ROBERT GORDON UNIVERSITY ABERDEEN

Web Security

Dr Hatem Ahriz
h.ahriz@rgu.ac.uk

ROBERT GORDON UNIVERSITY ABERDEEN

Outline

ROBERT GORDON UNIVERSITY ABERDEEN

Aim

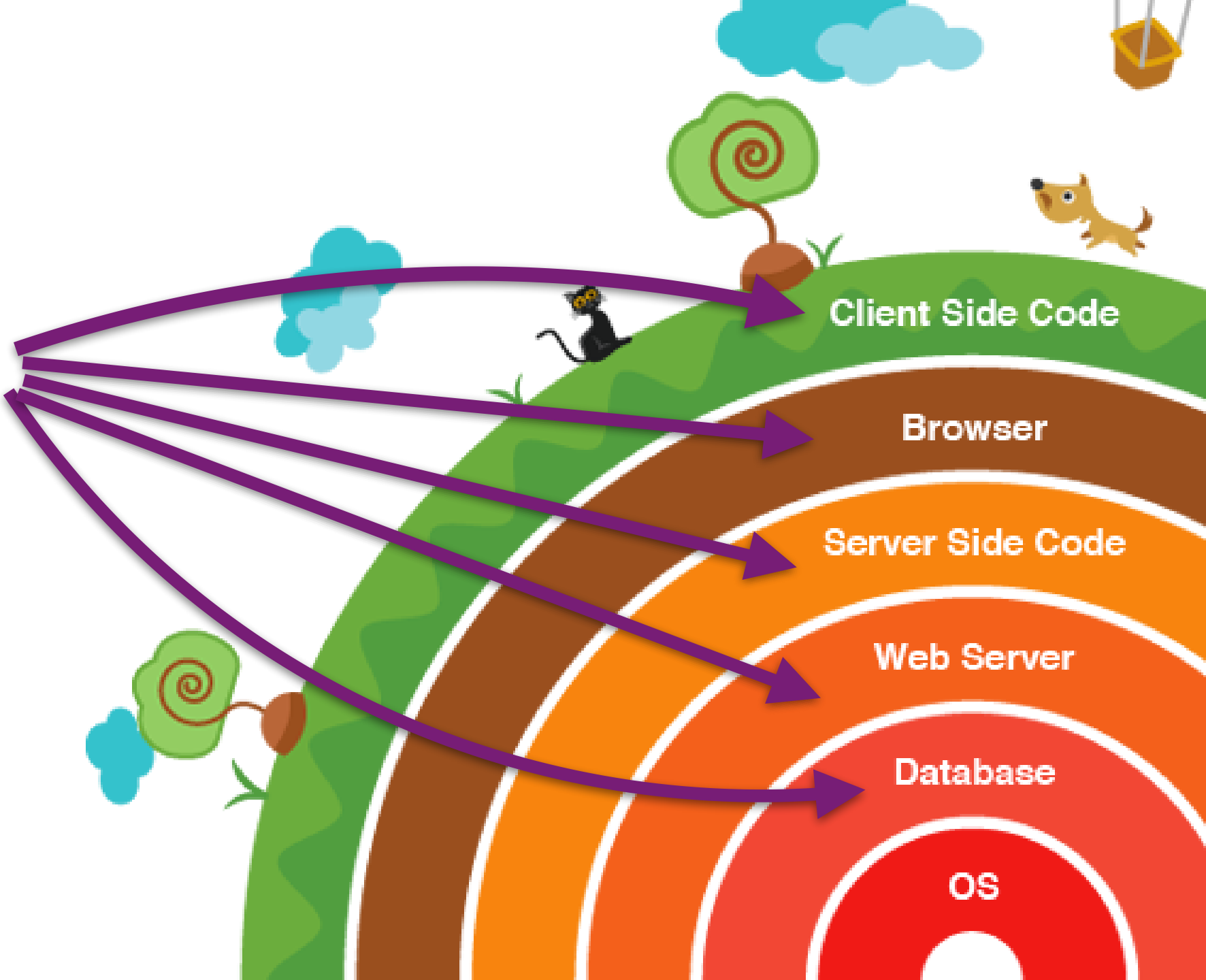
- Understand the main security threats to web apps.
- Develop skills to secure web apps.

- What this topic isn't:
  - It's not a web development topic. You will, however, be expected to get familiar with the syntax of many technologies.
  - It's not a network/systems topic. You will, however, be exposed to networking/OS concepts, commands and tools.

ROBERT GORDON UNIVERSITY ABERDEEN

Aim

Security of these components



ROBERT GORDON UNIVERSITY ABERDEEN

Outline

Day	Lectures/Labs
1	HTTP, Client/Server, Proxy, Web Technologies, Mapping the web app
2	Databases and SQL; Attacking data stores (SQL injection)
3	Attacking Authentication and Session Management; Attacking users: Cross Site Scripting (XSS)
4	Attacking users: Cross Site Request Forgery (CSRF); Attacking Access Control and Backend components
5	API Security; Capture The Flag (CTF)

ROBERT GORDON UNIVERSITY ABERDEEN

Resources

- Web Security Academy - online web security training
- Hacker101 - free web security videos
- TryHackMe - gamified security challenges
- HackTheBox
- Hack.me - sandbox based web application exploit environment
- VulnHub - several virtual machines ready to be downloaded and attacked!
- Database of cheat sheets for all kinds of web exploits

ROBERT GORDON UNIVERSITY ABERDEEN

Warning!

- Do not try any of the techniques learnt here module on any real systems.
- Doing so may get you in trouble with the law!

Certifications

Certified Ethical Hacking

- Package allows you to access:
  - CEH Courseware (for 1 year)
  - CEH Virtual Labs (for 6 months)
  - An exam voucher (valid for 1 year)
  - £450
  - http://www4.rgu.ac.uk/finance/forms/page.cfm?pge=106270

ISC2 Certifications

- Certified in Cyber Security (Free)
- Various online certification training packages.
- ISC2 Online Self-Paced Training | RGU

ROBERT GORDON UNIVERSITY ABERDEEN

Introduction to Web Security

Lecture 1

ROBERT GORDON UNIVERSITY ABERDEEN

Web & Mobile Security: why worry?

- Web & Mobile: the de facto platforms for doing business, sharing content… yet offer an attractive way to breaching into the defence perimeter
- Security: Why are data breaches all over the news? Why is it difficult to get right?
  - Asymmetric race between defenders and attackers

Defenders	Attackers
Need to secure **everything** (every port, service, application, computer… on the organisation's network)	Need to find only **one** vulnerability to breach into the network
Need a vast amount of **resources** to maintain a good security hygiene	May only need a freely available **tool**

- You can not rely on Technology on its own to assure security:
  - Example: Insecure design; Insider threat (malicious or otherwise)

ROBERT GORDON UNIVERSITY ABERDEEN

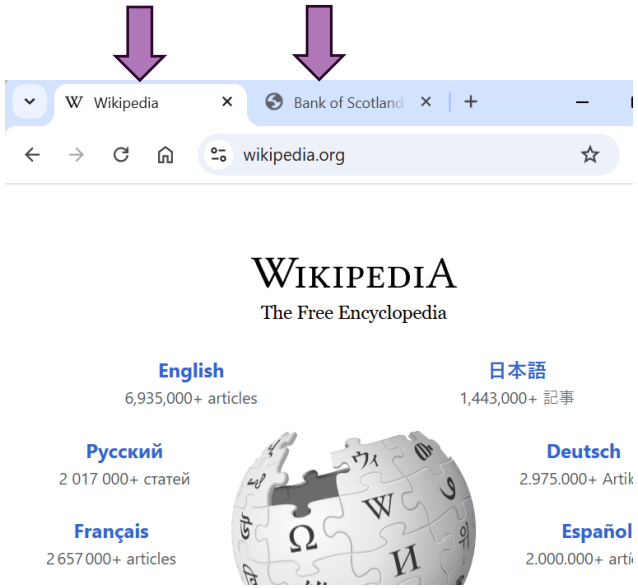
Web & Mobile Security: why worry?

- **What is the core problem?** Users can submit arbitrary input:
  A. Users can interfere with any piece of data transmitted between the client and the server.
  B. Users can interact with the web app in a way that was not expected or desired by the app's developer.
  - Both A and B above can be facilitated by widely available tools that can operate alongside, or independently of, a browser to help attack web apps.
- Factors that exacerbate this problem:
  - **Underdeveloped security awareness**
  - **Rapidly evolving threat profile**
  - **Web development pitfalls**:
    - Custom development
    - Deceptive simplicity of web technologies
    - Resource and time constraints
    - Increasing demand on functionality
    - Backward compatibility requirement.

