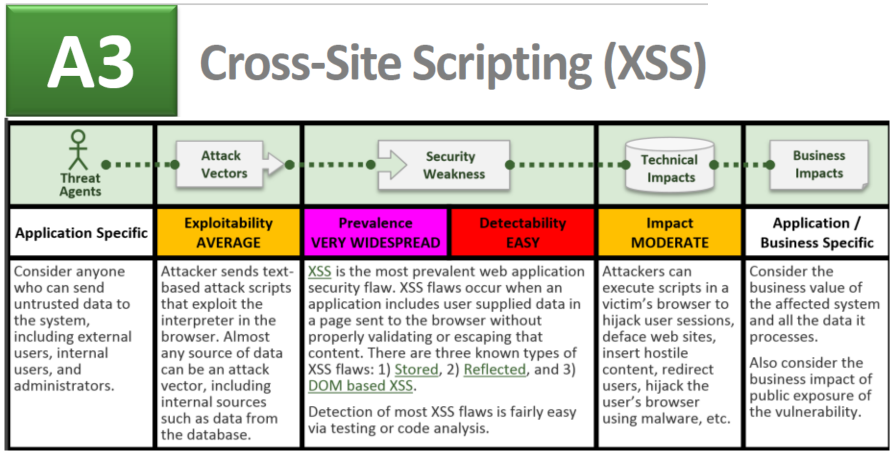
COM1008: Web and Internet Technology

Lecture 18. Information Security Part 2



A3 Cross-Site Scripting (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS. Detection of most XSS flaws is fairly easy via testing or code analysis.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

https://www.owasp.org/index.php/Category:
OWASP_Top_Ten_Project

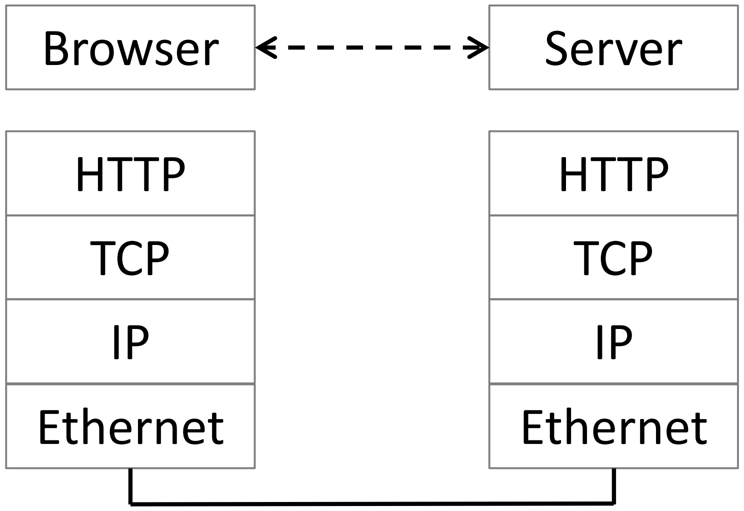
Dr. Steve Maddock
s.maddock@sheffield.ac.uk

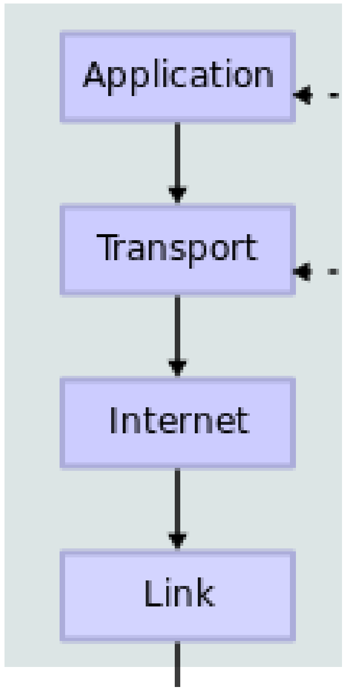
1. Introduction

- *Last week*: Information; security; risk – vulnerabilities, threats, attacks; Three classical goals of information security: Confidentiality, Integrity, Availability; legal frameworks
- *Today*: computer and network security: some practicalities
  - HTTPS
  - Cookies
  - Top 10 web application security flaws
  - Cross-Site Scripting (XSS)
  - FormMail.pl
  - Denial of Service
- *Next lecture*: cryptography

2. Hypertext Transfer Protocol (HTTP)

- From earlier lecture:



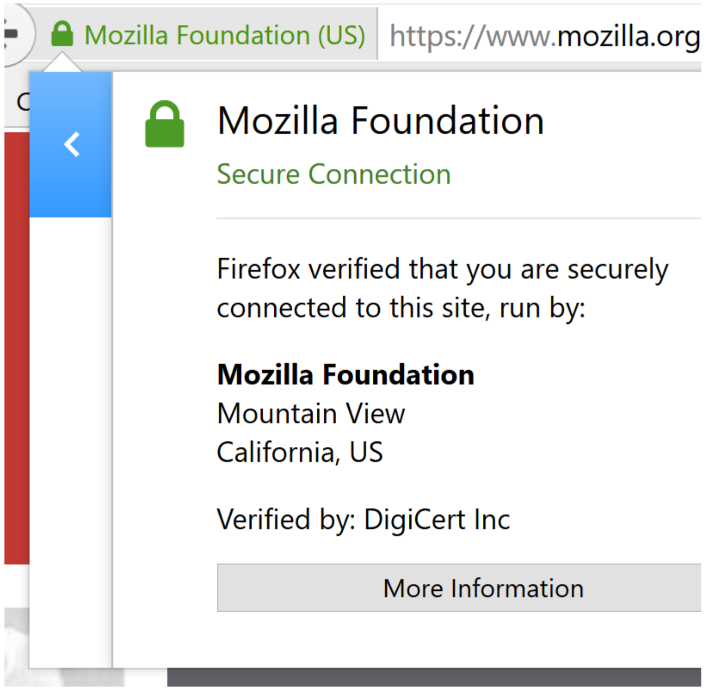


TCP/IP 4-layer model

2.1 HTTPS - Protocol for secure transmission

- HTTP + connection encrypted by Transport Layer Security (or the older Secure Sockets Layer)
  - Creates a secure channel over an insecure network
- Makes use of certificate authorities
  - public-private keys (see next lecture)
- Authentication of server and website
  - Protects against man-in-the-middle attacks
- Bidirectional encryption of communications
  - Protects against eavesdropping

Green for extended validation certificate


Mozilla Foundation (US) https://www.mozilla.org

Mozilla Foundation
Secure Connection

Firefox verified that you are securely connected to this site, run by:

