Cyber Security

Introduction and Demonstration

Andrew Alexander
Software Engineer/Cloud Architect, Capital One

Follow along

https://github.com/andrewalexander/talks/tree/master/cyber-security

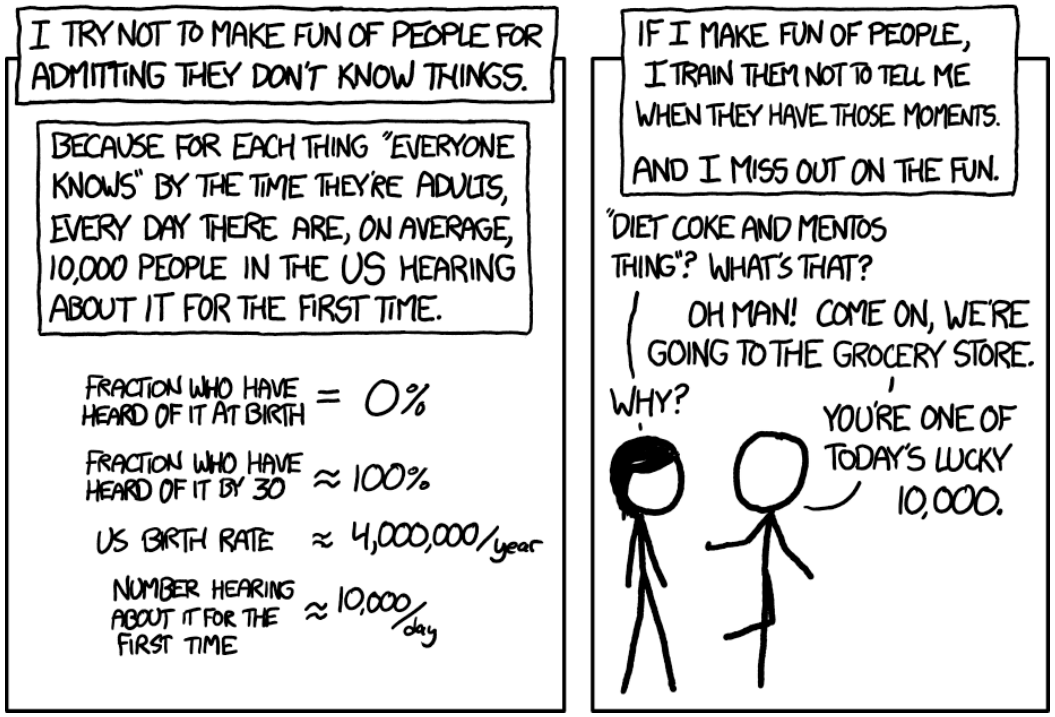
Who am I?

- Front of desk - sw engineer, architect, evangelist
- Buzz word love/hater
- Coffee addict/aficionado
- Craft beer enthusiast (bourbon & wine too)
- Conspiracy theory appreciator
- Golang lover

Point of this talk - "Learn about security"

- Understand why the internet is inherently insecure (hint: history)
- Gain a deeper appreciation for the branches of cyber security
- See common mistakes/understandable assumptions when building web applications (But more importantly how to fix/prevent them)
- "L3g1t H4x0rs" - Demoing some attacks with tools designed for this purpose - educational/training tools
- Learn what "enough security" means
- Don't trust users, especially not their input

My favorite xkcd/my view on learning


I TRY NOT TO MAKE FUN OF PEOPLE FOR ADMITTING THEY DON'T KNOW THINGS.
BECAUSE FOR EACH THING "EVERYONE KNOWS" BY THE TIME THEY'RE ADULTS, EVERY DAY THERE ARE, ON AVERAGE, 10,000 PEOPLE IN THE US HEARING ABOUT IT FOR THE FIRST TIME.
FRACTION WHO HAVE HEARD OF IT AT BIRTH = 0%
FRACTION WHO HAVE HEARD OF IT BY 30 ≈ 100%
US BIRTH RATE ≈ 4,000,000/year
NUMBER HEARING ABOUT IT FOR THE FIRST TIME ≈ 10,000/day
IF I MAKE FUN OF PEOPLE, I TRAIN THEM NOT TO TELL ME WHEN THEY HAVE THOSE MOMENTS.
AND I MISS OUT ON THE FUN.
"DIET COKE AND MENTOS THING"? WHAT'S THAT?
OH MAN! COME ON, WE'RE GOING TO THE GROCERY STORE.
WHY?
YOU'RE ONE OF TODAY'S LUCKY 10,000.

