

Cybersecurity – EC521

Web Security

Manuel Egele
PHO 337
megele@bu.edu
Boston University

Web Application Security

- When an organization puts up a web application, they invite everyone to send them HTTP requests.
- Attacks buried in these requests sail past firewalls without notice because they are inside legal HTTP requests.
- Even “secure” websites that use SSL just accept the requests that arrive through the encrypted tunnel without scrutiny.
- This means that your web application code is part of your security perimeter!

Web Application Security

- The security issues related to the Web are not new. In fact, some have been well understood for decades
 - For a variety of reasons, major software development projects are still making these mistakes and jeopardizing not only their customers' security, but also the security of the entire Internet.
 - There is no “silver bullet” to cure these problems. Today's assessment and protection technology is improving, but can currently only deal with a limited subset of the issues at best.
 - To address the security issues, organizations will need to change their development culture, train developers, update their software development processes, and use technology where appropriate.

Why is it Important?

- Easiest way to compromise hosts, networks and users (Remember Equifax?)
- Widely deployed
- No Logs! (POST Request payload)
- Difficult to defend against or to detect
- Firewall? What firewall? I don't see any firewall...
- Encrypted transport layer does not help much

~~News~~ Hacks From The Field

Equifax got hacked

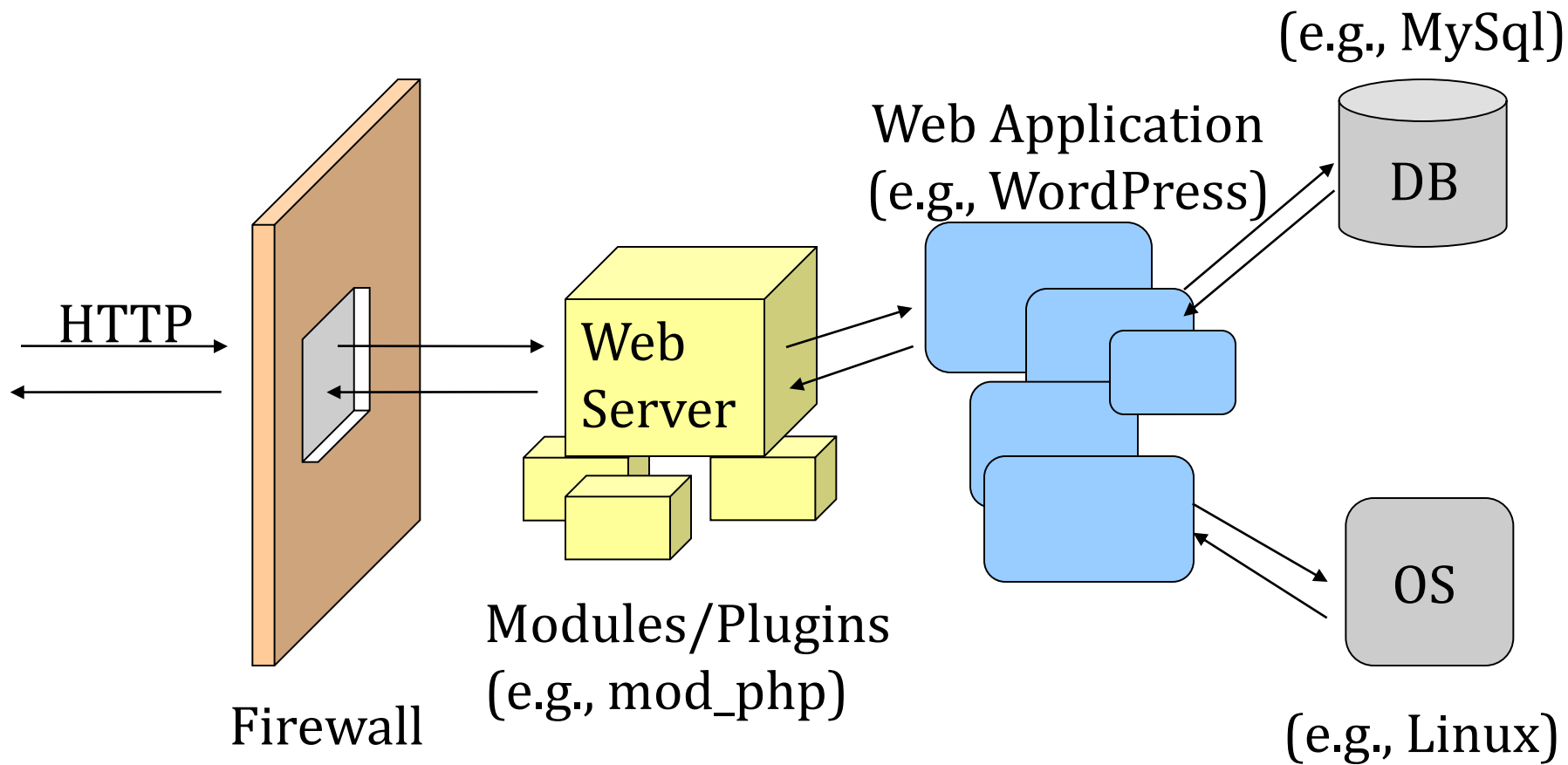
- One of the big three credit bureaus
- Has credit history and PII on almost all Americans (lost data of 143M) and many foreigners (e.g., millions of Brits)
- Vulnerability in Apache Struts framework (CVE-2017-9805) *remote code execution*
- Security vulnerabilities happen and hopefully get fixed but ...