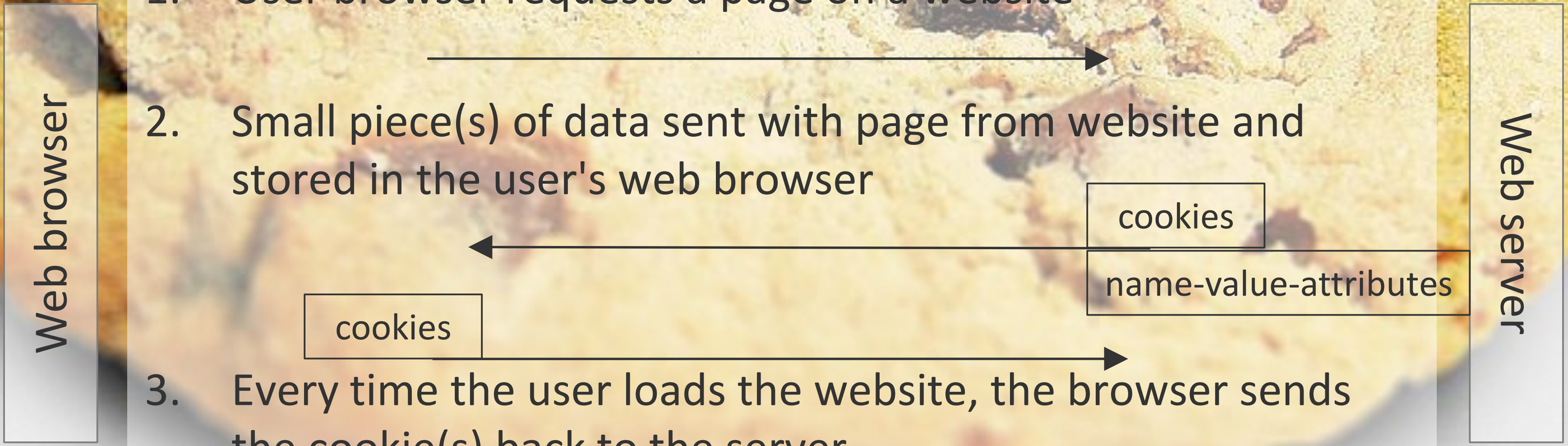
Mozilla Foundation
Mountain View
California, US

Verified by: DigiCert Inc

https://support.mozilla.org/en-US/kb/page-info-window-view-technical-details-about-page

3. HTTP cookies

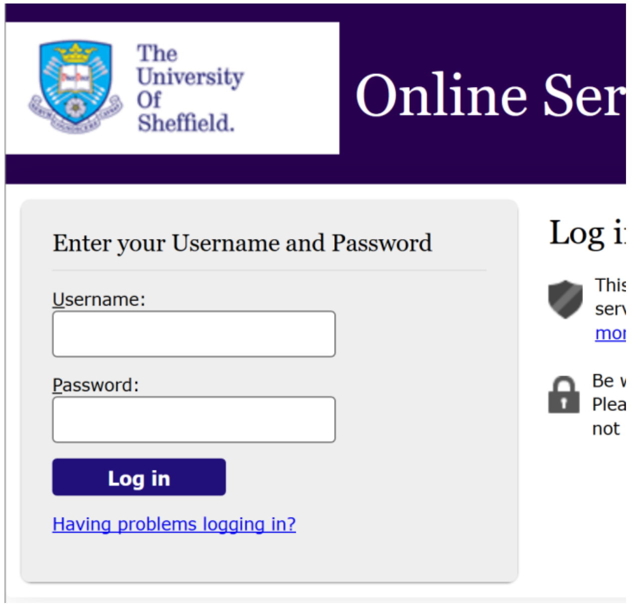
- HTTP is a stateless protocol
  - The HTTP server does not retain information about the user
  - How does server know if two requests are from same Web browser?
- Use HTTP Cookies to implement states or sessions



1. User browser requests a page on a website
2. Small piece(s) of data sent with page from website and stored in the user's web browser
3. Every time the user loads the website, the browser sends the cookie(s) back to the server
   - user's previous activity is known

Information: https://en.wikipedia.org/wiki/HTTP_cookie
Cookie picture by brainloc on sxc.hu (Bob Smith) (stock.xchng) [CC BY 2.5 (http://creativecommons.org/licenses/by/2.5)], via Wikimedia Commons

3.1 HTTP cookie uses


The University Of Sheffield.

Online Ser

Enter your Username and Password

Username:

Password:

Log in

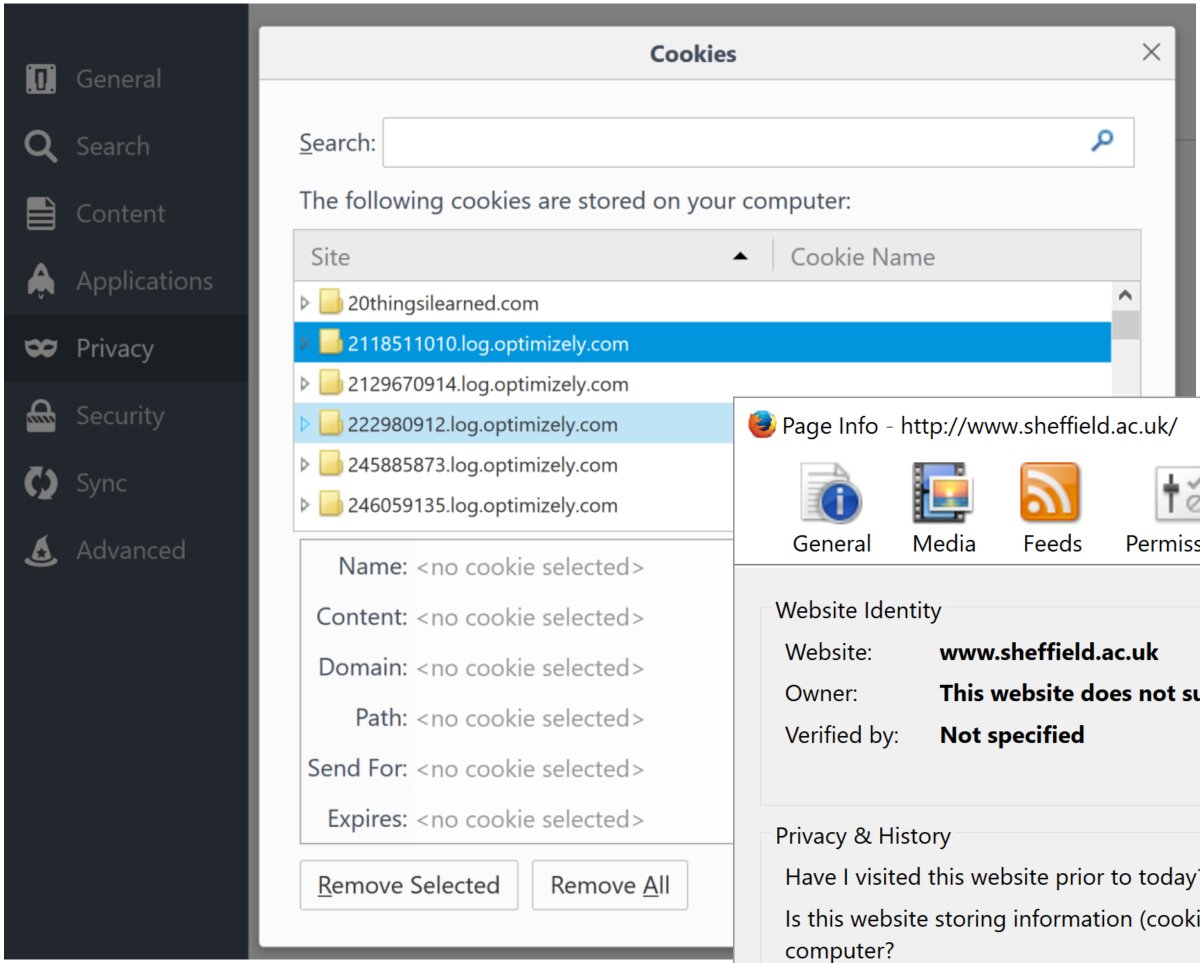
Having problems logging in?