Saying 'what kind of an idiot doesn't know about the Yellowstone supervolcano' is so much more boring than telling someone about the Yellowstone supervolcano for the first time.

Security in a nutshell

My definition:

Ensuring that any/all systems involved in electronic communication are only used by the **intended entities** and only in the way they were **intended to be used**

- Intended entity

```
Bank website - can only see your accounts
Chat program - can only access your chats
Voting machines - only the intended human who claims to be casting the vote
```

- Intended use

```
Website signs user up for event; shouldn't be able to delete the entire database
Website transfers money from account A to B; subject to balance restrictions,
    no negative amounts, etc.
App is intended to control IoT toaster, takes over home security/automation system/network
Web form - box for phone number: no letters... box for address: validate it's a real
    address. Don't trust user input
```

Parts of cyber security

- Parts you expect

```
Cryptography
Access control/Identity Management
Web application penetration testing
"Big Brother" monitoring
```

- Parts you may not...

```
Physical Penetration Testing - People breaking into buildings
Business Continuity Planning
 - Disaster Recovery
 - Playbooks for worst case scenarios/doomsday
 - Flood/Fire plans
Risk Acceptance
```

- Like operations: when it works well, you don't know it's there. When it doesn't, you wonder where it's been.

Various roles in security

- Vulnerability testing - automated and manual
- Network infrastructure building/monitoring - "Big brother"

```
- Perimeter monitoring/fortification
- IDS/IPS - Intrusion Detection/Prevention Systems - Host-based (HIDS) and Network-based (NIDS)
- Logging all the things
- Compiling audit data from all the logs
```

- Incident Response/Forensics
- Architecture/Processes
- Information Security Officers/Consulting - advise, not fix

