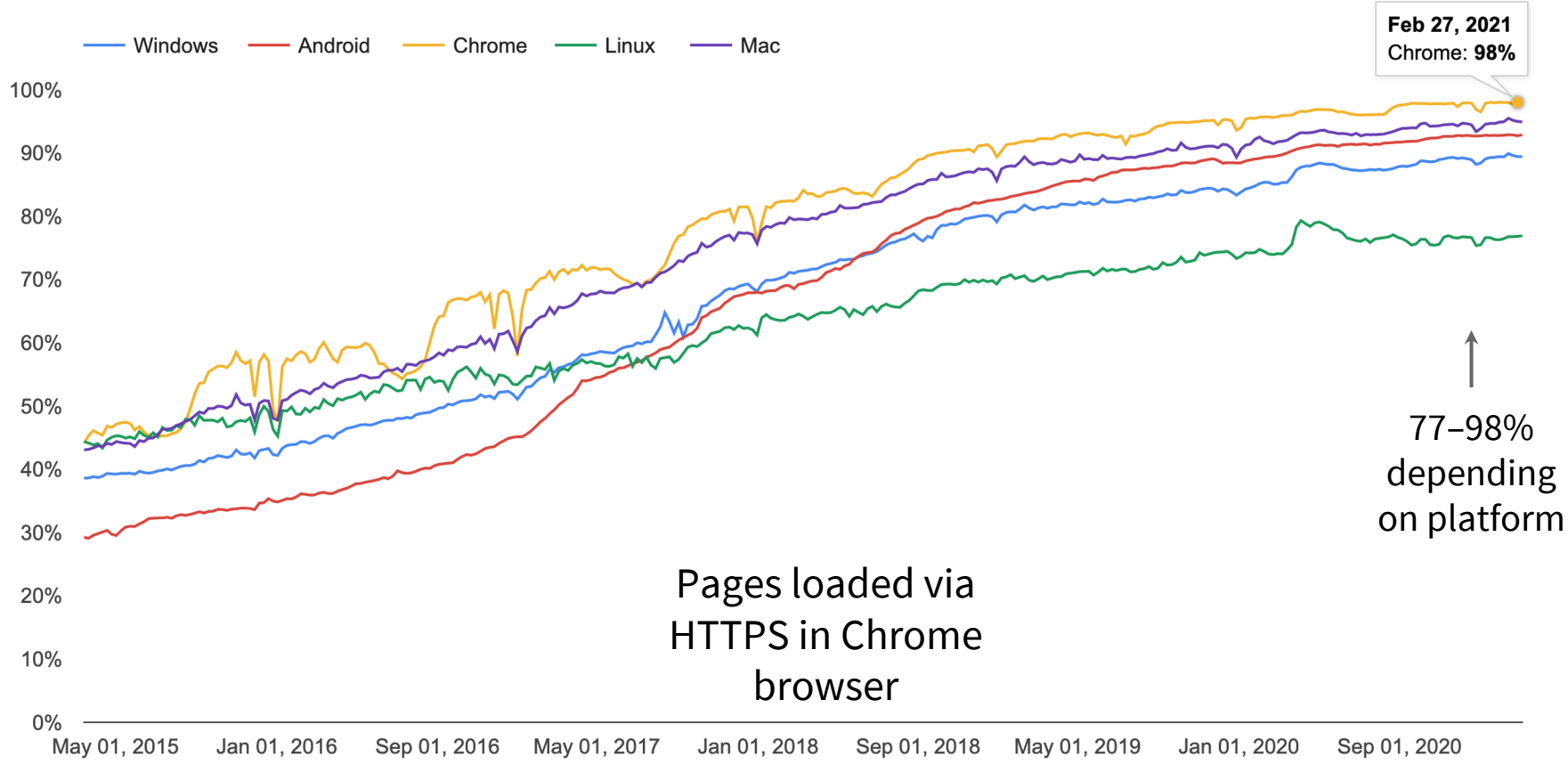
UNIVERSITY OF LEEDS

- HTTP is **stateless**, so we need an additional mechanism to recognise a sequence of interactions from a user
- A **session** is represented by a persistent token (**session ID**), stored client-side as a **cookie** and presented to server
- Session IDs can be for unauthenticated users (e.g., to track shopping carts) or can be issued after authentication
- Sessions can have associated state, persisted locally (e.g., via cookies, HTML5 web storage) or on the server
    - Server storage is more secure!

Attacking the Connection

UNIVERSITY OF LEEDS

Fortunately, HTTPS is becoming the norm
(60.9% of 'Alexa Top 1 Million' sites in March 2020)



Attacking the Connection

UNIVERSITY OF LEEDS

But HTTPS is not perfect!

- Crypto issues (e.g., DROWN attack)
- Lack of **HSTS** (HTTPS Strict Transport Security)
  - Allows initial insecure request via HTTP
  - Attacker could hijack this via DNS spoofing and become a man-in-the-middle
- Misuse of certificates
  - 2015: Lenovo's Superfish adware
  - 2018: Sennheiser HeadSetup

Attacking the Server

UNIVERSITY OF LEEDS

Already discussed:

- SQL & command injection
- Directory traversal

Covered today:

- Open redirects
- URL jumping
- Weak authentication
- Flawed session management
- Malicious XML payloads

Open Redirects

UNIVERSITY OF LEEDS

- Found in web apps that require you to authenticate before giving you the page you wanted to visit
- You get sent to a login page but the page you wanted is included in query string, to facilitate a redirect after login:

  `www.foo.com/login?page=myaccount`

- What if attacker gave you a link like this?

