Contents

1 Cookies **2**
- 1.1 Types of Cookie . . . 2
- 1.2 Third Party Cookies . . . 2
- 1.3 Cookie Vulnerabilities . . . 2

2 SSL/TLS **2**
- 2.1 TLS . . . 2
- 2.2 TLS handshake . . . 2
- 2.3 TLS Vulnerabilities . . . 2

3 Cross-site Scripting (XSS) **3**
- 3.1 Reflected XSS . . . 3
- 3.2 Persistent XSS . . . 3
- 3.3 Preventing XSS . . . 3

1 Cookies

- HTTP is a stateless protocol, but most of what we do online is stateful
- Cookies are small text files used to provide persistence
- Servers can provide cookies during HTTP responses, using Set-Cookie
- Browsers will return any cookies for a given domain in GET and POST requests

1.1 Types of Cookie

- Session – **Deleted when the browser exits**, contain no expiration date
- Persistent – Expire at a given time
- Secure – Can only be used over HTTPS
- HTTPOnly – Inaccessible from JS

1.2 Third Party Cookies

- Cookies are associated with the domains that produced them. Amazon.com cookies don't go to google.co.uk
- Some websites include request to other domains, such as 3rd party advertisers. These serve cookies – a lot

1.3 Cookie Vulnerabilities

- How a website uses a cookie is up to the server
- Many create an SID to authenticate users, for example to "keep me logged on"
- Obtaining this cookie – Cookie Stealing – lets you hijack their session
  - HTTP Cookies can be stolen simply by monitoring
  - HTTPS will require Cross-site scripting attacks or DNS poisoning

2 SSL/TLS

- There are dangers associated with sending plain text cookies, passwords etc.
- SSL, and the newer TLS provide authenticated and encrypted sessions
- Secure Socket Layer (SSL) came first, then after v3.0 it became Transport Layer Security (TLS), currently v1.2 / v1.3

2.1 TLS

Transport Layer Security has two layers:

- The Record Layer: using established symmetric keys and other session info, will encrypt application packets, very like IPSec
- The Handshake Layer: used to establish session keys, as well as authenticate either party – usually the server using a public-key certificate

2.2 TLS handshake

Look at notes

2.3 TLS Vulnerabilities

- The man-in-the-middle vulnerabilities are usually countered using public-key authentication
- The majority of TLS problems are implementation: Heartbleed
- Protocol downgrade attacks are still a concern – many servers still allow weak cipher suites. FREAK and Logjam force the use of 512-bit keys

3 Cross-site Scripting (XSS)

- A type of injection attack, similar in many ways to an SQL Injection
- HTML is read by a browser, and is a combination of content (text) and structure (html tags)
- If we can inject html structures into the content of a website, the browser will simply execute these – e.g. ¡script¿ tags

3.1 Reflected XSS

A malicious URL that inserts an exploit directly into the page returned by a server

3.2 Persistent XSS

- Even worse, no need to trick people into clicking links
- Any website that doesn't properly sanitise html tags from user input is vulnerable
- Blog posts with comment sections are obvious targets, but there are many more
- Forums, web comments, shopping reviews, etc

3.3 Preventing XSS

- Websites must aggressively escape HTML characters from any user input / output
- You also need to find all of the bizarre obfuscated versions of XSS
- When you consider all of the things people input on interactive websites, this can be a real problem

3.4 Cross-site request forgery

- When a user puts in an HTTP request, they will also send any relevant session cookies. E.g. an SID from having logged in
- If the user has already authenticated, a malicious URL can then perform some action on their account

3.5 Cross-site Request Forgery (XSRF)

- When a user puts in an HTTP request, they will also send any relevant session cookies. E.g. an SID from having logged in
- If the user has already authenticated, a malicious URL can then perform some action on their account

3.6 XSRF in POST

- Most websites use POST, this is little defence
- The phishing email just points to a convincing website with a malicious form on it

3.7 Preventing XSRF

- XSS vulnerabilities make XSRF a lot easier! Fix these!
- Use synchroniser tokens
- Each website form has a one-time token that the server validates when the form is submitted

Reference section

placeholder