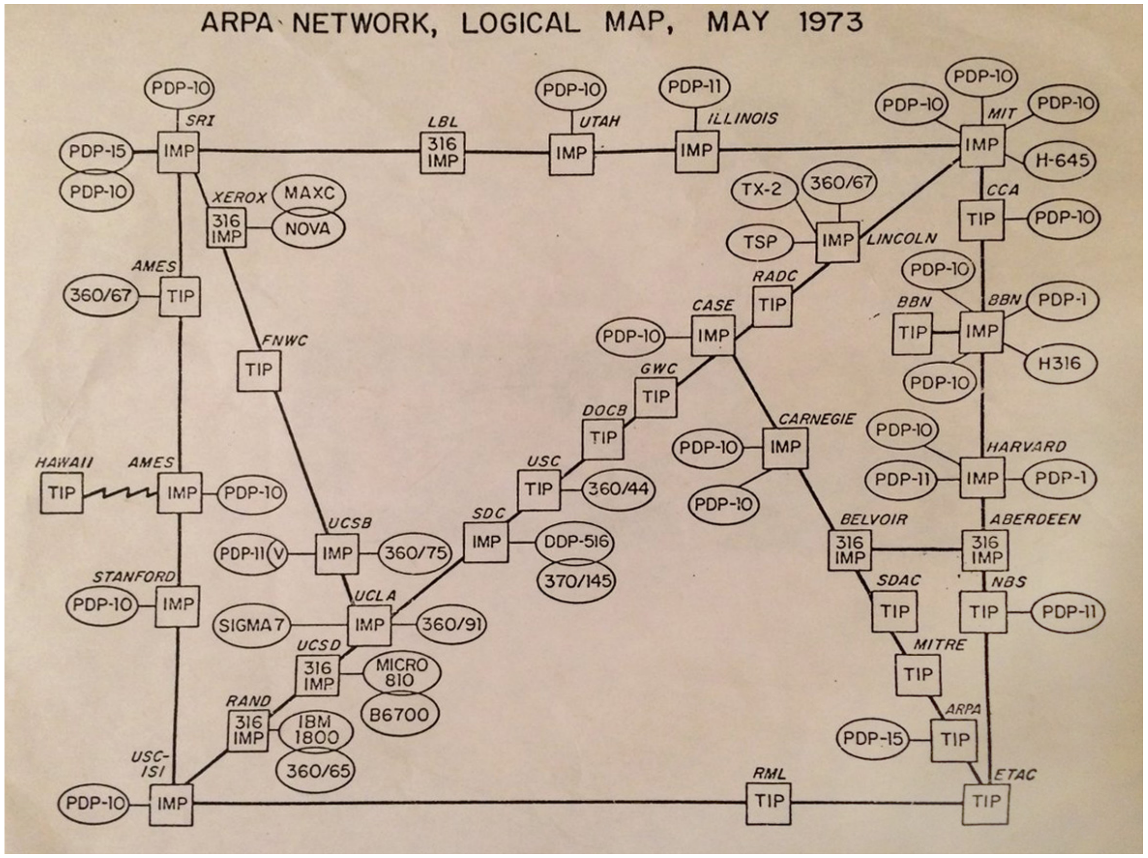
Sources of risk

Ultimately, everything comes down to risk and priorities. Not all risk can be mitigated. Some must be accepted. Challenge is finding the amount of acceptable risk - "enough security"

Systems are more than just the software. All people, technology, and processes compose a system. All components of the system introduce risk.

- Modern web applications (more later)
- Humans
- Weather - Natural Disasters
- Incorrect assessments/acceptance of too much risk
- Humans
- Software updates
- Also Humans

History of the internet/why it is insecure



History of the internet/why it is insecure (cont'd)

- ARPANET - guy had three terminals in his office, other ends connected to computers in Berkeley, Santa Montica, and MIT; had to get up to other terminal to talk to specific people in each office... He had a better idea.
- DARPA was also researching a network that could survive nuclear war - half the nodes could be wiped out, but the network of remaining computers could still communicate
- "Trust" was done through access to the physical links. If you were able to connect to the secure physical lines, you belonged there
- DARPANET grew from there... Later becoming the Internet
- That trust model never changed. Anyone can access anything. As designed.
- All "Security" we have (especially in IPv4) is bolted on, not secure by design

Technology/Security Words and Stuff

We live in a sea of buzzwords, and I tend to use terms interchangably or incorrectly and confuse people...
Mostly a brain dump of some security concepts that we will be showing off in more detail later
---
API - Application Programming Interfaces, commonly used to provide complex services/tasks with a simple contractual interface
Front-end - Part of application the end-user interfaces with. Responsible for translating and sending user input to the back-end
Client - Usually an alias for front-end, but can also refer to the end user using the web application
Back-end - Part of application the front-end interfaces with. Responsible for handling front-end input and parsing request, usually resulting in calls to databases, other servers, or external/third-party APIs
Server - Usually an alias for back-end, but can also refer to the actual machines running code, likely on x86 hardware

Definitions/Acronyms (cont'd)

Authentication - Who you are; one of:

```
1) what you know (password)
2) what you are (biometrics)
3) what you have (pre-registered device/token)
```

Two Factor/Multi-factor Auth (2FA/MFA) - combines 2 of the above, usually 1) and 3)
Authorization - What you're allowed to do
HTTP - Hypertext Transfer Protocol - application-level (layer 7) responsible for most browser-based web application traffic
HTTPS - Secure HTTP; adds Public Key Infrastructure to HTTP. This provides authentication/validation of the server's identity and (once it passes the checks), begins encrypting all traffic between the client and server.
SSL/TLS - Often used interchangably, but refer to distinct protocols - Secure Sockets Layer or Transport Layer Security. Responsible for HTTPS traffic; the protocols provide the means for authentication and encryption of traffic and rely on both public key cryptography (and corresponding infrastructure) and symmetric cryptography to function.