Log i

This serv mor

Be v Plea not

- Session management
  - Originally, used for shopping baskets; now done on server
  - *Example*: website login page; cookie from server with session identifier; thereafter user granted access to services
- Personalization
  - Remember user preferences in form completion; autofill form fields
- Tracking
  - Track users' web browsing habits
- Different kinds of cookie have different lifespans
  - session, persistent, secure, HTTP-only, third-party
- EU cookie directive, 2002, 2009
  - Includes a policy requiring end users' consent for the placement of cookies



Cookies

Search:

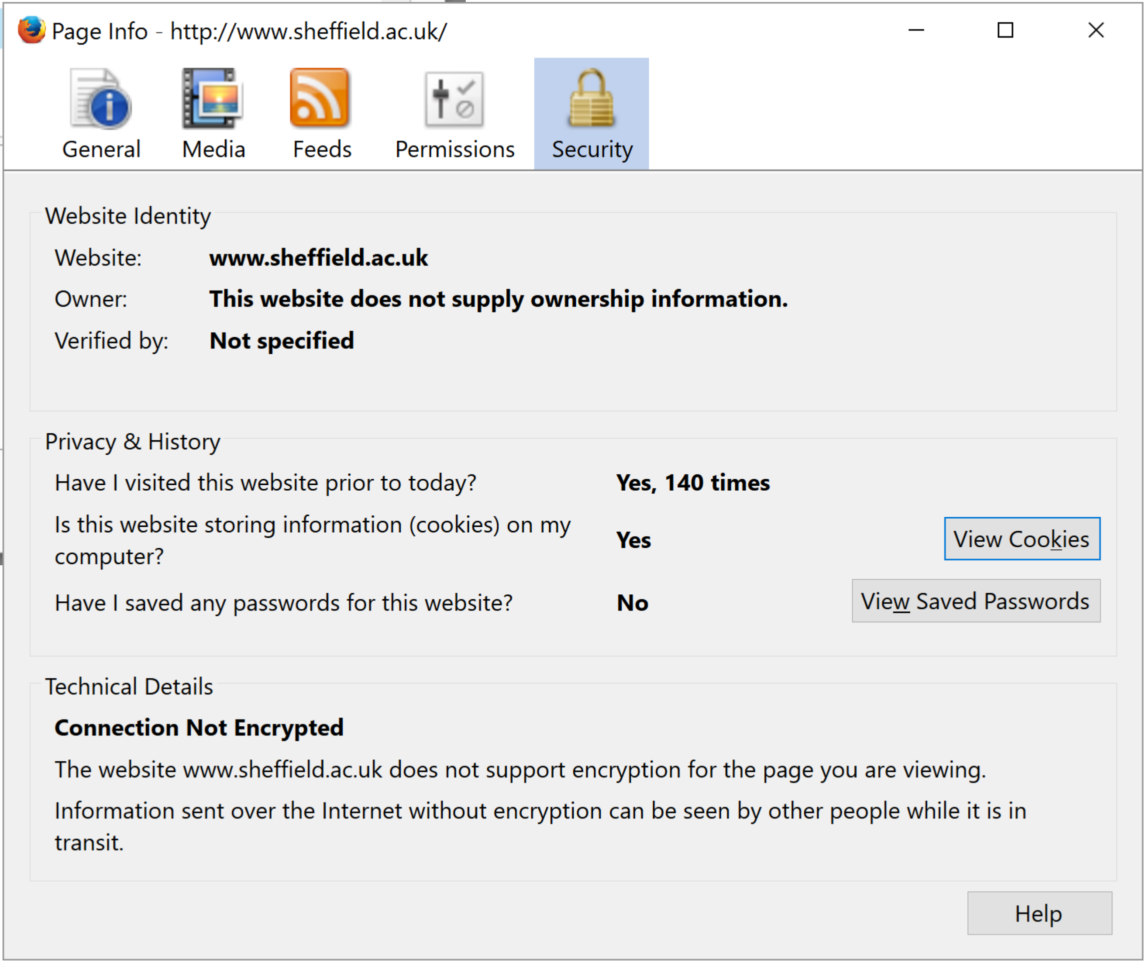
The following cookies are stored on your computer:

Site	Cookie Name
20thingsilearned.com	
2118511010.log.optimizely.com	
2129670914.log.optimizely.com	
222980912.log.optimizely.com	
245885873.log.optimizely.com	
246059135.log.optimizely.com	

Name:	<no cookie selected>
Content:	<no cookie selected>
Domain:	<no cookie selected>
Path:	<no cookie selected>
Send For:	<no cookie selected>
Expires:	<no cookie selected>

3.2 Firefox and cookies

View all your cookies


Page Info - http://www.sheffield.ac.uk/

Website Identity

Website:	**www.sheffield.ac.uk**
Owner:	**This website does not supply ownership information.**
Verified by:	**Not specified**

Privacy & History

Have I visited this website prior to today?	**Yes, 140 times**
Is this website storing information (cookies) on my computer?	**Yes**
Have I saved any passwords for this website?	**No**