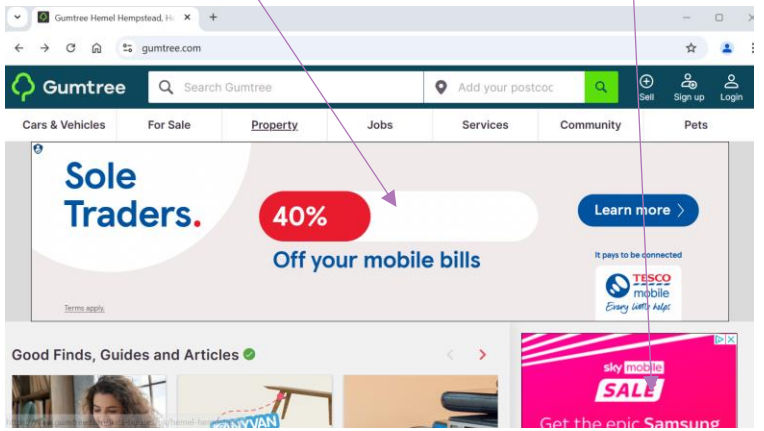
ROBERT GORDON UNIVERSITY ABERDEEN

Security perspectives

- **Browser Security**: Isolate sites from each other, while running in the same browser (Same Origin Policy)
  - Differing views for the web:
    - Simple document viewer, or
    - Powerful app platform?
- **Server Security**: Attackers can send anything to server (and not necessarily through browser)
- **Client Security**: prevent user from being attacked while browsing a site; protect user from trackers, etc.



Different sites interacting i
the same tab (mashups)




```
curl
      -d '{"user":"Alice", "permission":"admin"}'
      -H "Content-Type: application/json"
      -X POST http://example.com/data
```

ROBERT GORDON UNIVERSITY ABERDEEN

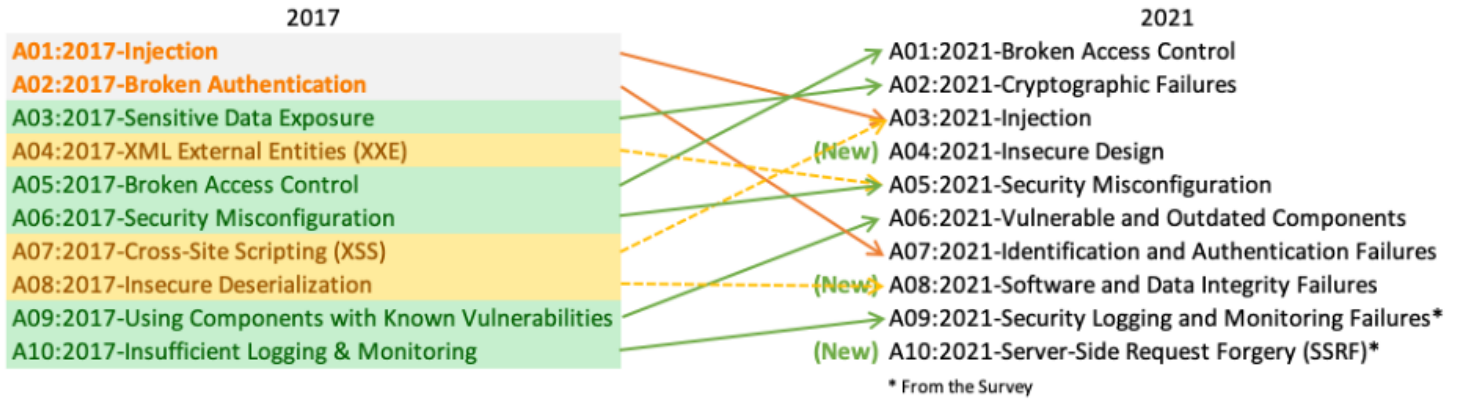
OWASP Top 10 Security Framework

ROBERT GORDON UNIVERSITY ABERDEEN

OWASP (Web) Top 10

https://owasp.org/Top10/

- An open source project that describes the 10 most critical Web app security risks and how to protect against them.



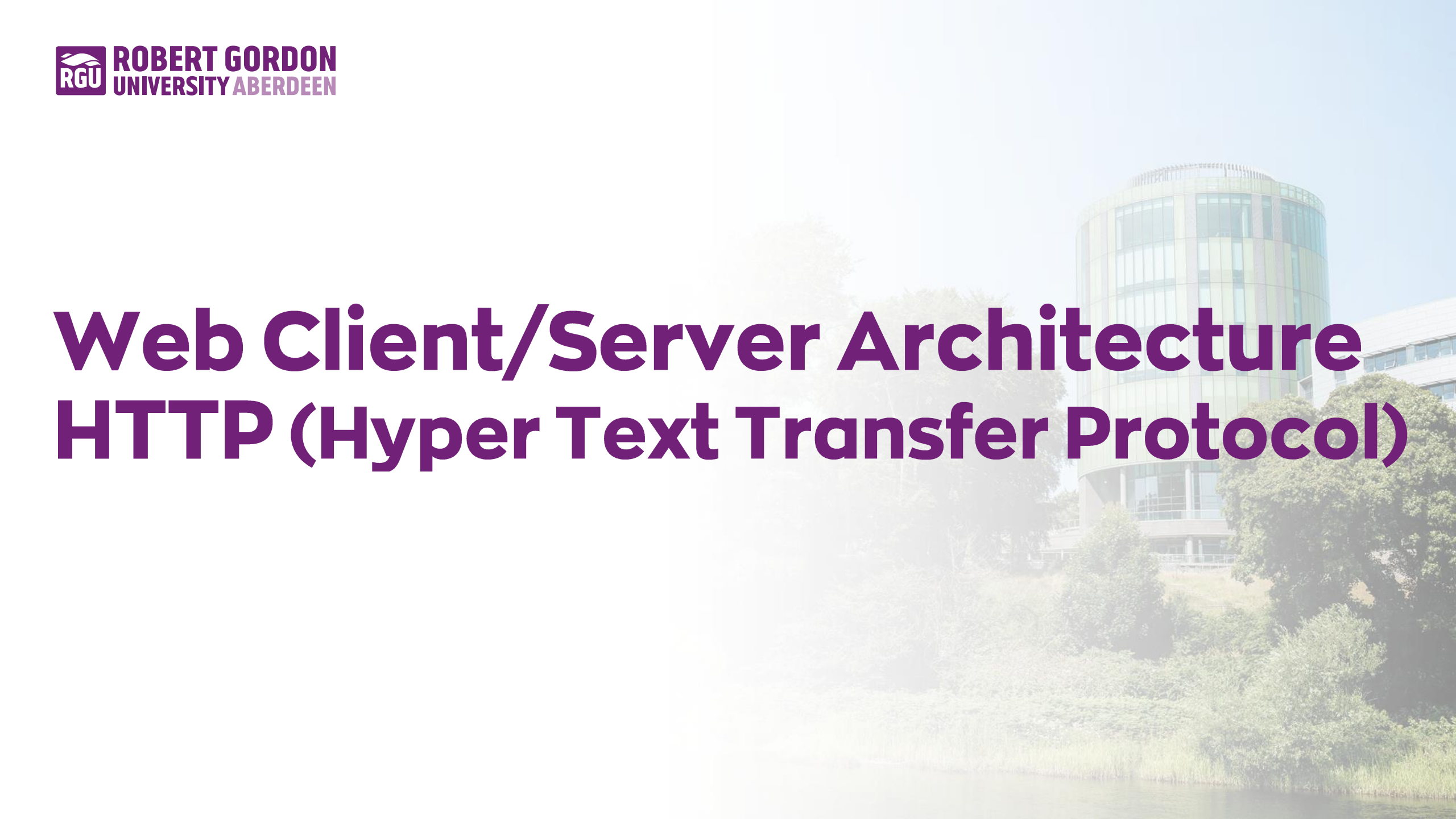
ROBERT GORDON UNIVERSITY ABERDEEN

OWASP Top 10

- A01: Broken Access Control - users acting outside of their intended permissions
- A02: Cryptographic Failures – broken crypto leading to sensitive data exposure
- A03: Injection – user-supplied data not validated and executed as code
- A04: Insecure design – risks related to design and architectural flaws
- A05: Security misconfiguration – improperly configured systems, default settings, etc.
- A06: Vulnerable and outdated components
- A07: Identification and Authentication Failures – flaws in user identification, authentication, and session management
- A08: Software and Data Integrity Failures – relying on plugins, libraries or modules from untrusted sources (e.g., updates downloaded without integrity verification)
- A09: Security logging and monitoring features – failures impacting incident alerting and forensics.
- A10: Server-side Request Forgery (SSRF) – app fetching remote resource without validating user-supplied URL

ROBERT GORDON UNIVERSITY ABERDEEN

Web Client/Server Architecture
HTTP (Hyper Text Transfer Protocol)



ROBERT GORDON UNIVERSITY ABERDEEN

What happens when you browse the Internet?