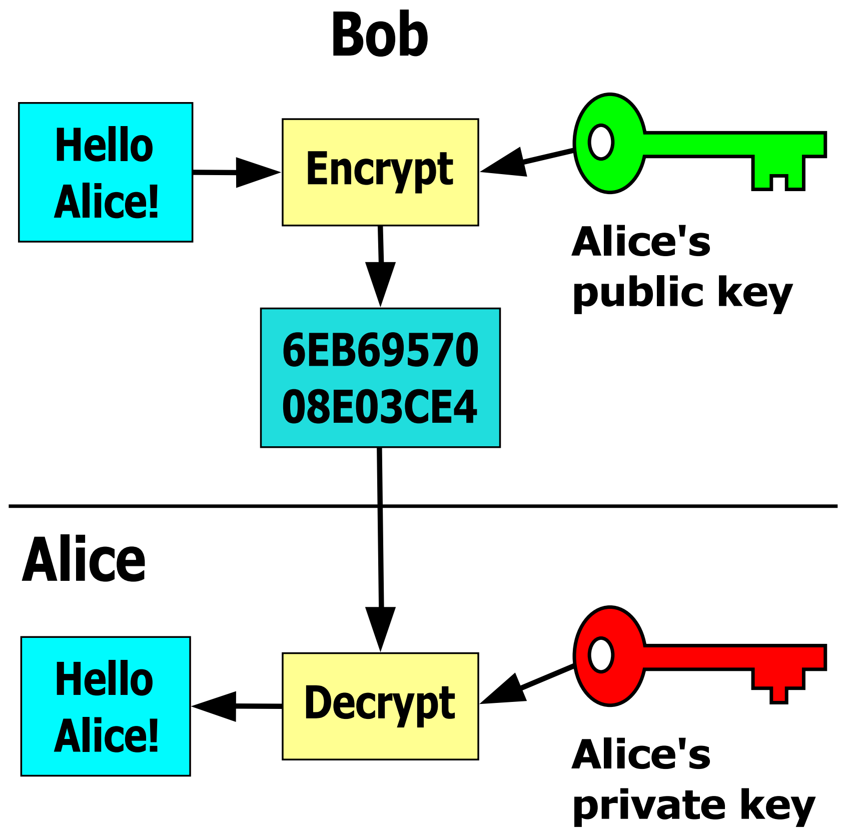
Definitions/Acronyms (cont'd)

Encryption - Applying a cryptographic operation with a piece of plaintext and a cryptographic key as inputs, producing ciphertext as output.
Decryption - Applying a cryptographic operation with a piece of ciphertext and a cryptographic key as inputs, producing plaintext as output.
Public Key Cryptography - The use of two distinct, yet cryptographically related keys to both encrypt data in one direction and validate the sender of a signed piece of data. Commonly comes with a public and private component, either referred to as keys or certificates. Also referred to as **Asymmetric Cryptography**
Symmetric Cryptography - The use of a single cryptographic key for both encryption and decryption
MITM - Man in the middle [attack]; when a "man" is in between a client and server such that the client believes it is communicating with the real server and the server believes it is communicating with the real client. HTTPS makes this incredibly difficult to achieve without compromising the client's browser or Certificate Authority (CA)

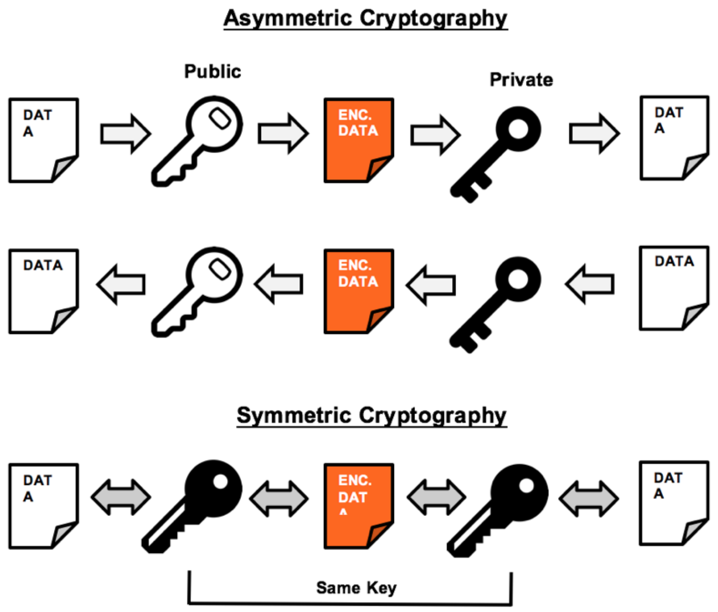
Definitions/Acronyms (cont'd)