Big Problem: Vulnerability was known *and* patches released two months before Equifax got hit! i.e., They didn't update (negligent!)

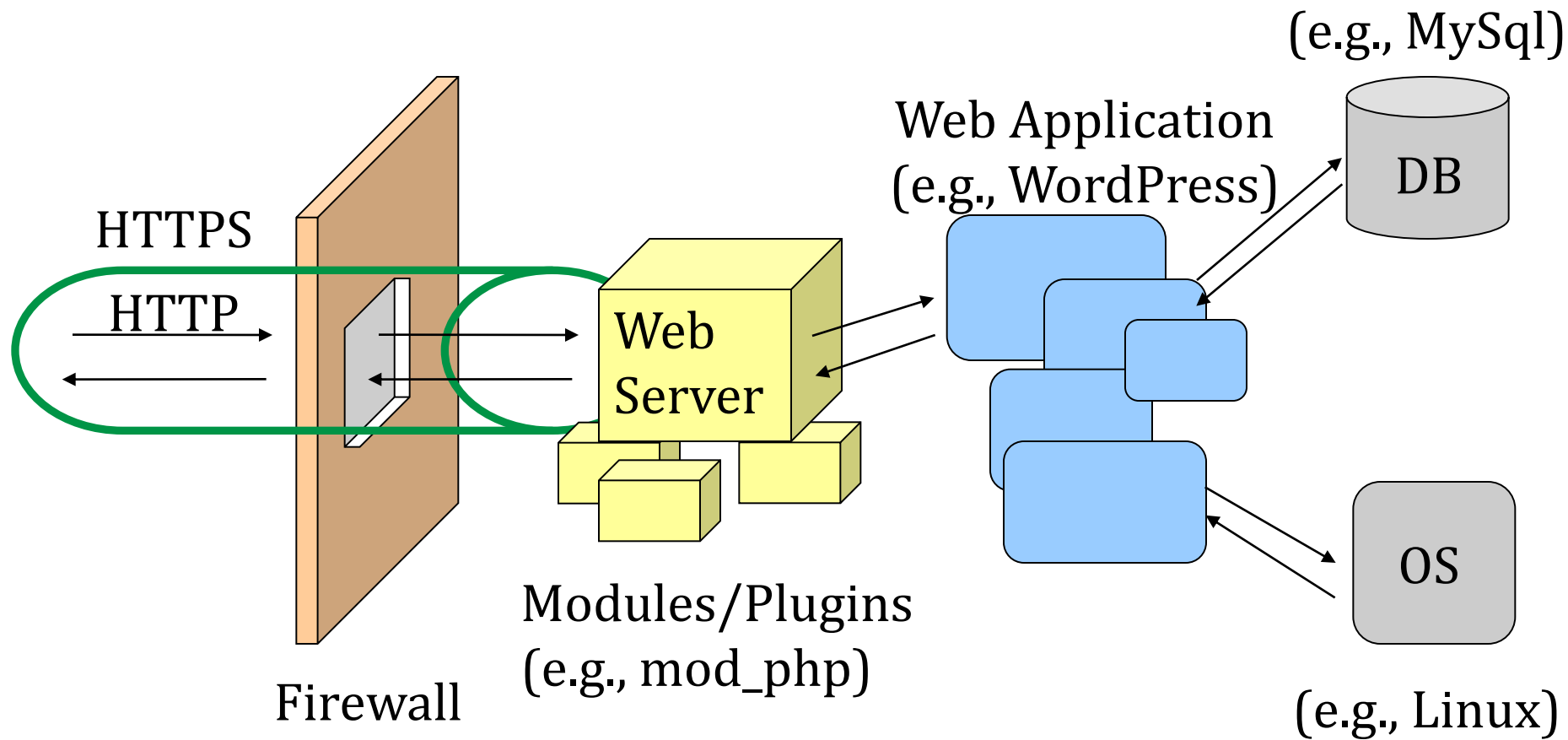
Why is it Important?