- You type in a URL in the browser: http://www.example.com/home.html
- Browser (client) resolves the server name of the URL (www.example.com) into an IP address using DNS (Domain Name System): for instance, 216.3.128.12
- Browser requests resource (home.html) from the server by sending HTTP request (via TCP port 80):

```
GET /home.html HTTP/1.1
HOST: www.example.com
...
```

- Server sends HTTP response indicating success (hopefully!) followed by the content of the requested resource:

```
HTTP/1.1 200 OK
Content-Length: 16824
...
<!DOCTYPE html>
<html>
...
</html>
```

- Browser parses the HTML (and may make additional requests for resources required in the home.html file) and renders the page.

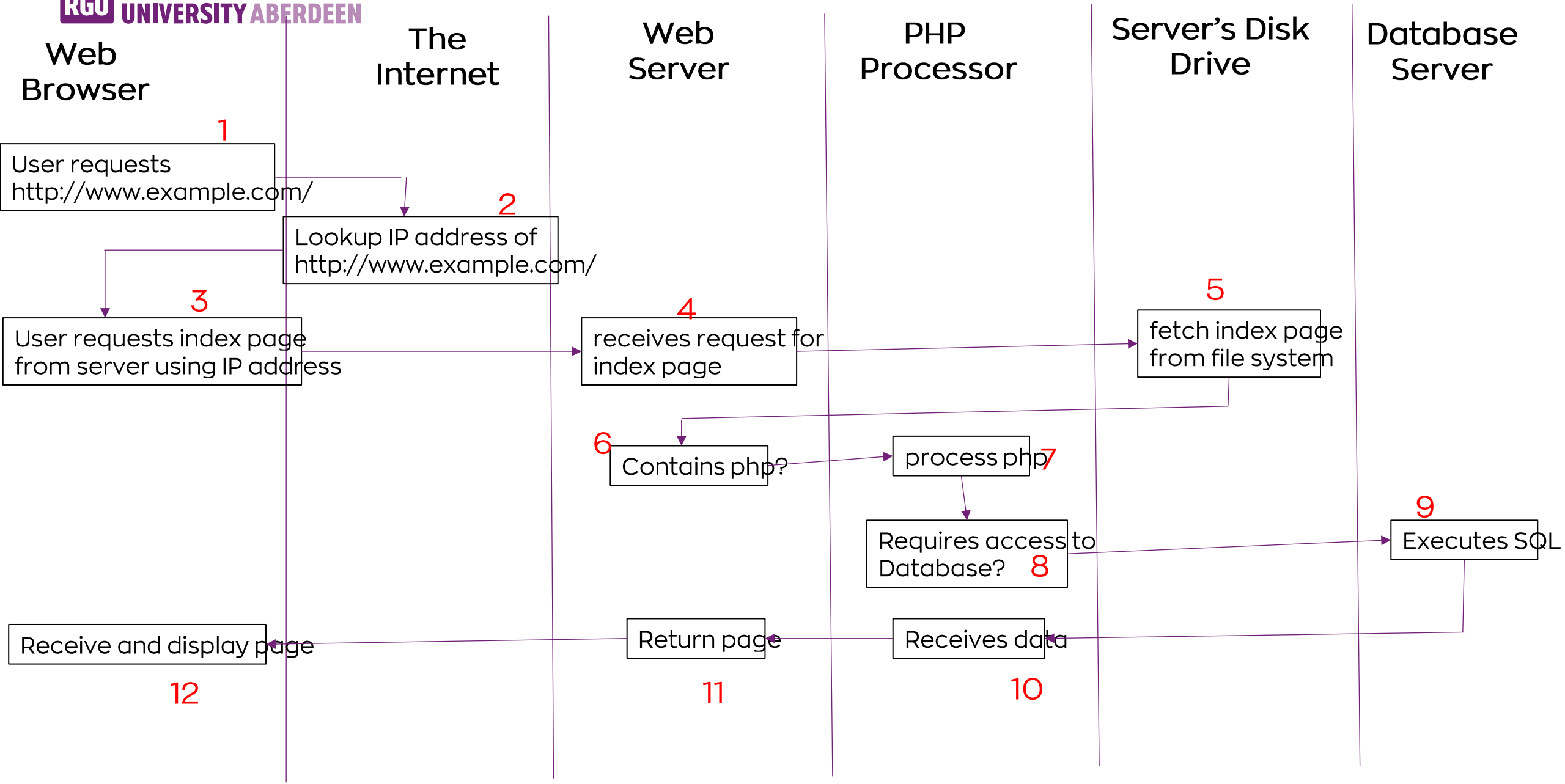
Web Application 3/4 Tier Architecture

ROBERT GORDON UNIVERSITY ABERDEEN

	Client	Web/Application Server	Database Server
Machine Name			
Tier Name	Presentation Tier	Logic Tier/ Data Access Tier	Data Tier
Technology Name	Web browsers: HTML, CSS & JavaScript, JSON, AJAX	Apache/Tomcat, IIS... PHP, JAVA, ASP.NET... / JDBC, SOAP...	MySQL, Oracle...

Client/Server Request/Response

ROBERT GORDON UNIVERSITY ABERDEEN



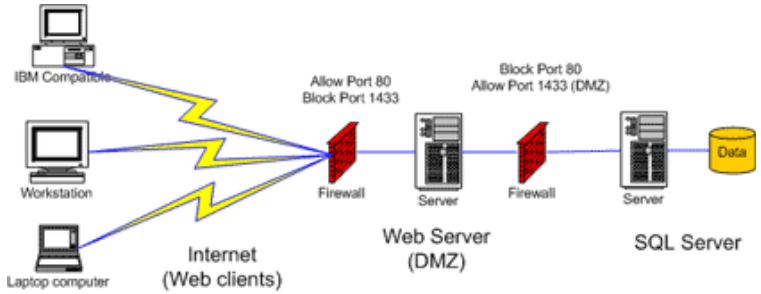
ROBERT GORDON UNIVERSITY ABERDEEN

Some attacks against web servers

- **DoS/DDoS**: Attackers send numerous fake requests to the web server resulting in a crash or becoming unavailable to legitimate users
- **DNS Server Hijacking**: attacker changes DNS setting to redirect traffic to their malicious server
- **Man-in-the-Middle/Sniffing**: Attacker acts as proxy between user and server.
- **Phishing**: tricking user to visit malicious web site
- **Website defacement**: altering a web page (inserting attacker's info or propaganda)
- **Web cache poisoning**: attacker swaps cached content for their own content
- **Password cracking**: attacker targets SMTP server, FTP server…

ROBERT GORDON UNIVERSITY ABERDEEN

Basic Security Measures



- Place web servers and databases in separate security segments on network
- Apply Patches and Updates
- Protocols: block all unnecessary ports and protocols (e.g. NetBIOS, SMB), and ICMP traffic… use secure protocols (instead of FTP, Telnet…)
- Accounts: remove unused modules; disable default user accounts, use strong passwords, run processes using least privileged accounts…
- Files & Directories: disable directory listings; remove non-web files (backup files…); disable serving certain file types; eliminate sensitive config info…
- Detect hacking attempts: monitor logs and use website change detection systems
- TLS Server Certificates: ensure validity…
- Prevent DNS hijacking: use accredited ICANN registrar; protect registrant account info, use DNS monitoring tools…
- Provide physical security to data centre/server room
- Provide training to web developers