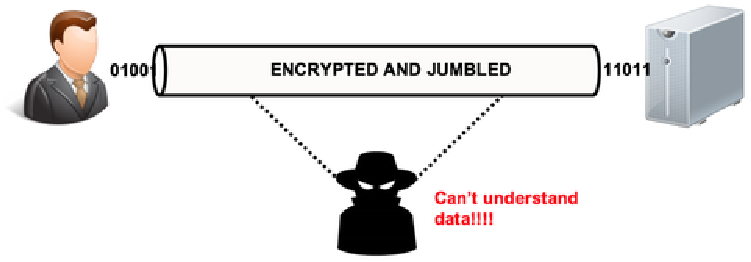
Client-side encryption - Encrypting a payload before sending it through a network. If sending through HTTPS, HTTPS further encrypts around the client-side encrypted data, rendering even the most complex man in the middle ineffective.
Signing - Applying a cryptographic operation with a piece of plaintext and a cryptographic key as inputs, producing a cryptographic signature around the plaintext as output. It is important to note this still leaves the message payload intact. If the payload was not previously encrypted, it should be treated as plaintext. This cryptographic signature can be validated with a separate cryptographic operation that can vary, but usually involves a cryptographic operation with a public key, assuming a private key was used to sign the message (most common).
XSS - Cross-site scripting; executing code on a client or server that the application writers did not intend to be executed.
CSRF - Cross-site request forgery; making a server believe a fraudulent request came from a trusted client.
Session Hijacking - stealing a valid user's cookies or other session tokens after they log in and using them to create an otherwise-valid session.

Public Key Cryptography

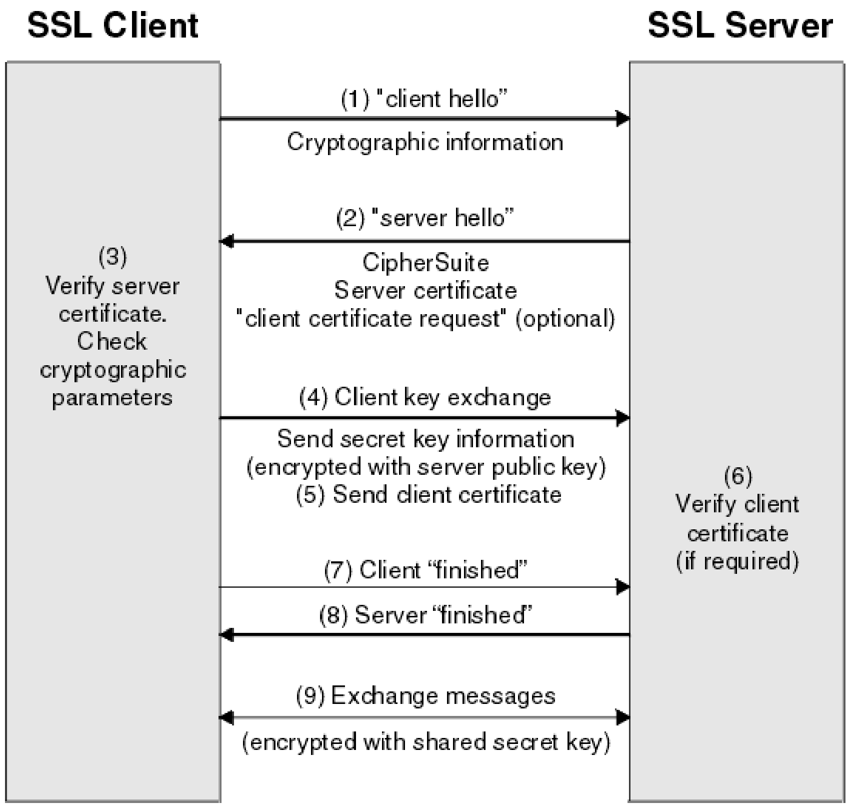


Asymmetric vs. Symmetric encryption





TLS basics


SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server certificate.
Check cryptographic parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

SSL uses both asymmetric and symmetric encryption. Asymmetric to prove identity of server/client, symmetric for data sent during the session

Modern Web Applications - Biggest Offender

Modern applications have a ton going on - one vulnerability in any part means the entire application is compromised

- Rich, logic-heavy front-ends, often in the form of thick clients

```
Mobile phone apps
Chrome web applications
Video Game Console Apps
```

- Large, often distributed back-end systems

```
Absurd amounts of APIs - requests routinely come/go from other sites/servers
CORS, anyone?
```

- Can run on insecure or compromised Protocols/Networks

```
HTTP vs. HTTPS
Entities decrypting SSL traffic - man in the middle (MITM)
  - Can be good or bad... companies monitoring associates' laptops vs. oppressive governments
```

- Cookies - can be used for tracking or session hijacking

Cookies

- Computerphile has two amazing videos on this:

  www.youtube.com/watch?v=LHSSY8QNvew (https://www.youtube.com/watch?v=LHSSY8QNvew)