Technical Details

Connection Not Encrypted

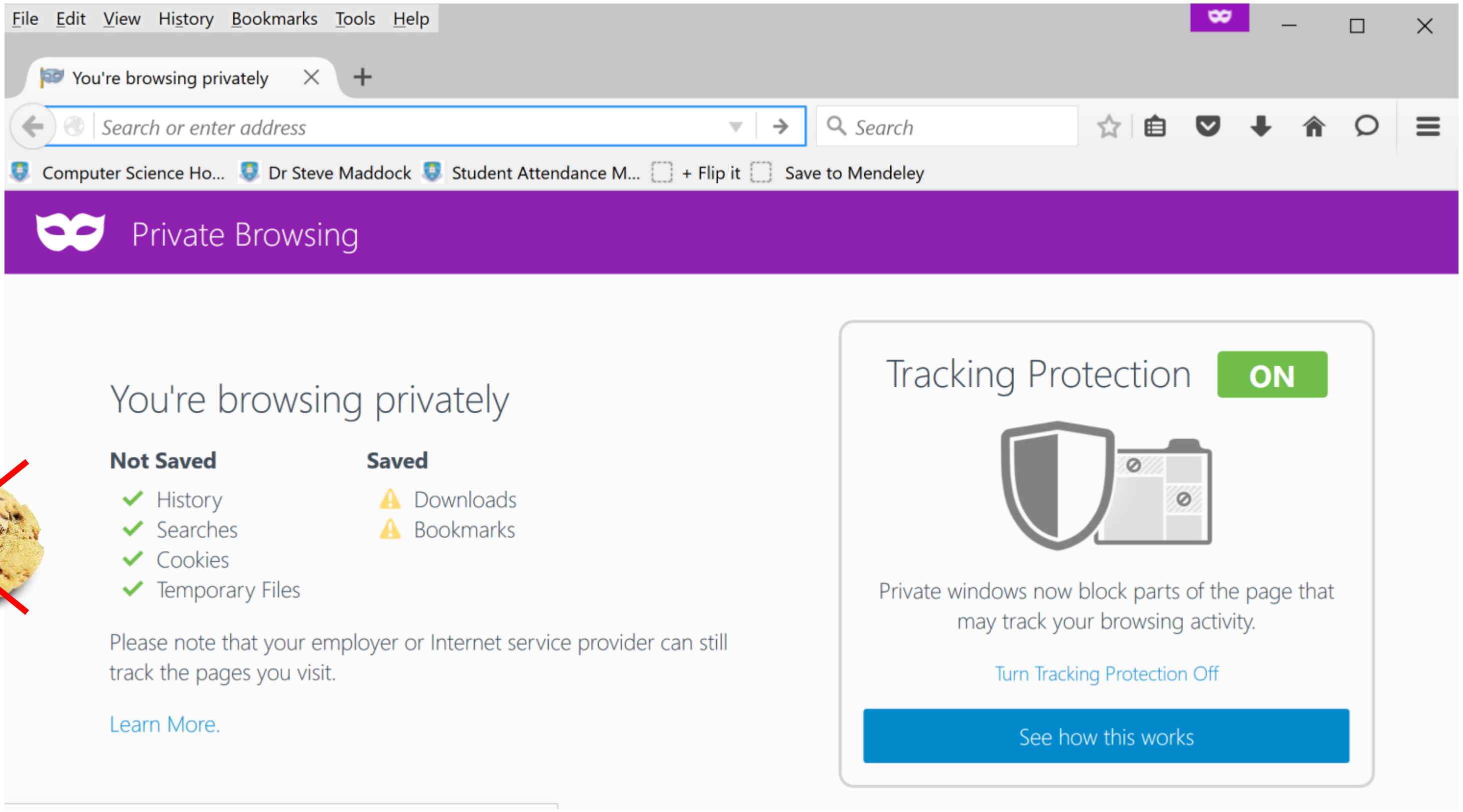
The website www.sheffield.ac.uk does not support encryption for the page you are viewing.

Information sent over the Internet without encryption can be seen by other people while it is in transit.

View cookies for a specific domain

3.3 Private browsing

- Most browsers these days support a privacy option called 'Incognito' or 'Private Browsing' mode



Private Browsing

You're browsing privately

Not Saved	Saved
✓ History	⚠ Downloads
✓ Searches	⚠ Bookmarks
✓ Cookies	
✓ Temporary Files	

Please note that your employer or Internet service provider can still track the pages you visit.

Learn More.

Tracking Protection ON

Private windows now block parts of the page that may track your browsing activity.

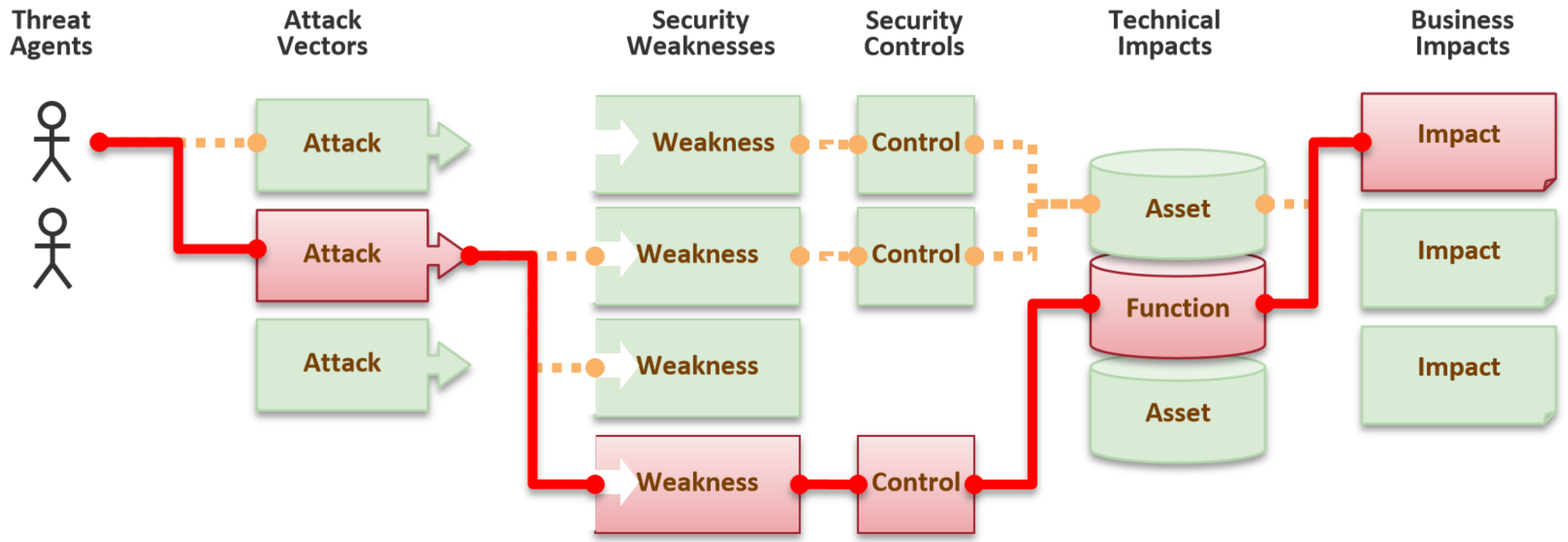
Turn Tracking Protection Off

See how this works

4. The most critical web application security flaws

“The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software.” [https://www.owasp.org]

- “The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are”