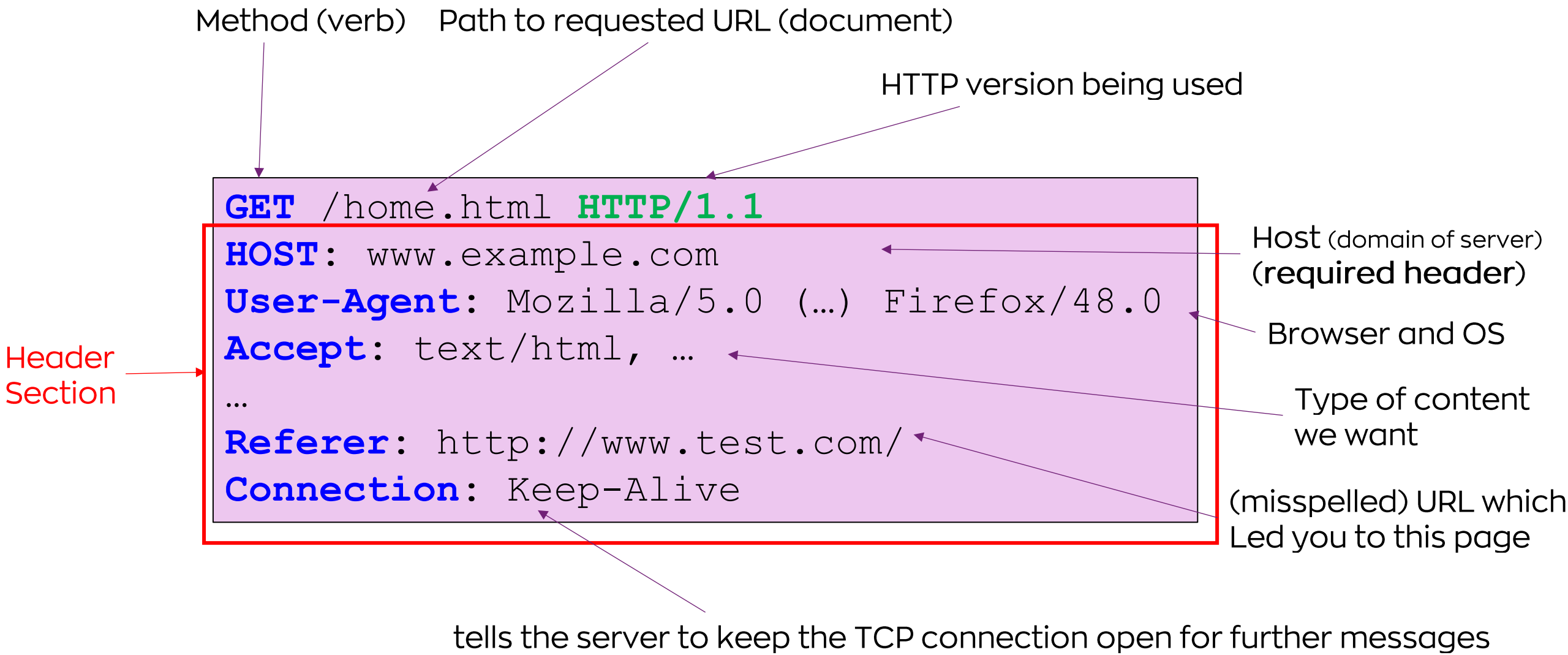
ROBERT GORDON UNIVERSITY ABERDEEN

Hypertext transfer protocol (HTTP)

- Communications protocol used to access the Web.
- Uses a message-based model in which a client sends a request message and the server returns a response message.
- It is connectionless (stateless): although it uses the stateful TCP protocol as its transport mechanism, each exchange of request and response is an autonomous transaction and may use a different TCP connection.
- All HTTP messages (requests and responses) consist of one or more headers, each on a separate line, followed by a mandatory blank line, followed by an optional message body.
- Versions 1.1, and now 2.0 (improves page load speed…)
- Make a HTTP request example: open a command prompt window and type: **curl** https://www.rgu.ac.uk

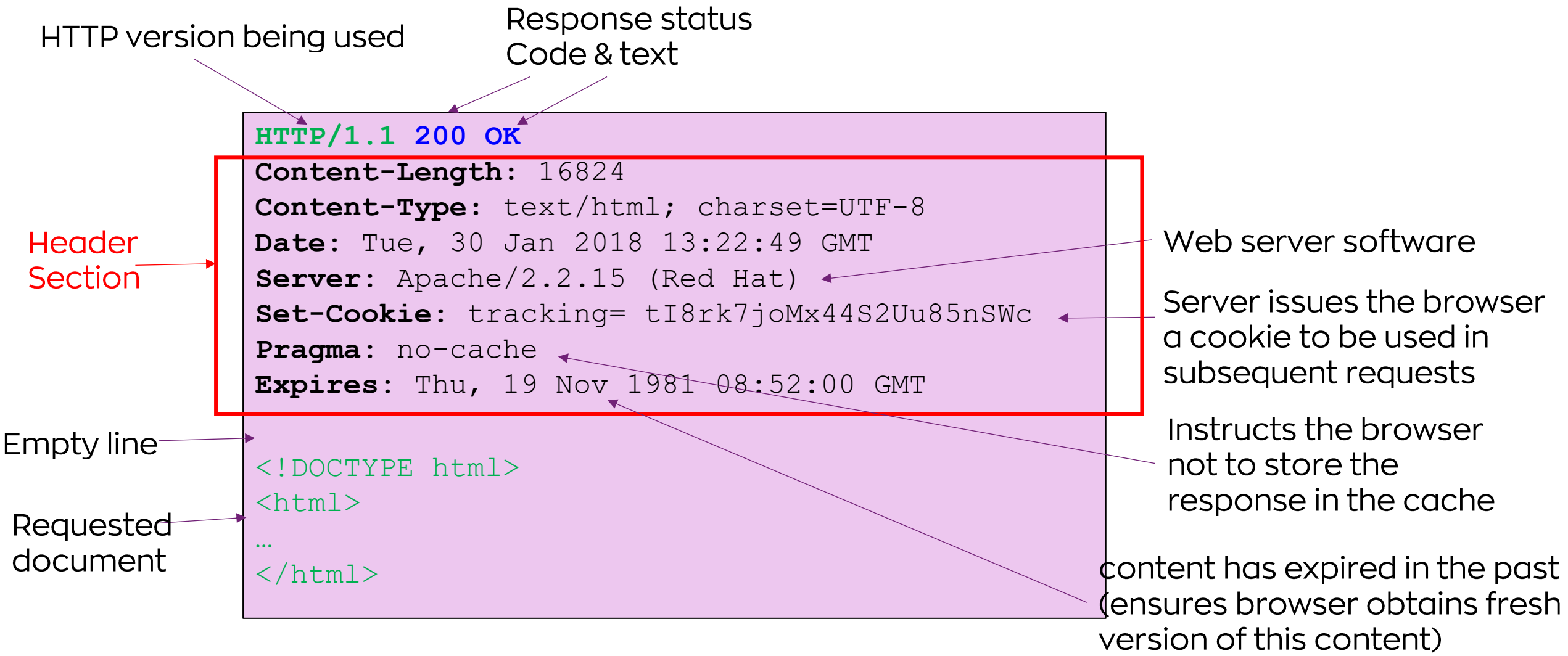
ROBERT GORDON UNIVERSITY ABERDEEN

HTTP Request



ROBERT GORDON UNIVERSITY ABERDEEN

HTTP Response



ROBERT GORDON UNIVERSITY ABERDEEN

HTTP Methods - GET

- When you are attacking web applications, you will be dealing almost exclusively with the most commonly used methods: **GET** and **POST**.
- **GET** retrieves resources from the server (e.g., when clicking a link). It can be used to send parameters to the requested resource in the **URL**:
  - http://www.example.com/page.php?param1=value1&param2=value2
- URLs are displayed on-screen and are logged in various places (e.g., browser history, web server's logs). They are also transmitted in the Referer header to other sites.
  - For these reasons, the query string should not be used to transmit any sensitive data.

ROBERT GORDON UNIVERSITY ABERDEEN

HTTP Methods - POST

- **POST** sends user-generated data to the Web server (e.g., when you fill in a form and click the Submit button). The values are sent in the HTTP **body** and are not visible in the URL.
  - Any data posted will be excluded from the various locations in which logs of URLs are maintained (bookmarks, history, logs, Referer header).
- Is POST, therefore, safer than GET? Yes and No!
  - POST should be preferred to GET when transmitting sensitive data
  - However, user can still tamper with requests sent to server (GET, POST or otherwise) using a proxy (e.g., Burp Suite).

ROBERT GORDON UNIVERSITY ABERDEEN

URLs and REST

- A uniform resource locator (URL) is a unique identifier for a web resource through which that resource can be retrieved. The format of most URLs is as follows:

protocol://hostname:port/path/file?Query#Fragment

https://example.com:4000/a/b.html?user=Alice&year=2025#p2

  - The port is optional, and is only included if it differs from the default used by the relevant protocol (e.g., 80 for http)
  - Query: list of param=value pairs combined with & (and)
  - The protocol (e.g., http) can also be figured by the browser
- REpresentational State Transfer (REST) style URLs will have the parameters of the query string represented as a file path:
  - URL: http://www.example.com/search?make=honda&model=jazz
  - REST-style URL: http://www.example.com/search/honda/jazz

ROBERT GORDON UNIVERSITY ABERDEEN

Ways to specify URLs

- Full URL: 2024 News
- Relative URL: September News
  - Same as http://rgu.ac.uk/news/2024/september
- Absolute URL: Events
  - Same as http://rgu.ac.uk/events
- Fragment URL: Jump to Section 3
  - Scrolls to <a name='section3' /> within page
  - Same as http://rgu.ac.uk/events#section3

ROBERT GORDON UNIVERSITY ABERDEEN

Some HTTP status codes

- Useful to know for security admins when looking at logs, and for intrusion detection systems to recognise possible attacks.
- Also useful when writing scripts/tools to automate interaction with web server (for both legitimate and malicious use).

