JavaScript Security

What You Need to Know to Build Secure Web Apps

Jared Smith
Cyber Warfare Research Team
Oak Ridge National Laboratory

@jaredthecoder
jaredmichaelsmith.com

How are you?!

“All the programs that will ever be needed have been written.”

–COMPUTER SCIENCE PROFESSOR FROM **[REDACTED]** UNIVERSITY IN 1982

About me

- Security Researcher and Software Engineer at Oak Ridge National Laboratory in the Cyber Warfare Research Team
- Formerly Software Security Engineer at Cisco System's Advanced Security Initiatives Group
- Senior at UTK in Computer Science
  - Founded HackUTK and VolHacks
  - Published in ACM and IEEE journals
- CTO of Prometheus Group, building a travel security application based on methods from the US Special Forces

Storytime!

A (slightly exaggerated) life of a dev

Lead Developer: Boss, we've been building this application for weeks. We think we're finally at 1.0 and the client likes the preliminary results. Even all of our continuous integration tests are passing! Can we launch it?!

Manager: Let me check with the PM and the CTO and I'll get back to you ASAP.

[at all-hands meeting the following day]

CTO: We've been waiting for this day for a long time…blah blah blah buzz word blah blah blah…blah blah buzz word blah… LAUNCH IT NOW!

Lead Developer: […scurrying to keyboard…types `git push origin release`…anxious waiting…terminal returns `TESTS PASSED. V. 1.0 DEPLOYED TO RELEASE BRANCH`…]

Slack Bot: 🎉🎉🎉

6 MONTHS

LATER...

*[…massive leak of **credit card info and poorly hashed passwords**…hackers breached internal systems as well]*

Slack Bot: SEND 25 BITCOINS TO WALLET 1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2 TO GET YOUR BOT BACK - L337 HAX0r 5quAD


alex

Baby steps...

1) Understand the problem

World's Biggest Data Breaches

Selected losses greater than 30,000 records

(updated 15th Nov 2016)

interesting story

YEAR | BUBBLE COLOUR: YEAR | METHOD OF LEAK | BUBBLE SIZE: NO OF RECORDS STOLEN | DATA SENSITIVITY | SHOW FILTER

latest

2016

2015

2014

2013

2012

Brazzers

ClixSense

Code.org

Interpark

Mail. r

25, 000, 000

Mine

Mutuelle Generale de la Police

Red Blooc

Privatization Agency of the Republic of Serbia

egram

World Check

We

43000000

VK

100, 544, 93-

Anthem

80, 000, 000

Friend Finder Network

412, 000, 000

MySpac

164, 000, 000

Philippin Commissi on Election

55, 000, 000

Syrian government

Wendy's

Turkish izenship atabase

49, 611, 709

TalkTalk

US Office of Personnel Management

Verizon

National Childbirth Trust

Sanr

ack

US C.. of Personnel Management (2nd Breach)

Adult Friend Finder

British

vs

Linux Ubur forur

Premera

Australian Immigration Department

Carefirst

IRS

Voter Database

191, 000, 000

Hacking Tea

Securus echnologies

70, 000, 000

uTorrent

AshleyMadison.co

CarPhone Warehous

Dominios Pizzas (France)

Experi T-m

Invest Bank

omtech

MSpy

European Central Bank

Neiman Marcus

VTech

Community Health Systems

Japan Airlines

a Credi ureau

Staples

Uber

AOL

2, 400, 000

Ebay

145, 000, 000

Home Dep

Yahoo

500000000

Twitch.tv

NASD

New York Taxis

Advoca Medic Group

D&B, Altegrity

Florida Courts

Morga Chase

76, 000, 000

MacRumou

Mozilla

UPS

Target

70, 000, 000

ssndob.ms

Citigroup

Sony Pictures

Vodafone

Adobe

36, 000, 000

Everno

50, 000, 000

Crescent Health Inc., Walgreens

Indian Univ

Florida Department of Juvenile Justice

Kirkwood Community College

Living ocial

000, 000

Nintendo

)V Scribd

UbiSoft

eHarmony, Last.fm

California Departm of Ch Supp Servic

Central Hudson Gas & Electric

Ap

upa

Facebook

Dropbox

Kissinger Cables

NMBS

Medica

TerraCom & YourTel

Tumblr

65, 000, 000

Yahoo Japan

Twitter

Global

Source: informationisbeautiful.net


SSLv3

And it's a **hard** problem.