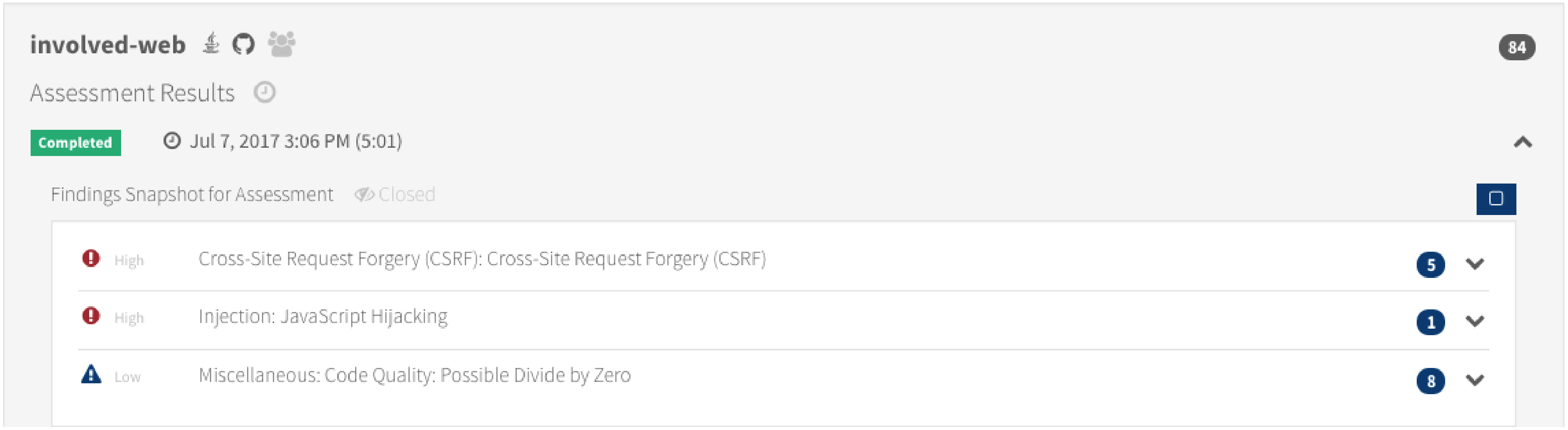
  www.youtube.com/watch?v=T1QEs3mdJoc (https://www.youtube.com/watch?v=T1QEs3mdJoc)

- Cookies can only be set/retrieved by the domain in which they were meant; Google can't see Facebook's cookies
- Websites can add code to make a request to Google, Facebook, Amazon, or any other company that sells targeted advertisements
- When that request is made, Google, Facebook, Amazon, etc. can see that it's the same person, sets an identical cookie
- Simple demo

What we will demonstrate today

Underprotected web applications/APIs in the form of:
- XSS
- CSRF
- Injection (SQL and otherwise)
- (maybe) tracking cookies

Goals:
- show why input sanitization is important
- show how code that most of us have written introduces real risks/vulnerabilities



involved-web 84

Assessment Results

Completed Jul 7, 2017 3:06 PM (5:01)

Findings Snapshot for Assessment Closed

High	Cross-Site Request Forgery (CSRF): Cross-Site Request Forgery (CSRF)	5
High	Injection: JavaScript Hijacking	1
Low	Miscellaneous: Code Quality: Possible Divide by Zero	8

Warnings/Disclaimers

WARNING:
It should go without saying, but hacking is **illegal**.

These demonstrations are just that - *demonstrations*. They are occurring on systems intentionally set up to be compromised, and as they are owned by me - I gave the express permission to "hack" them. If you do some of these things to REAL websites, **you can and will be charged to the fullest extent of any / all applicable laws**.

DISCLAIMER:
I may have cheated a little on the OWASP Juice Shop stuff - great write up on all them on this website: https://incognitjoe.github.io/hacking-the-juice-shop.html

XSS - Cross Site Scripting

Game from Google Application Security: https://xss-game.appspot.com

Level 1:
Script tags; hmmm....

Level 2:
Bit tougher; Need to close out the existing tags.