2xx: Success

- 200 OK – The request was successful

3xx: Redirection – The client is redirected to a different resource

- 301 Moved Permanently – redirects browser to a URL specified in the Location header. Client should use new URL in the future rather than original.
- 302 Found – redirects browser temporarily to a different URL (specified in Location header). Client should revert to original URL in subsequent requests.

4xx: Client Error

- 400 Bad Request – client submitted invalid request.
- 401 Unauthorized – server requires HTTP authentication before request is granted.
- 403 Forbidden – no one is allowed to access the requested resource
- 404 Not Found – requested resource does not exist

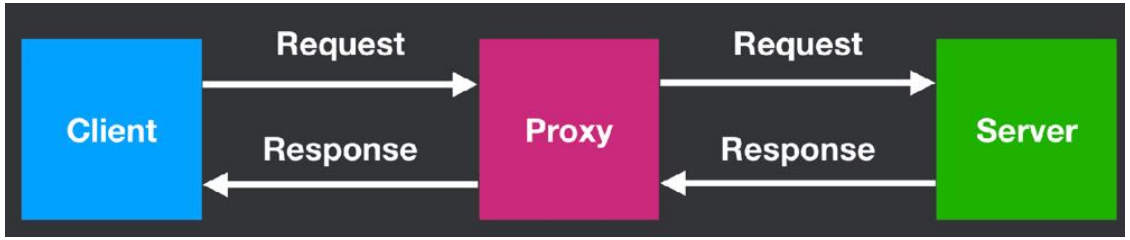
5xx: Server Error

- 500 Internal Server Error – unhandled error within the application
- 503 Service Unavailable –web server is functioning, but app not responding.

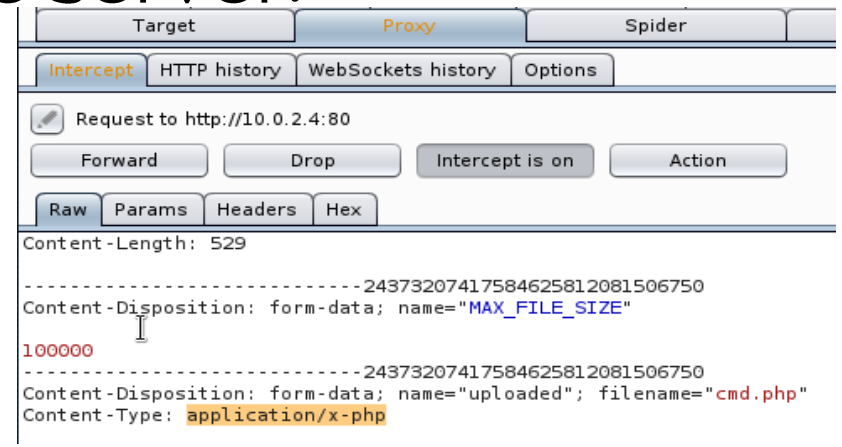
ROBERT GORDON UNIVERSITY ABERDEEN

HTTP Proxies

- An HTTP proxy is a server that mediates access between the client browser and the destination web server.
  - Can cache content
  - Can block content (e.g. malware)
  - Can modify content
  - Can sit in front of many servers ("reverse proxy")



- An HTTP proxy, such as Burp Suite, can be used to bypass client-side controls by intercepting and tampering with requests before they are sent to the server:



ROBERT GORDON UNIVERSITY ABERDEEN

Web Hacker's Methodology & Toolkit

ROBERT GORDON UNIVERSITY ABERDEEN

Methodology:

- Map the Application's Content (visible/hidden content, public resources…)
- Analyse the Application (functionality, technologies, data entry points…)
- Test Client-Side Controls
- Test Authentication Mechanism (unsafe transmission of credentials, password recovery mechanism…)
- Test the Session Management Mechanism (cookies, tokens…)
- Test Access Controls (test with multiple accounts…)
- Test for Input-Based Vulnerabilities (SQL injection, XSS, file inclusion…)
- Test for Logic Flaws (test multistage processes…)
- Test for Shared Hosting Vulnerabilities
- Test for Application Server Vulnerabilities

Toolkit:

- Must have tools:
  - Web browsers
  - Integrated Testing Suites (intercepting proxies, spiders, scanners…): e.g., Burp Suite
- Nice to have tools:
  - Kali Linux: to exploit vulnerabilities using pre-built frameworks and tools

ROBERT GORDON UNIVERSITY ABERDEEN

Lecture 2

Brief Overview of Web Technologies & Gathering Info about the target app

ROBERT GORDON UNIVERSITY ABERDEEN

Client-Side Web Technologies

HTML, CSS, JavaScript, JSON, XML

HTML Forms: capture user input

- Input elements: <input type=… name=…>
  - **Type** can be: Text/password/date/email/url/file/hidden/Submit/radio/checkbox…
- Input attributes:
  - Value/Placeholder/Size/Maxlength/Min and Max/Required/Disabled…
- Form Validation: Can be done in HTML or Java Script

HTML 5: more features and, potentially, larger threat surface

- Support for multimedia: <video>…
- Web Storage: key-value pair storage within the browser (a few MBs). Unlike cookies, they are not sent to server. Data can be Stored with no expiration date or for one session only.
  - User privacy issues, client-side SQL injection…
- Geolocation: provides a method that returns the user's position (User privacy issues)
- Lots of tags: <img>, <canvas>, <link>, <style> , <script>…

Client-side validations can be bypassed and so should be used to enhance usability, not security. Always use validation both client-side and server-side.

ROBERT GORDON UNIVERSITY ABERDEEN

CSS and JavaScript

```
<!-- External CSS file -->
<link rel='stylesheet' href='/path/to/styles.css' />

<!-- Inline CSS -->
<style> body { color: aqua; } </style>


<!-- External JS file -->
<script src='/path/to/script.js'></script>

<!-- Inline JS -->
<script> window.alert('hello')</script>
```

ROBERT GORDON UNIVERSITY ABERDEEN

Ajax & JSON

- Ajax: technique that allows developers to read/update data from/to the server without reloading the page.
  - For example, when clicking Add item to a shopping cart.
- Implementation through:
  - XMLHttpRequest (XHR) objects and requests; or Fetch requests
    - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_xmlhttp
  - XML or JSON format for data
    - Example of JSON: {“name”: “John Smith”, “email”: j.smith@rgu.ac.uk}

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy (SOP)

- Same-Origin Policy (SOP) is a security mechanism implemented in browsers to prevent a website (attacker.com) from accessing the content (e.g., response body) of requests to a different origin (victim.com) unless explicitly allowed by the server at victim.com
- Therefore, even if JavaScript running on attacker.com uses fetch() to send a GET request to victim.com, the browser will block access to the response body unless victim.com sends the appropriate **CORS** headers.
- **Origin** is defined by:
  - Scheme (e.g., HTTPS) + hostname + port
  - protocol://hostname:port for example, https://example.com:4000

  - Therefore, attacker.com is considered a different origin from victim.com.

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy

- This is the fundamental security model of the web:
  - **Two pages from different origins should not be allowed to interfere with each other**
- Specifically, given two separate **JavaScript execution contexts**, one should be able to access the other only if shares the same origin
- An origin is analogous to an OS process
- The web browser itself is analogous to an OS kernel
- Sites rely on the browser to enforce all the system's security rules
- If there's a bug in the browser itself then all these rules go out the window (just like in an OS)

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy or not?

https://example.com/a/
https://example.com/b/

- Yes!

https://example.com/a/
https://www.example.com/b/

- No! Hostname mismatch!

https://example.com/
http://example.com/

- No! Protocol mismatch!

http://example.com/
http://example.com:81/

- No! Port mismatch!

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy (SOP)

- SOP focuses on restricting cross-origin access to sensitive data (like cookies, localStorage, or DOM content). It doesn't enforce restrictions on how a website can be displayed (e.g., framing).
- Open your browser and go to https://www.wikipedia.org then open the browse's console and try:

```
const res = await fetch('https://example.com') // Not allowed!


const iframe = document.createElement('iframe')
iframe.src = 'https://example.com'
document.body.append(iframe) // Allowed!


iframe.contentDocument.body.style.backgroundColor = 'red' // Not allowed!
```

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy – exceptions

Which of these requests from https://example.com are allowed?

```
<!doctype html>
<html lang='en'>
<head>
<meta charset='utf-8' />
<link rel='stylesheet' href='https://other1.com/style.css' />
</head>
<body>
<img src='https://other2.com/image.png' />
<script src='https://other3.com/script.js'></script>
</body>
</html>
```

ROBERT GORDON UNIVERSITY ABERDEEN

Same Origin Policy – exceptions

Answer: All of them!