Vulnerabilities Can Be Everywhere

TCP/IP Model	Protocols and Services	OSI Model
Application	HTTP, FTP, Telnet, NTP, DHCP, PING, DNS, BGP, JSON-RPC, SOAP, Exchange ActiveSync, IMAP, LDAP, POP, RTP, SIP, VOIP, SMTP, SSH, TLS/SSL, WebSockets, DDP, Tor, STOMP, RDP, NFS, MQTT, IRC BitTorrent, Bitcoin, SOCKS, SPDY, ASCII, UTF-8, MIME, AFP, PFIF, XML-RPC	Application
		Presentation
		Session
Transport	TCP, UDP, PPTP	Transport
Network	IPv4, IPv6, ICMP, IGMP, IPSEC	Network
Network Interface	MAC (Ethernet, DSL, ISDN, FDDI), 802.11 (Wi-Fi), ARP, ATM, LLDP, NDP, PPP, OSPF, L2TP	Data Link
		Physical

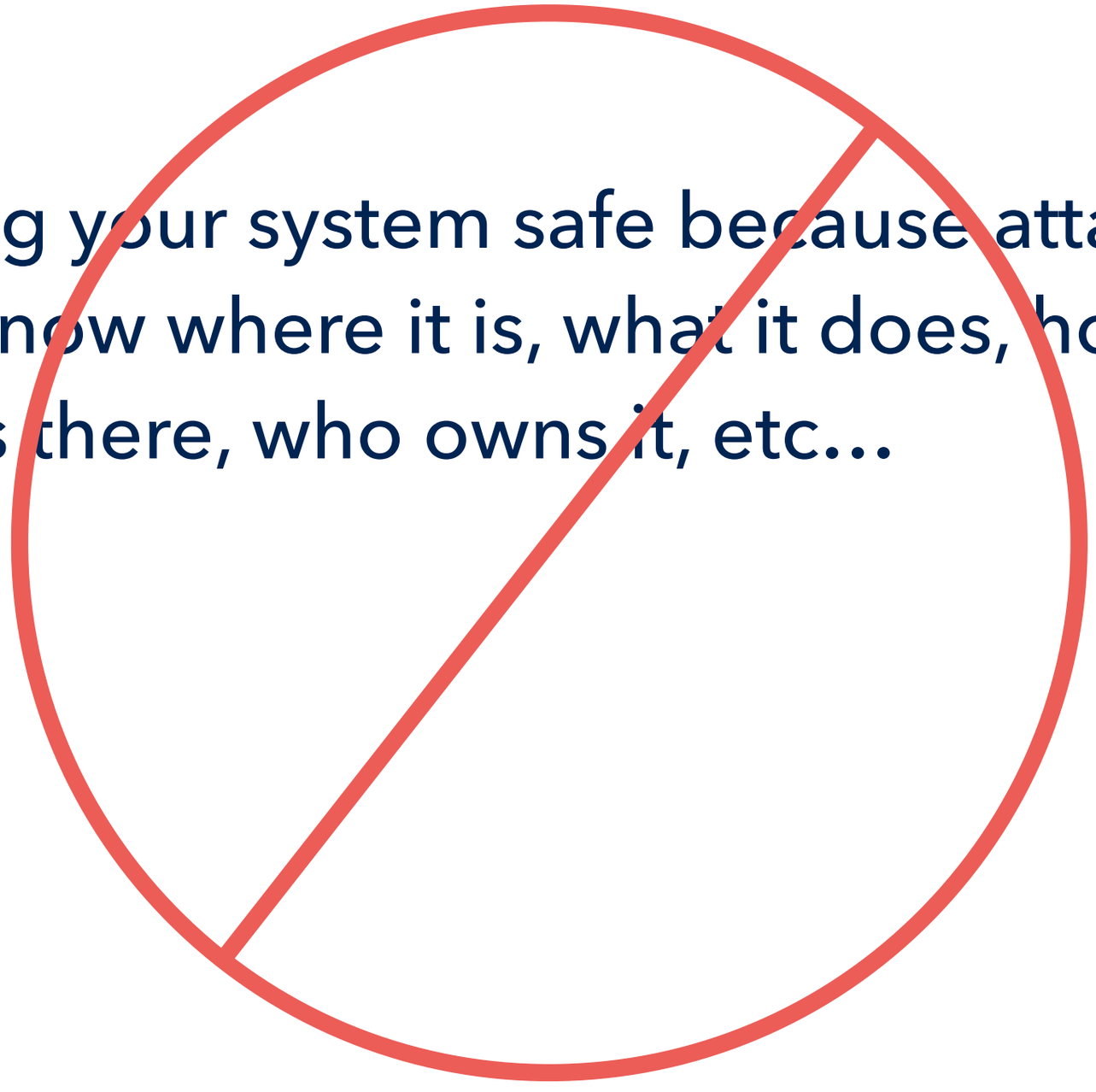
But there is hope.