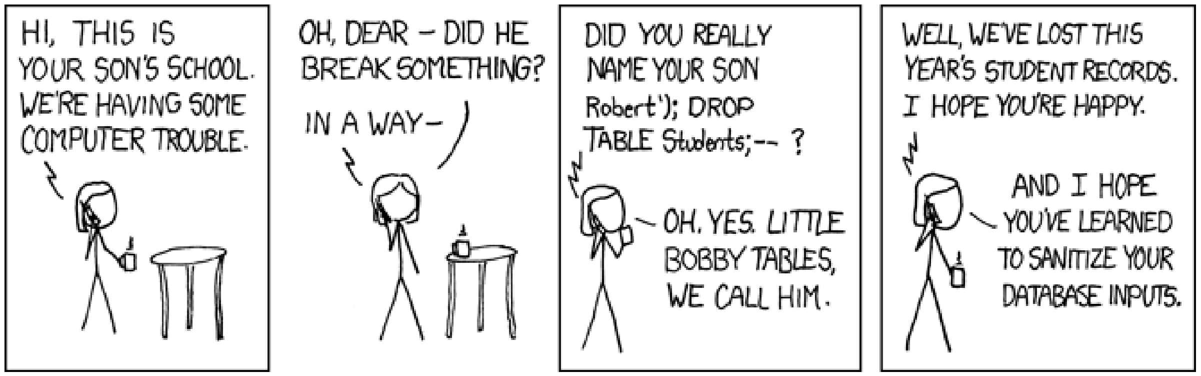
But... `<script>alert('got em')</script>` won't work here
Check out image tags and the `onerror` property 😉

Levels 3-6 escalate even further - save for later!

SANITIZE YOUR INPUT!

SQL Injection

Taking advantage of unsanitized user input


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR — DID HE BREAK SOMETHING?
IN A WAY—
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Her daughter is named Help I'm trapped in a driver's license factory.

SQL Injection demonstrated

OWASP Juice Shop - Retrieve a list of all user credentials via SQL injection

0) Find places we know will execute SQL...
1) Panic when you realize it isn't pulling the right table
2) Feel awesome when you learn/remember `UNION` SQL statements
3) Get credentials!

CSRF - Cross Site Request Forgery

OWASP Juice Shop - Change Bender's password into SlurmCl4ssic without using SQL Injection

0) SQL Inject to log in as Bender (yes, it's SQLi but this doesn't count ;))
1) Go to Change Password... Oh no! we need current password... or do we?
- Can do some URL manipulation!
- This is *technically* XSS if done on this website; we can do better
2) Craft and get victim to click bad link...
3) CSRF just happened and Bender's pw is now changed! (profit)

Some fun tools/links

Pentester Lab - Actual vulnerabilities/CVEs in the wild demonstrated with VMs and walkthroughs (https://pentesterlab.com/exercises/)

Hacksplaining - Learn to hack/protect yourself (https://www.hacksplaining.com/)

OWASP Zed Attack Proxy (ZAP) (https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)

Kali Linux (https://www.kali.org/downloads/)

- has a lot of tools preinstalled/configured, many of which typically only have source code installers

Metasploit and (https://www.metasploit.com/)

Metasploitable (https://sourceforge.net/projects/metasploitable/files/Metasploitable2/)

SQLmap - automatic SQL injection/takeover penetration testing tool (http://sqlmap.org/)

More links

- Intentionally vulnerable web apps/servers

Over the Wire - Wargames (http://overthewire.org/wargames/)

OWASP WebGoat (https://github.com/WebGoat/WebGoat/wiki/Running-WebGoat)

Damn Vulnerable Web Application (DVWA) (http://www.dvwa.co.uk/)

Juice Shop (what we used today) (https://github.com/bkimminich/juice-shop)

- Educational videos

Computerphile (https://www.youtube.com/user/Computerphile)

```
- So many good videos. XSS, CSRF, Password Cracking, 2FA, Encoding/Error Correction, Stacks, etc.
```

Tom Scott (https://www.youtube.com/user/enyay)

- Frequently on Computerphile; his standalone channel also has some good stuff (electronic voting machine video for sure)

Thank you

Andrew Alexander
Software Engineer/Cloud Architect, Capital One
andrew.alexander@capitalone.com (mailto:andrew.alexander@capitalone.com)
https://andrewalexander.io (https://andrewalexander.io)
https://github.com/andrewalexander (https://github.com/andrewalexander)
@andrewalex1992 (http://twitter.com/andrewalex1992)