- Embedded static resources can come from another origin
  - Images (e.g. hotlinking to memes)
  - Scripts (e.g. Facebook like button, ads, tracking scripts)
  - Styles (e.g. Google Fonts)
- Why was it designed this way?

ROBERT GORDON UNIVERSITY ABERDEEN

Cross Origin Resource Sharing (CORS)

- CORS: provides a controlled relaxation of SOP to allow cross-origin requests. The server must explicitly allow it by setting appropriate CORS headers.
- See: https://portswigger.net/web-security/cors

Client makes *preflight* request
To determine if server is willing to share a resource

```
OPTIONS / HTTP/1.1
HOST: friendly.com
Origin: http://mywebsite.com
Access-control-Request-Method: PUT
```

Server indicates whether the resource can be retrieved via cross-domain request

```
HTTP/1.1 200 OK
…
Access-Control-Allow-Origin: http://mywebsite.com
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 10
```

ROBERT GORDON UNIVERSITY ABERDEEN

Content Security Policy (CSP)

- Websites can restrict which external domains are allowed to provide resources (e.g., scripts and images) using a CSP.
- To enable CSP, HTTP response must contain a header called Content-Security-Policy with a value containing the policy. The policy itself consists of one or more directives, separated by semicolons.

```
Content-Security-Policy: default-src 'self';
                         script-src 'self' https://trusted-scripts.com 'sha256-abc123...' 'strict-dynamic';
                         style-src 'self' https://trusted-styles.com 'unsafe-inline';
                         img-src 'self' https://images.com data:;
                         font-src 'self' https://fonts.com;
                                  …
                         report-uri https://csp-report.example.com/report;
```

In the example above, `script-src` will only allow scripts to be loaded from the same origin as the page itself as well as from `https://trusted-scripts.com` and a hash of the trusted script is included

…

https://portswigger.net/web-security/cross-site-scripting/content-security-policy

ROBERT GORDON UNIVERSITY ABERDEEN

Bypassing Client-Side Controls

Bypassing HTML Form Restrictions

- Restrictions such as maxlength, required, hidden… can be bypassed
  - Using your browser's Web Dev tools
    - Example: here I changed the input type for CampusMoodle password field from password to text:
  - Using an HTTP proxy (e.g., Burp)
    - Example: looking at the HTML source for this form we see that there is a hidden input:

Product: iPhone X
Price: 999
Quantity:
Buy

```
<form action="shop.php?prod=1" method="post">
Product: iPhone X <br>
Price: 999 <br>
Quantity: <input type="text" name="quantity"><br>
<input type="hidden" name="price" value="999"><br>
<input type="submit" value="Buy">
</form>
```

```
POST /shop.php?prod=1 HTTP/1.1
…
quantity=1&price=999
```

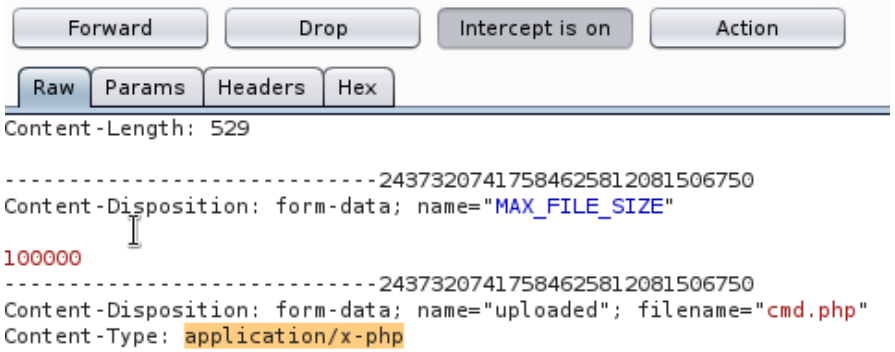
Price can be intercepted and changed!

Obviously, in the case where data is transmitted over the URL, you don't need any tools to change those!

ROBERT GORDON UNIVERSITY ABERDEEN

Changing other parts of the HTTP request

- For example, if there is a restriction on the type of file we can upload to web site, we can change the value of the `Content-Type` header (e.g., from php (`application/x-php`) to jpeg format `image/jpeg`)



- `Referer` header: you can imagine a scenario where a server checks that you are coming from a certain page before allowing you to reset your password. You can change the `Referer` to the value required.
- `Cookie` header: after a customer has logged on, server responds with a header `Set-Cookie: DiscountAgreed=25`. If the server trusts the value posted back by the client you can send: `Cookie: DiscountAgreed=100`

ROBERT GORDON UNIVERSITY ABERDEEN

Server-Side Technologies

- Dynamic content is generated by code executing on the server.
- When browser requests a dynamic resource, normally it does not simply ask for a copy of that resource. It also submits various parameters that enable the web application to generate content that is tailored to the individual user.
- HTTP requests can be used to send parameters to the application in the:
  - URL query string (Get method) or file path of REST-style URLs
  - HTTP cookies
  - Body of requests using the POST method
- In addition, the server-side application may in principle use any part of the HTTP request as an input to its processing. For example, an application may process the `User-Agent` header to generate content optimised for the type of browser used.

ROBERT GORDON UNIVERSITY ABERDEEN

Server-Side Technologies

- Web applications employ a wide range of technologies on the server side to deliver their functionality:
  - Scripting languages such as PHP, Perl, Java, C#, JavaScript…
  - Web servers such as Apache and IIS
  - Databases such as MS-SQL, Oracle, and MySQL
  - Other back-end components: filesystems, SOAP services, and directory services
- Each of these technologies can come bundled in a web stack, such as:
  - LAMP (Linux, Apache, MySQL, PHP). XAMPP is a variation of LAMP.
  - WISA (Windows, IIS, SQL Server, ASP)
  - MEAN (Mongo DB, Express, Angular.js, Node.js)
- Web frameworks for MVC development, such as:
  - Laravel for PHP, Strust or Spring for Java…

ROBERT GORDON UNIVERSITY ABERDEEN

PHP

- Basic syntax:

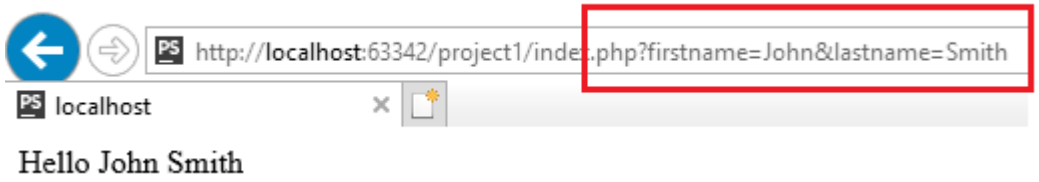
```
<?php
// some code
?>
```

- Php files can contain html, css, javascript and php code.
- Variables start with $, are case sensitive and do not need to be declared
- Example: https://www.w3schools.com/php/showphp.asp?filename=demo_intro

PHP handling of user input

- PHP can receive input from a number of sources:
  - HTTP Get Requests: use &_GET

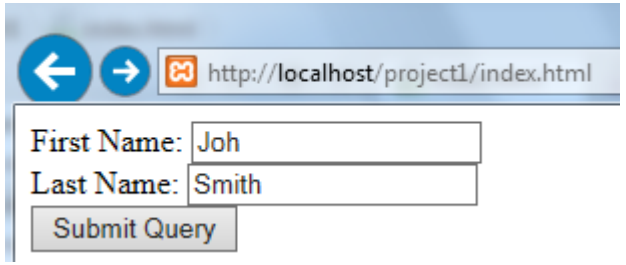
```php
<?php
$f_name=$_GET["firstname"];
$l_name=$_GET["lastname"];
echo"Hello ".$f_name. " ".$l_name;
?>
```



  - HTML Form: using &_POST or &_GET

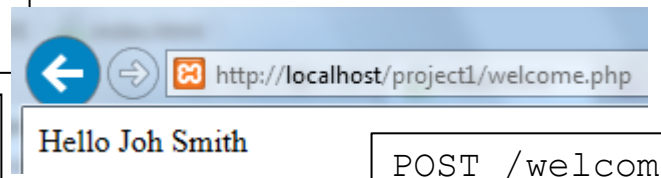
index.html

```html
<!DOCTYPE html>
<html>
<body>
<form action="welcome.php" method="post">
    First Name: <input type="text" name="firstname"><br>
    Last Name: <input type="text" name="lastname"><br>
    <input type="submit">
</form>
</body>
</html>
```



Welcome.php

```php
<html>
<body>
<?php echo"Hello ".$_POST["firstname"]. " ".$_POST["lastname"];?>
</body>
</html>
```




```
POST /welcome.php HTTP/1.1
…
firstname=John&lastname=Smith
```