Cue epic Star Wars music...

2) Understand the **right** way

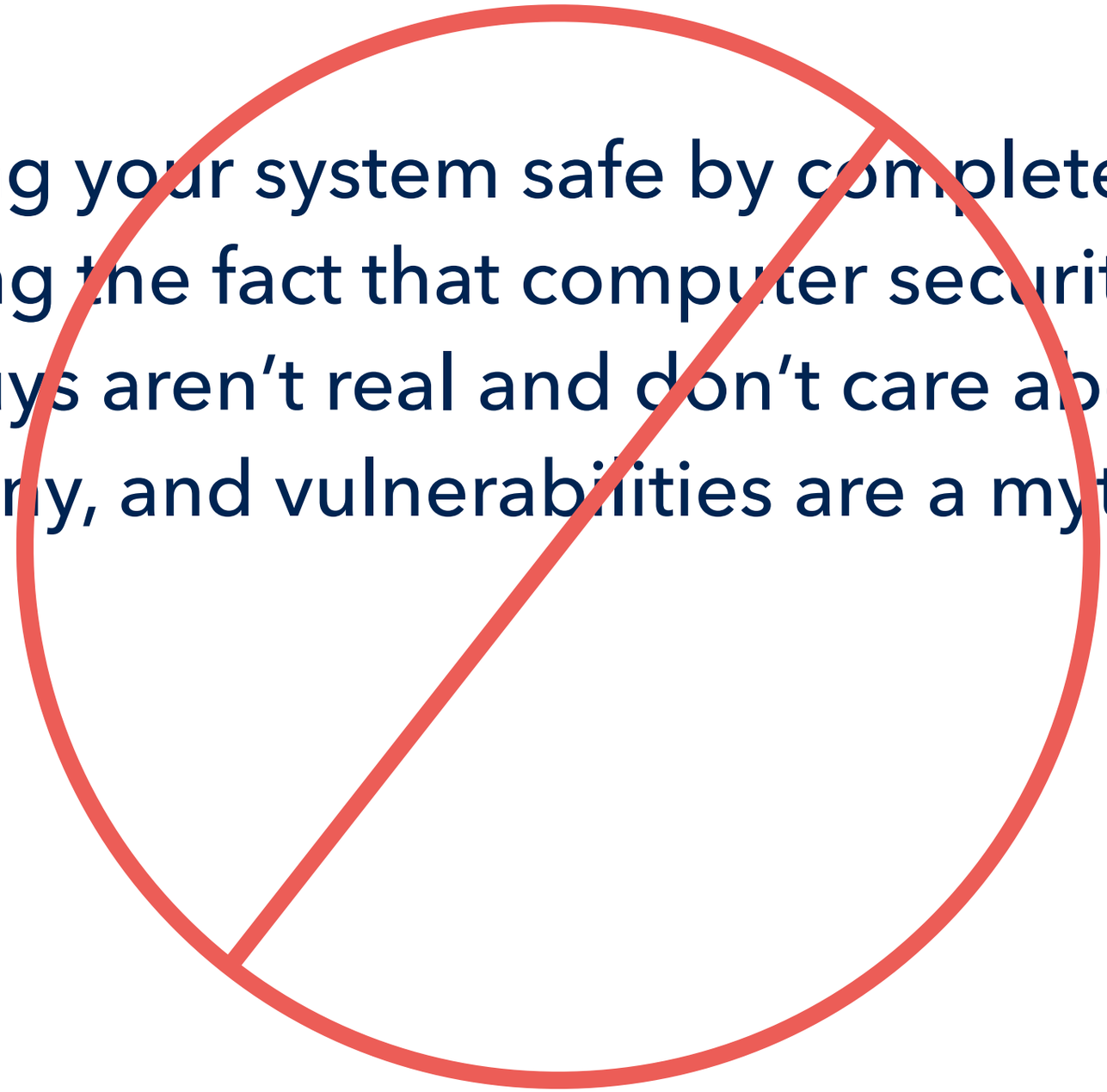
Security through Obscurity

- Keeping your system safe because attackers don’t know where it is, what it does, how it works, why it’s there, who owns it, etc…



Security through Ignorance

- Keeping your system safe by completely ignoring the fact that computer security exists, bad guys aren't real and don't care about your company, and vulnerabilities are a myth



Security by Default / Security by Design

- Keeping your system safe by designing it to be secure from the ground up
- Acknowledging vulnerabilities can occur in code
- Maintaining and prioritizing a security team (even if it’s just one person)
- Thinking about how attackers will try to hurt you

3) Understand your enemy and yourself

Hats? What hats?

- Black hats, white hats, grey hats, etc…
- Used to classify “hackers” by their motivations, purpose, compensation, and **generalized** characteristics

Motivation, who cares?

- It is helpful to understand how different parties are motivated

- Money, power, destruction…

- Morality, responsibility, protection of users

White Hats

NCIS

White Hats

- Security Researchers
- Practice “responsible disclosure”
- Participate in Bug Bounties
- Spread security awareness
- Maintain active Twitter accounts

Black Hats

Hackers, 1995

Black Hats

- Motivations usually include at least one of four of the following:
  - Money - LinkedIn data breach
  - Power - North Korea vs. Sony
  - Destruction - Ukrainian power grid
  - Revenge - “Hacktivists”, Anonymous group

What's a vulnerability?