https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

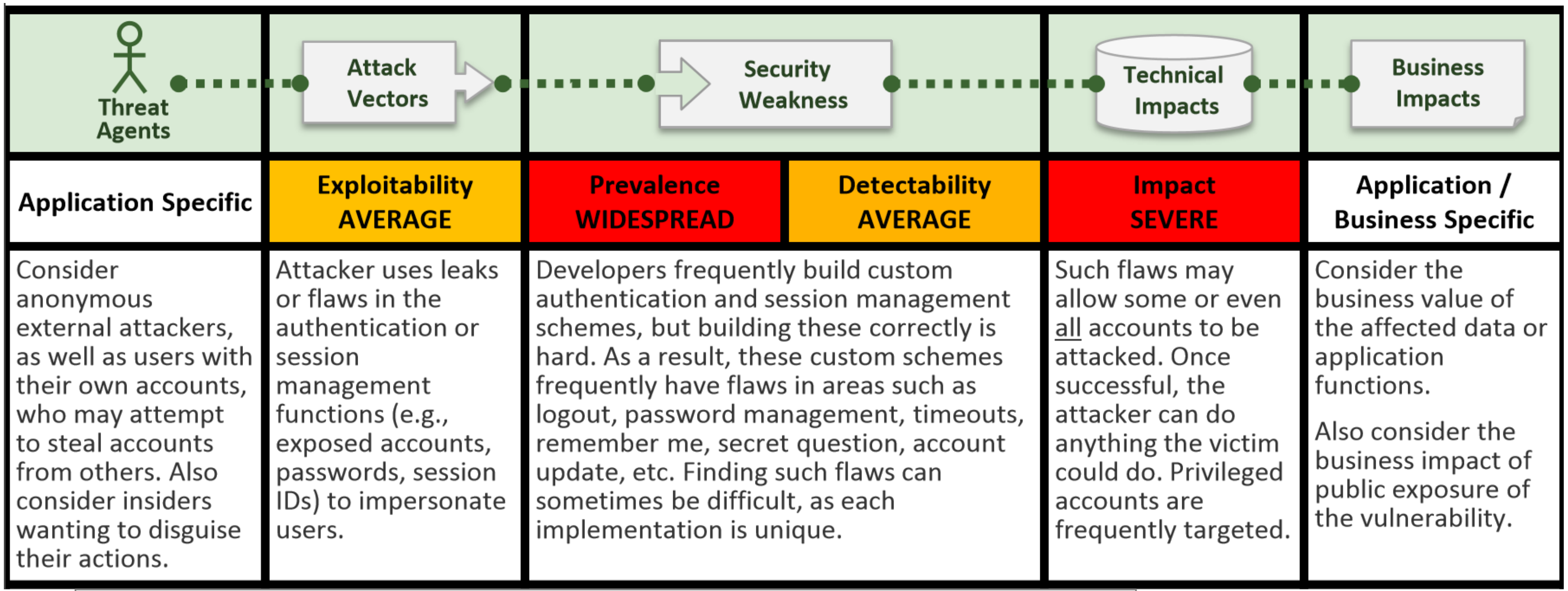
4.1 OWASP Top 10 – 2013

- A1 – Injection (e.g. SQL injection)
- A2 – Broken Authentication and Session Management
- A3 – Cross-Site Scripting (XSS)
- A4 – Insecure Direct Object References
- A5 – Security Misconfiguration
- A6 – Sensitive Data Exposure
- A7 – Missing Function Level Access Control
- A8 – Cross-Site Request Forgery (CSRF)
- A9 – Using Known Vulnerable Components
- A10 – Unvalidated Redirects and Forwards

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

4.2 A2 – Broken Authentication and Session Management

- *Example*: “Application’s timeouts aren’t set properly. User uses a public computer to access site. Instead of selecting “logout” the user simply closes the browser tab and walks away. Attacker uses the same browser an hour later, and that browser is still authenticated.”

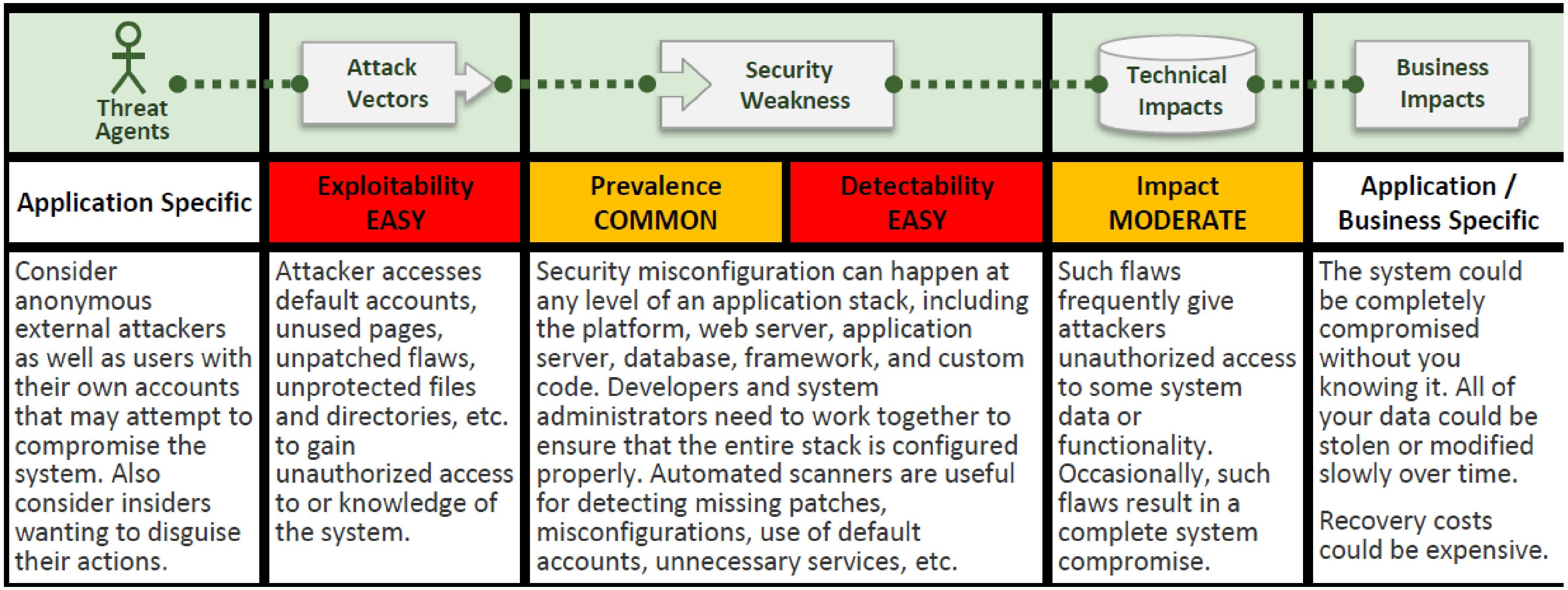


Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence WIDESPREAD**	**Detectability AVERAGE**	**Impact SEVERE**	**Application / Business Specific**
Consider anonymous external attackers, as well as users with their own accounts, who may attempt to steal accounts from others. Also consider insiders wanting to disguise their actions.	Attacker uses leaks or flaws in the authentication or session management functions (e.g., exposed accounts, passwords, session IDs) to impersonate users.	Developers frequently build custom authentication and session management schemes, but building these correctly is hard. As a result, these custom schemes frequently have flaws in areas such as logout, password management, timeouts, remember me, secret question, account update, etc. Finding such flaws can sometimes be difficult, as each implementation is unique.		Such flaws may allow some or even all accounts to be attacked. Once successful, the attacker can do anything the victim could do. Privileged accounts are frequently targeted.	Consider the business value of the affected data or application functions. Also consider the business impact of public exposure of the vulnerability.