ROBERT GORDON UNIVERSITY ABERDEEN

PHP Form Validation

- htmlspecialchars() function converts special characters to HTML entities:
  - replace HTML characters like < and > with < and >.
  - This prevents attackers from exploiting the code by injecting HTML or Javascript code.
- stripslashes() removes backslashes (\) from the user input data
- trim() removes extra space, tab, newline characters from the user input data
- preg_match() searches a string for pattern, returning true if the pattern exists, and false otherwise. Example:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space
allowed"; }
```

- filter_var() function can be used with flags to validate common data types and data input, such as: FILTER_VALIDATE_EMAIL, FILTER_SANITIZE_STRING...

ROBERT GORDON UNIVERSITY ABERDEEN

Encoding Schemes

Web apps employ several encoding schemes for their data. When you are attacking a web app, you will frequently need to encode data using a relevant scheme to ensure that it is handled in the way you intend (e.g., to **defeat input validation**)

ROBERT GORDON UNIVERSITY ABERDEEN

URL Encoding

- URLs are permitted to contain only the printable characters in the US-ASCII character set (range 0x20 to 0x7e, inclusive). several characters within this range are restricted because they have special meaning within the URL scheme itself or within the HTTP protocol.
- The URL-encoding scheme is used to encode any problematic characters within the extended ASCII character set so that they can be safely transported over HTTP.
- The URL-encoded form of any character is the % prefix followed by the character's two-digit ASCII code expressed in hexadecimal.
- Examples:

Character	URL Encoding
=	%3d
%	%25
Space	%20

ROBERT GORDON UNIVERSITY ABERDEEN

Unicode Encoding

- Unicode is a character encoding standard that is designed to support all of the world's writing systems. The 16-bit Unicode-encoded form of a character is the % prefix followed by the character's Unicode code point expressed in hexadecimal:
  - For example: é is encoded as %u00e9
- **UTF-8** is a variable-length encoding standard that employs one or more bytes to express each character expressed in hexadecimal and preceded by the % prefix:
  - For example: © is encoded as %c2%a9

ROBERT GORDON UNIVERSITY ABERDEEN

HTML Encoding

- Used to represent problematic characters so that they can be safely incorporated into an HTML document. For example:

Character	HTML encoding	HTML encoding using ASCII code	HTML encoding using Hex form
“	&quot;	&#34;	&#x22;
‘	&apos;	&39;	&#27;
<	&alt;		
>	&gt;		
&	&amp;		

ROBERT GORDON UNIVERSITY ABERDEEN

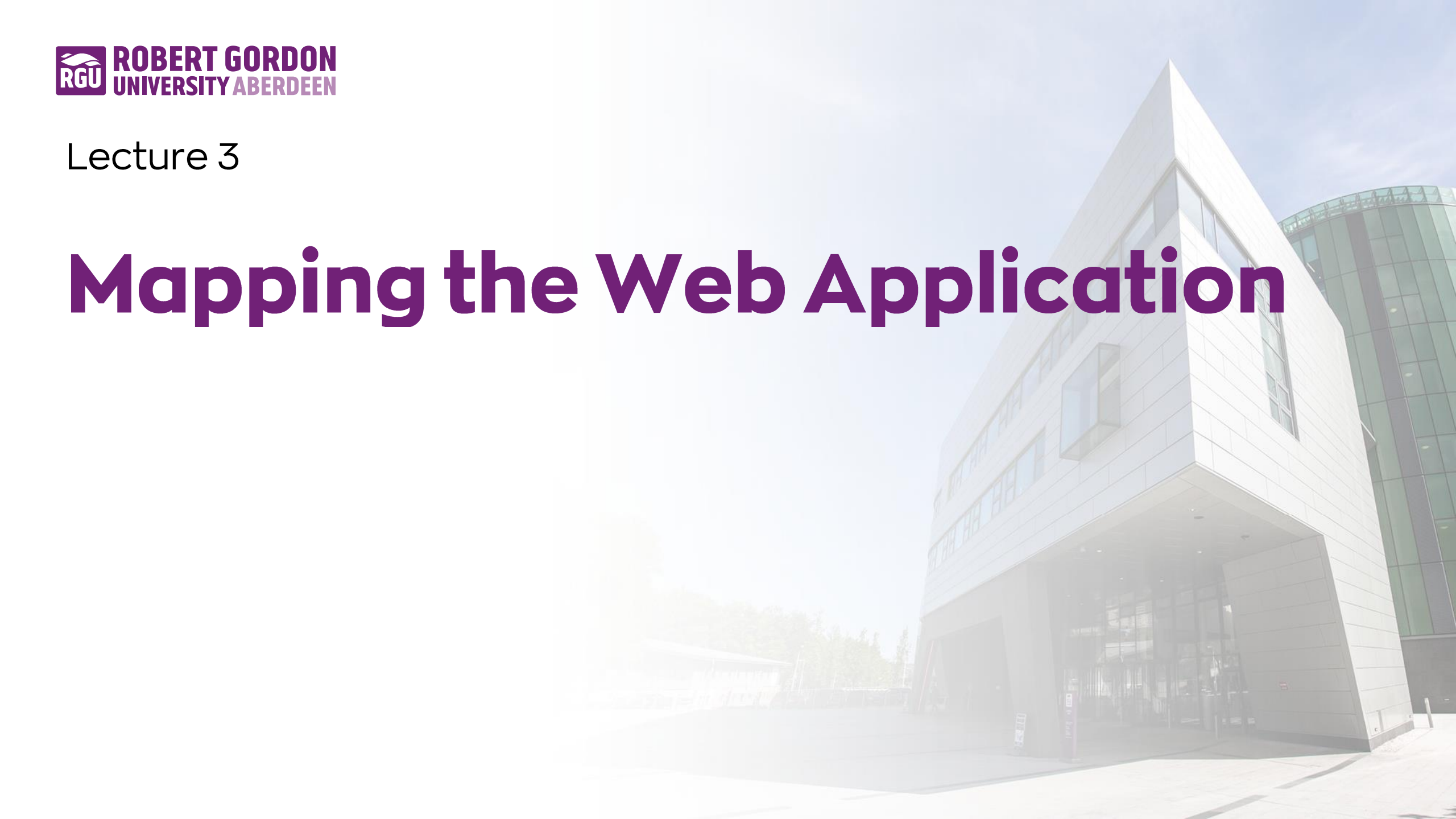
Base64 Encoding

- Allows any binary data to be safely represented using only printable ASCII characters (all letters of alphabet in capital and lower case, plus all digits, plus + and /).
- It is commonly used to encode e-mail attachments for safe transmission over SMTP. It is also used to encode **user credentials** in basic HTTP authentication (e.g., within cookies) or to **obfuscate** sensitive data!
- Data is processed in blocks of three bytes. Each block is divided into four chunks of six bits. If the final block results in fewer than three chunks of output data, the output is padded with one or two = characters.
- Example: Database and Web Security is encoded as RGF0YWJhc2UgYW5kIFdlYiBTZWN1cml0eQ==

ROBERT GORDON UNIVERSITY ABERDEEN

Lecture 3

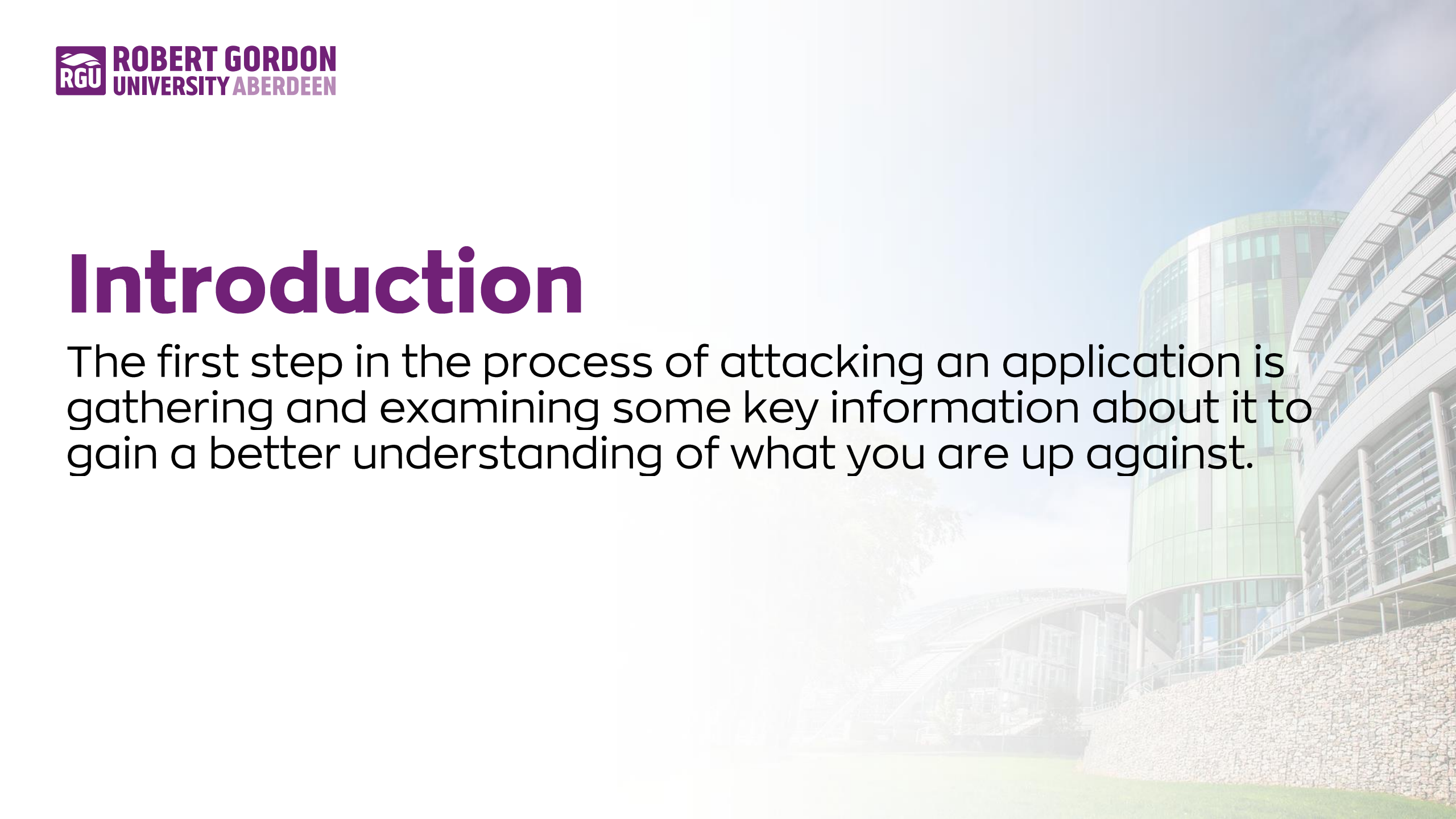
Mapping the Web Application



ROBERT GORDON UNIVERSITY ABERDEEN

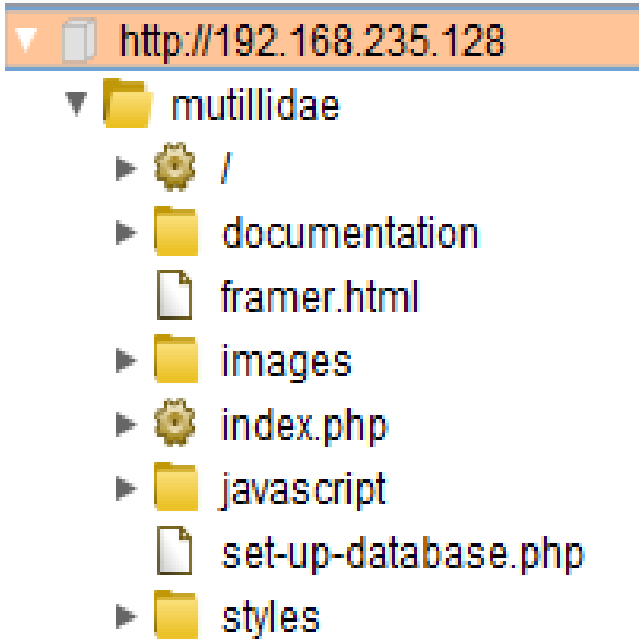
Introduction

The first step in the process of attacking an application is gathering and examining some key information about it to gain a better understanding of what you are up against.



ROBERT GORDON UNIVERSITY ABERDEEN

Enumerating Content and Functionality – Web Spidering



- The majority of the content and functionality can be identified via **manual** browsing. Also consult «site map» if it exists.
- However, to perform a rigorous inspection, you must employ more advanced techniques such as **web spidering**.
- Web spiders work by requesting a web page, parsing it for links to other content, requesting these links recursively until no new content is discovered. Some tools also parse client-side JavaScript to extract URLs pointing to further content.
- Example: Burp Suite and OWASP ZAP.

ROBERT GORDON UNIVERSITY ABERDEEN

Some Limitations of Web Spiders

- Menus dynamically created using JavaScript code.
- Input validation checks. The spider may not be intelligent enough to understand error returned and get passed the validation.
- Authentication: some spiders can be configured with a token for an authenticated session or credentials to submit to the login page. However, sessions could be terminated for various reasons, example:
  - By following all URLs, the spider will request the logout page, causing the session to terminate
  - After multiple invalid input
- To counteract these limitations: user-directed spidering

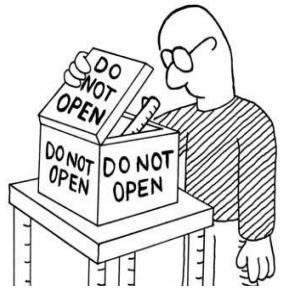
ROBERT GORDON UNIVERSITY ABERDEEN

Discovering Hidden Content

- It is common for applications to contain content and functionality that is not directly linked to the main visible content.
- Many web servers put a file **`robots.txt`** in the web root to list URLs that the site does not want search engines to index. Can be an interesting place to look.
- Backup archives and old version files
- Configuration files containing sensitive data such as database credentials.
- Log files containing sensitive information such as valid usernames, session tokens, and URLs visited.
- Source files from which the live application has been compiled.
- Comments in source code: may be left behind by developers
  - HTML: `<!-- TO DO -->`
  - JavaScript: /* Test this function */
- Tools using word lists of commonly used file and directory names could be used to discover some of the hidden content.

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /~joe/
```