- Easiest way to compromise hosts, networks and users (Remember Equifax?)
- Widely deployed
- No Logs! (POST Request payload)
- Difficult to defend against or to detect
- Firewall? What firewall? I don't see any firewall...
- Encrypted transport layer does not help much

On a Typical Web Server



On a Typical Web Server (w/ https)



On a Typical Web Server...

Your host has open 80/8080 port (firewall)

Following components are running

- OS
- Web Server
 - Main application (e.g., Apache)
 - Plugins (e.g., mod_php, mod_perl)
 - Servlets
 - Scripts (CGI, Perl, php, ...)

HTTP

- The web is built on the Hypertext Transfer Protocol (HTTP)
 - (Originally) text-based and stateless
- Messages have a header & an optional body
 - Header: request method, response status, resource paths, versions, key-value pairs
 - Body can contain request parameters and resource contents

HTTP Request Example

Method Resource Version (Virtual) host

GET / HTTP/1.1

Host: www.reddit.com

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0)...

Accept: text/html,application/xhtml+xml,application/xml...

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Cookie: __cfduid=d32...

DNT: 1

Connection: keep-alive

Browser version

Accepted languages
/encoding

Do not track

Cookies

Connection type

HTTP Reply Example

Protocol version Status message Resource

↓ ↓ ↓

Status code MIME type, charset

HTTP/1.1 200 OK text/html; charset=UTF-8

Date: Mon, 29 Sep 2014 20:44:50 GMT

Content-Type: text/html; charset=UTF-8

Transfer-Encoding: chunked ← Encoding type

Connection: keep-alive ← Connection type

x-ua-compatible: IE=edge

x-frame-options: SAMEORIGIN ← Frame options

x-content-type-options: nosniff

x-xss-protection: 1; mode=block ← Enable anti-XSS filter

Vary: Accept-Encoding

X-Moose: majestic

cache-control: max-age=0

CF-Cache-Status: HIT

Expires: Mon, 29 Sep 2014 20:44:50 GMT

Server: cloudflare-nginx ← Server name

CF-RAY: 171b055df54901ee-EWR

Content-Encoding: gzip ← Content encoding

HTTP Methods

- GET** Retrieve resource at given path
- HEAD** Identical to GET, but response omits body
- POST** Submit data to a given path, might create resources at new paths
- PUT** Submit data to a given path, creating or modify resource *at that path*
- DELETE** Deletes resource at a given path
- TRACE** Echoes request
- OPTIONS** Returns supported HTTP methods given a path
- CONNECT** Creates a tunnel to a given network location

Demo

<http://www.bu.edu>

On a Typical Web Server