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

4.3 A5 – Security Misconfiguration

Example: "If directory listing is not disabled on your server. Attacker discovers she can simply list directories to find any file. Attacker finds and downloads all your compiled Java classes, which she decompiles and reverse engineers to get all your custom code. She then finds a serious access control flaw in your application"

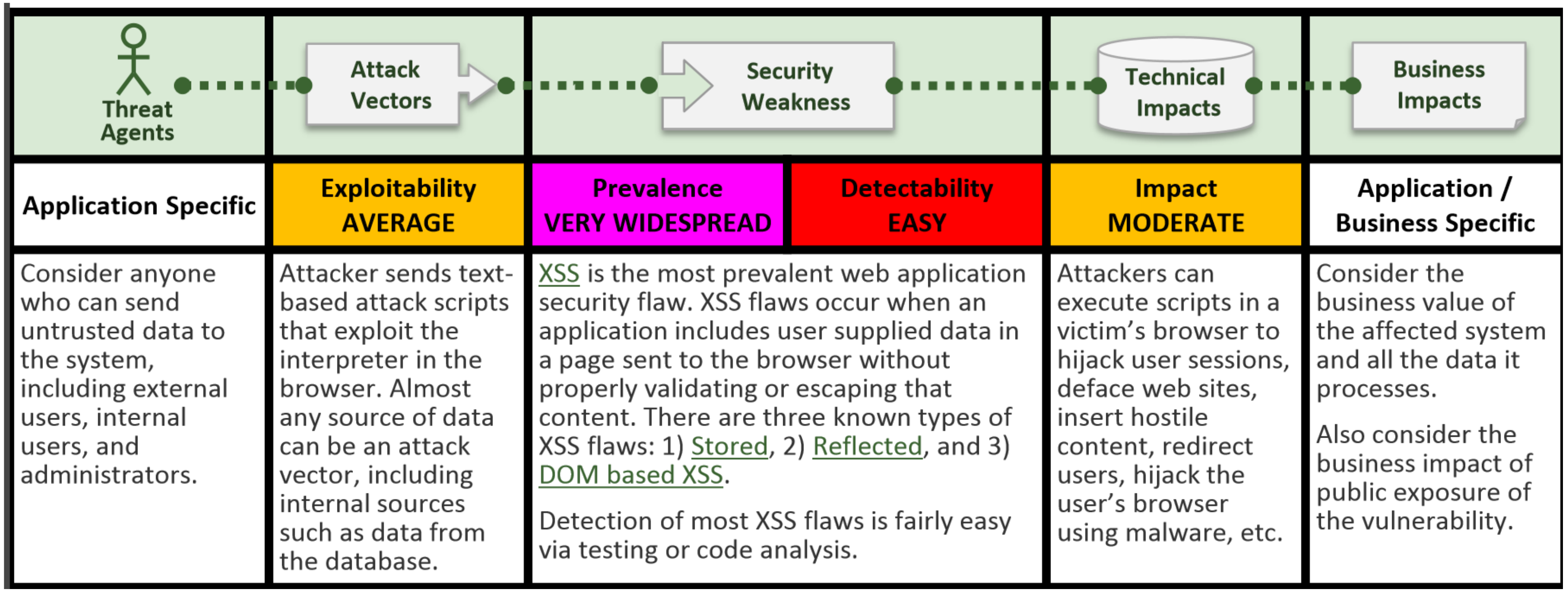


Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anonymous external attackers as well as users with their own accounts that may attempt to compromise the system. Also consider insiders wanting to disguise their actions.	Attacker accesses default accounts, unused pages, unpatched flaws, unprotected files and directories, etc. to gain unauthorized access to or knowledge of the system.	Security misconfiguration can happen at any level of an application stack, including the platform, web server, application server, database, framework, and custom code. Developers and system administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc.		Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise.	The system could be completely compromised without you knowing it. All of your data could be stolen or modified slowly over time. Recovery costs could be expensive.

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

5. A3 – Cross-Site Scripting (XSS)

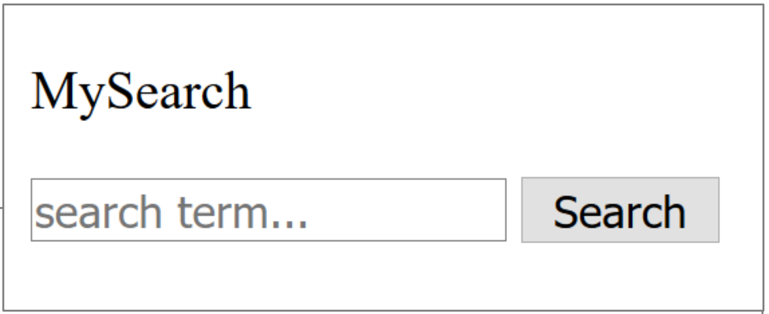
- Website accepts user input – user-supplied string may contain HTML and JavaScript (or carefully crafted img tags)
- If this is subsequently displayed on a Web page it could execute and send sensitive data to an attacker



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence VERY WIDESPREAD**	**Detectability EASY**	**Impact MODERATE**	**Application / Business Specific**
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS. Detection of most XSS flaws is fairly easy via testing or code analysis.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

5.1 XSS example



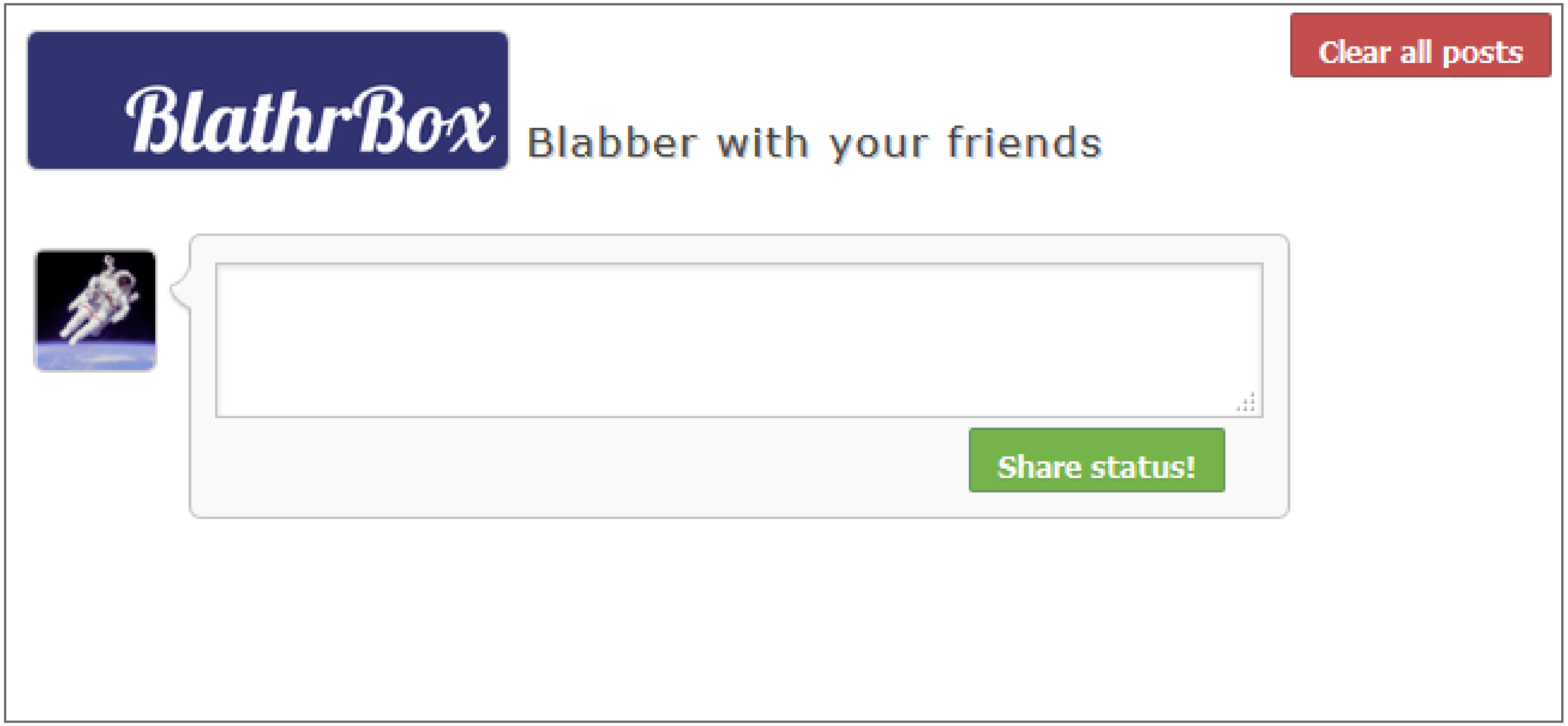
```
<body>
  <p>MySearch</p>
  <form action="javascript:search();" method="GET">
    <input id="q" name="q" placeholder="search term...">
    <input id="button" type="submit" value="Search">
  </form>

  <script>
    pageHeader=…; pageFooter=…;

    function search() {
      document.getElementById(q);
      var message = "sorry, no results found for " + q.value;
      message += " <a href='?'>Try again</a>."
      document.write(pageHeader+message+pageFooter);
    }
  </script>
</body>
```