- When you make an application do something that it's not supposed to do
- To find vulnerabilities, you have to understand the application
- That's **GREAT** for us developers!

Know the application

- What's the intended functionality?
- What's the intended behavior?
- What does the application use as input?
- What does the application produce as output?

Wikipedia vs. CNN

- You find that authenticated, regular users can edit page content
- Vulnerability?
- wikipedia.com
- cnn.com

Attacking is a process

- **Step 1**: Reconnaissance
- **Step 2**: Develop vulnerability hypothesis
- **Step 3**: Test vulnerability hypothesis
- **Step 4**: Develop exploit
- ***Step 5**: Profit/Get Swag/Protect the World

*mileage may vary

Injection Vectors

- Understand **ALL** input to the application!
- Examples for Web Applications:
  - Query parameters
  - URL path
  - PUT/POST parameters
  - Cookies
  - Headers (Referer header)
  - File uploads
  - Functionality from other websites
  - Emails
  - Form fields
  - Web Sockets
  - Browser Local Storage
  - ...

Understand data flow

- How does the input data flow through the program?
  - Example: Data on page X is displayed on page Y and used to calculate the result of page Z
- How does the output of a page flow through the program?
  - Example: the result of a calculation used as part of a tweet

4) The Rundown


@AARON

Legend

Important JS Module Description

https://link-to.software

Command Injection

- Occurs when untrusted data is sent to an interpreter as part of a command or query
- Injection of malicious code into programmer logic, through SQL, NoSQL, eval statements, OS instructions, LDAP, etc...