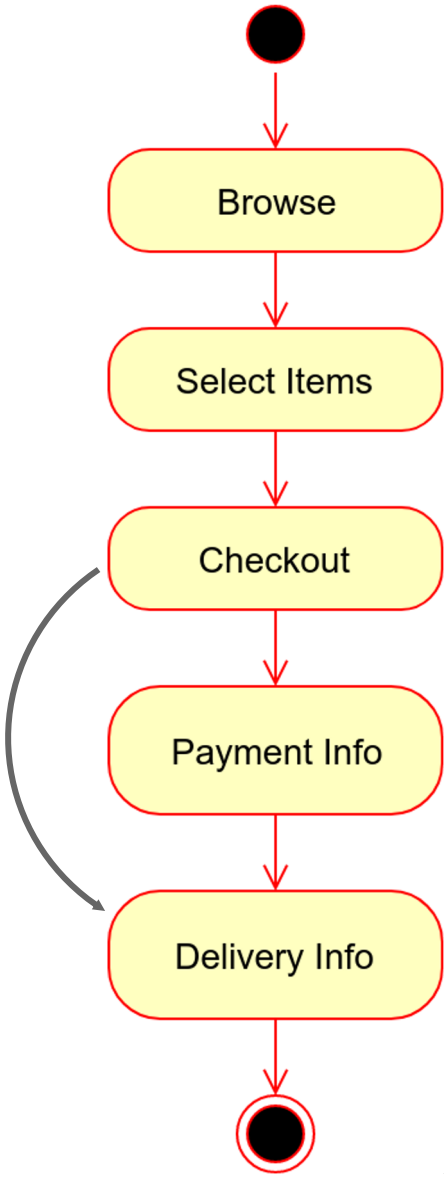
  `www.foo.com/login?page=http://www.evil.com`

- Simple fix is to validate the redirect parameter!

URL Jumping

UNIVERSITY OF LEEDS

e-commerce applications have an expected flow from page to page…



Can store last visited page using hidden form field, query parameter, cookie or HTTP Referer field – but these can be spoofed

Can attacker skip a step?

Best approach is to track the flow with session data stored on server

Typical Authentication Errors

UNIVERSITY OF LEEDS

- Not using HTTPS
- Using HTTP ‘basic’ or ‘digest’ auth mechanisms
- Disclosing whether a username is valid or not
- Allowing users to have weak passwords
- Allowing brute force attacks
- Not storing passwords securely on server
- Default accounts
- Hard-coded credentials

Secure Password Management

UNIVERSITY OF LEEDS

- Hash, don't encrypt
- Use a strong hash function (SHA-256 or bigger)
- Use plenty of random salt
- Make the algorithm very slow (e.g., iterating many times)
- Notify users promptly when you are breached!

See Exercise 11

Session ID Dangers

UNIVERSITY OF LEEDS

- Prediction by attacker
- Theft
  - Via attacks against victim's computer
  - Via cross-site scripting (see later)
  - Via packet sniffing
  - If exposed by URL rewriting

Remember: session ID allows attacker to impersonate the user!

Exposure Via URL Rewriting

UNIVERSITY OF LEEDS

- Session IDs are typically stored in cookies – but what if browser doesn't support cookies, or they have been disabled?
- As a workaround, web app might rewrite the URL so that session ID is included as a parameter:

  `/shopping/products?sessionid=c0fa96b3`

- But now session ID is more widely exposed!
  - Will be visible in user's browser history, could be cached by web proxies, etc

Managing Sessions Securely

UNIVERSITY OF LEEDS

- Generate IDs with a cryptographic PRNG
- Never reuse an ID
- Use HTTPS and enable `Secure` cookie option
- Establish a maximum session lifetime (e.g., 1 or 2 hours)
- Invalidate sessions that have been idle for a while
- Make it easy to log out & clear session explicitly
- Limit session concurrency
- Minimise use of client for storing session state, and encrypt anything stored client-side

Malicious XML Payloads

UNIVERSITY OF LEEDS

- Traditional web services involve the exchange of messages represented as XML documents
- An XML document sent as request to the server must be parsed – creating opportunities to exploit weaknesses in the parser implementation
- Example: exponential entity expansion (XXE)

‘Billion Laughs’ Attack

UNIVERSITY OF LEEDS

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
 <!ELEMENT lolz (#PCDATA)>
 <!ENTITY lol "lol">
 <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
 <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
 <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
 <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
 <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
 <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
 <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
 <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
 <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Why is this so named?

What type of threat does it represent?

Summary

UNIVERSITY OF LEEDS

We have

- Reviewed some details of web architecture
- Noted that attacks can target the client, the server or the connection between them
- Discussed the limitations of HTTPS
- Considered how expected page-to-page flow could be disrupted by open redirects or URL jumping
- Explored the dangers of weak authentication and poor session management

Follow-Up / Further Reading

UNIVERSITY OF LEEDS

- Exercise 19 and Exercise 20
- Sullivan & Liu, *A Beginner's Guide to Web Application Security*, McGraw-Hill 2012
- Sennheiser HeadSetup vulnerability
- Troy Hunt blog from 2014 on The Tesco Hack
- Password encryption at PlusNet (2012)
- Jetty session predictability (2006)