- Input is reflected directly to the web page

Based on example at https://www.google.co.uk/about/appsecurity/learning/xss/ - need to open in Google chrome

Google example

Demo application 2:



https://www.google.co.uk/about/appsecurity/learning/xss/

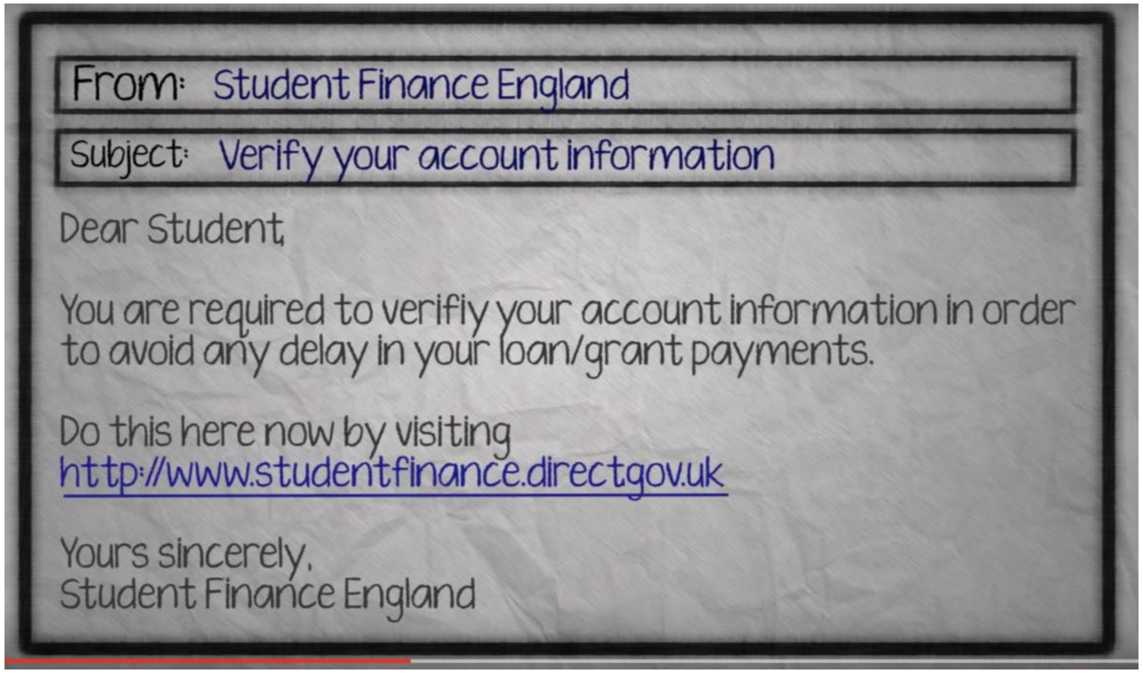
5.2 Possible XSS consequences

- Cookie theft
  - The attacker can access the victim's cookies associated with the website using document.cookie, send them to his own server, and use them to extract sensitive information like session IDs.
- Keylogging
  - The attacker can register a keyboard event listener using addEventListener and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as passwords and credit card numbers.
- Phishing
  - The attacker can insert a fake login form into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive information.

http://excess-xss.com/

5.3 Email - phishing

- "Phishing is the attempt to acquire sensitive information such as usernames, passwords, and credit card details... often for malicious reasons, by masquerading as a trustworthy entity in an electronic communication." https://en.wikipedia.org/wiki/Phishing



From: Student Finance England

Subject: Verify your account information

Dear Student,

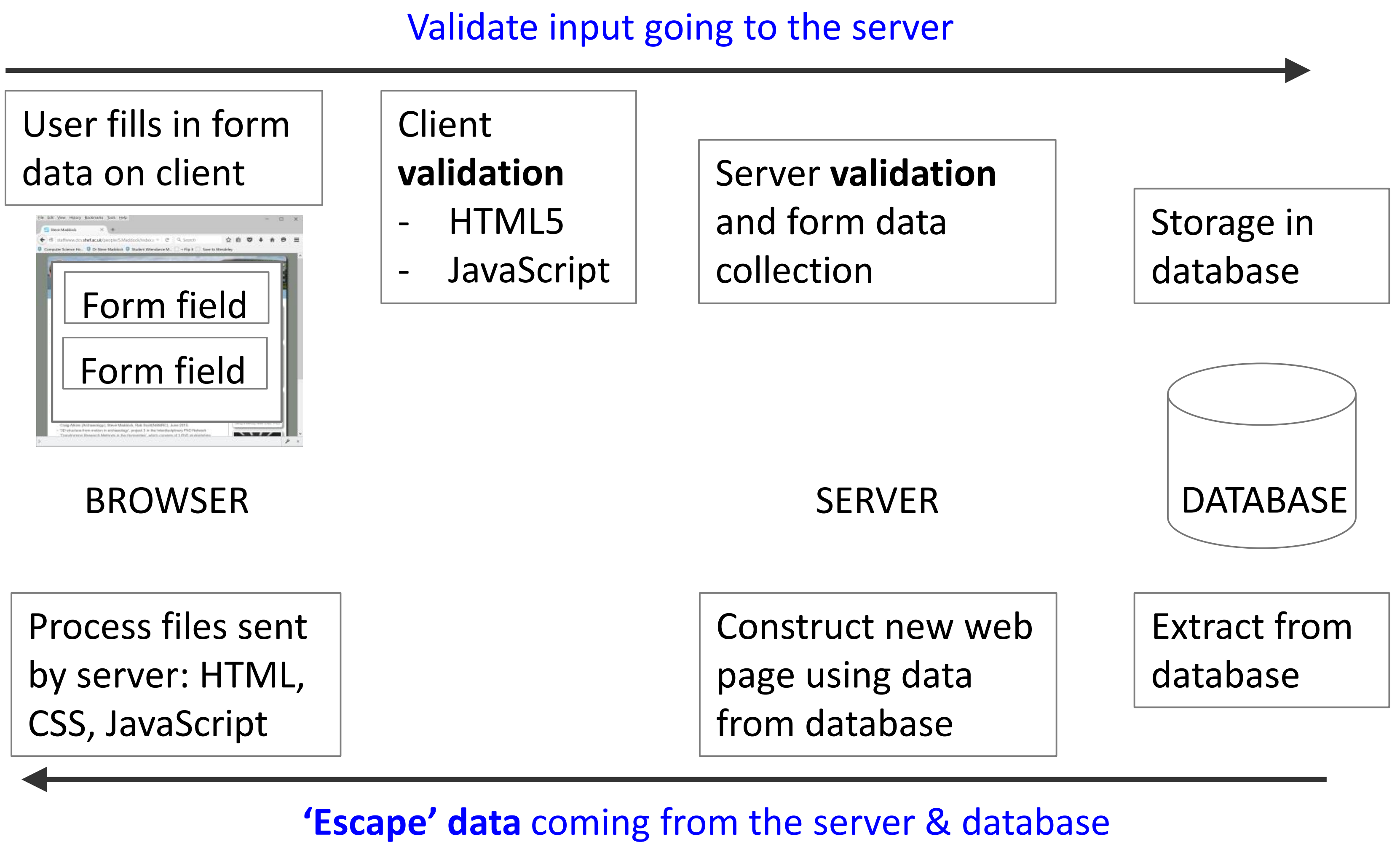
You are required to verifiy your account information in order to avoid any delay in your loan/grant payments.