Fun Robots.txt
https://searchengineland.com/fun-robots-txt-263796



ROBERT GORDON UNIVERSITY ABERDEEN

Using Inference to discover hidden content

- Many applications employ some kind of naming convention.
- For example, if there is a http://example.com/auth/**Add**User page, you may want to try:
  - http://example.com/auth/**Update**User
  - http://example.com/auth/**Delete**User
  - …
- Another example: if a web app links to **AnnualReport2024.pdf** then you can try to access **AnnualReport2025.pdf**
- You can use Burp (Intruder) to brute-force attack the part in the file name that is likely to change and let it use a word list of your choice.
  - Burp (Pro edition) has an easier feature for content discovery

ROBERT GORDON UNIVERSITY ABERDEEN

Use of Public Information (Passive Reconnaissance)

The type of information we can gather about a target using public information include:

- IP address and Domain name info
- Technologies used
- Other websites on the same server
- DNS records
- Sub-domains

Available resources to use include:

- Search engines: tend to have powerful spiders and can keep cached copies
- Web archives (e.g., WayBack Machine www.archive.org)
- Whois lookup: info about the owner of the target site
- Netcraft Site report: technologies used on the target

ROBERT GORDON UNIVERSITY ABERDEEN

Analysing the Application - Key areas to investigate

- The application's core functionality — the behaviour of expected actions
- Other application behaviour, including off-site links, error messages, administrative and logging functions, and the use of redirects
- The core security mechanisms: management of session state, access controls, and authentication mechanisms (user registration, password change, and account recovery)
- All the locations where the application processes user-supplied input: every URL, query string parameter, item of POST data, cookie, HTTP header that the application might process (`User-Agent`, `Referer`, `Accept`...).
  - Example of entry points:
    - URL: http://www.example.com/search?make=honda&model=jazz
    - REST-style URL: http://www.example.com/search/honda/jazz

ROBERT GORDON UNIVERSITY ABERDEEN

Vulnerability Scanners

ROBERT GORDON UNIVERSITY ABERDEEN

Vulnerability Scanners

Automate the spidering, content discovery and probing for common vulnerabilities including:

- Cross-site scripting, SQL injection, Path traversal, Command injection...

Examples of vulnerabilities that scanners cannot reliably detect:

- **Broken access controls**, which enable a user to access other users' data, or a low-privileged user to access administrative functionality
- Attacks that involve modifying a parameter's value in a way that has **meaning** within the application — for example, a hidden field representing the price of a purchased item or the status of an order.
- **logic flaws**, such as beating a transaction limit using a negative value
- **Session hijacking attacks** in which a sequence can be detected in the application's session tokens, enabling an attacker to masquerade as other users.
- **Leakage of sensitive information** such as listings of usernames and logs containing session tokens.

ROBERT GORDON UNIVERSITY ABERDEEN

The Ethics of Vulnerability Scanning

Running an unrestricted automated scan without any user guidance may be quite dangerous to the application and the data it contains.

The scan may discover an administration page that contains functions to reset user passwords, delete accounts... or can reveal sensitive data

If the scanner blindly requests every function, this may result in access being denied to all users of the application.

Never run a vulnerability scan without the explicit consent of the application/site owner. Make sure you establish ground rules for the scan, and stay within the confines of your scope.