Do this here now by visiting
http://www.studentfinance.directgov.uk

Yours sincerely,
Student Finance England

video - http://www.sheffield.ac.uk/cics/phishing

5.4 Defending against cross-site scripting


Validate input going to the server
User fills in form data on client
Form field
Form field
Client **validation**
- HTML5
- JavaScript
Server **validation** and form data collection
Storage in database
BROWSER
SERVER
DATABASE
Process files sent by server: HTML, CSS, JavaScript
Construct new web page using data from database
Extract from database
'Escape' data coming from the server & database

Jon Duckett. JavaScript & JQuery: Interactive Front-end Web Development, John Wiley & Sons, 2014

5.4 Defending against cross-site scripting

- ‘Escaping’ user content
    - Make sure every part of string is interpreted as a string primitive, not a control character
- *Example*: <script>alert('testing')</script>
- *Consider*: ‘&lt;’ is the HTML encoding for the ‘<‘ character
- *So use*: &lt;script&gt;alert('testing')&lt;/script&gt;
- *Display is*: <script>alert('testing')</script>
- *But it does not execute*.
- (Encoding rather than escaping.)

5.4 Defending against cross-site scripting

- Only add content from untrusted sources as text (not markup)
- Adding *user content* with JavaScript
  - DO use textContent or innerText;

```
document.getElementById("element").appendChild(
                              document.createTextNode(unsafeStr));
```

  - DO NOT use innerHTML

```
document.getElementById("element").innerHTML += unsafeStr;
```

5.4 Defending against cross-site scripting

- ‘Escaping’ user content
  - JavaScript function: escape(str) [deprecated – do not use anymore];
  - JavaScript function: encodeURI or encodeURIComponent

```
<script> // display as is
  var userInputA = "<p>test</p>";
  document.write("---"+userInputA+"---");
</script>
<script> // using escape
  var userInputB = "<p>test</p>";
  document.write("---"+escape(userInputB)+"---");
</script>
```

```
---

test

---
```

```
---%3Cp%3Etest%3C/p%3E---
```

5.5 FormMail.pl

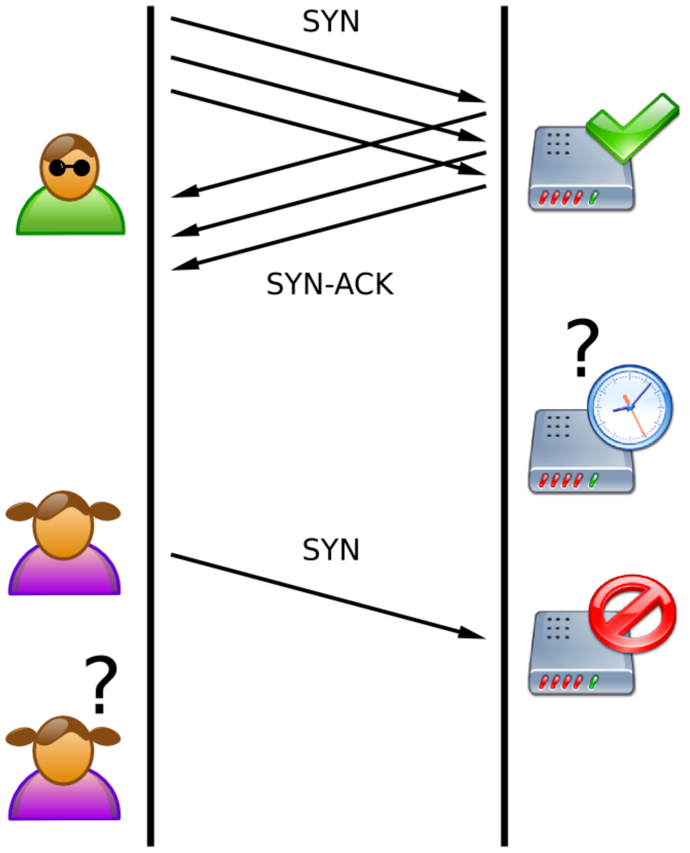
- "FormMail is a generic HTML form to e-mail gateway that parses the results of any form and sends them to the specified users." [http://www.scriptarchive.com/formmail.html]
- www.scriptarchive.com/readme/formmail.html#history
  - Version 1.0 06/11/95 - This script was created.
  - ...
  - Version 1.92 04/21/02 - Removed cross-site scripting vulnerabilities by converting all <, >, & and " into their HTML equivalents when displayed on a web page. These characters are left intact in the e-mail message.
  - Version 1.93 07/14/09 - Removed cross-site scripting and header injection/ http response splitting vulnerabilities from redirect and return_link_url fields.

No longer in OWASP top 10

6. DoS – Denial of service attack

Network example:

- Make a machine or network resource unavailable to its intended users
- *Example*: SYN flood
  - TCP three-way handshake: client SYN; server SYN-ACK; client ACK
  - server saturated keeping track of bogus connections so legitimate users cannot connect
- Distributed DoS (DDoS) – lots of attack sources



7. Summary

- HTTPS is a secure transmission protocol
- Cookies enable stateful communication between server and client browser
- XSS enables attackers to inject client-side script into web pages viewed by other users
- A DoS attack makes a machine or network resource unavailable to its intended users
- University online security information
  - Information: https://www.shef.ac.uk/cics/security
  - Policy: https://www.shef.ac.uk/cics/policies/infosecpolicy
  - Course: 'Protecting Information' at https://infosecurity.shef.ac.uk/

- *Next lecture*: cryptography