Command Injection

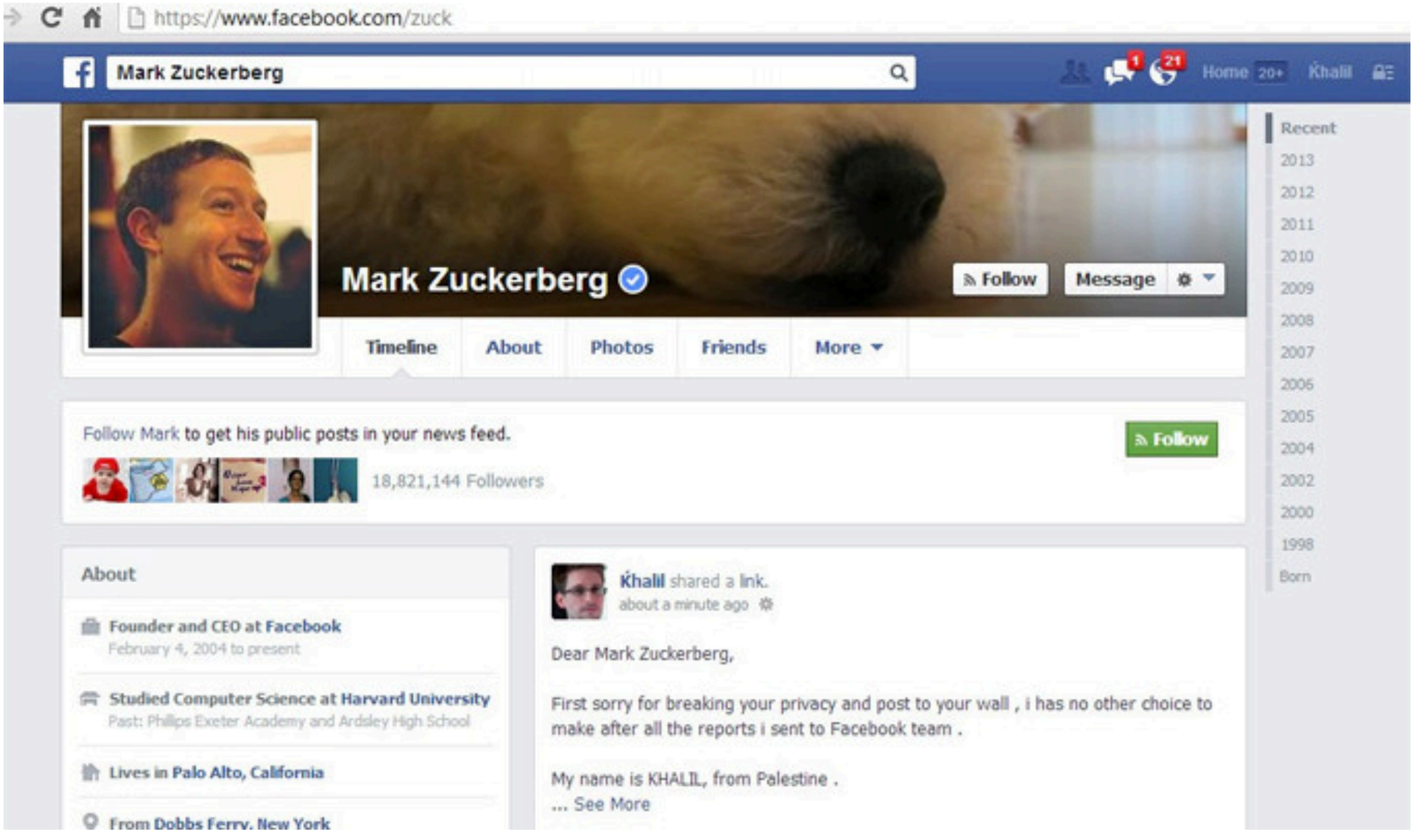
- Validate all user input on the server side
- Do not use eval() or other similar commands without validating user input
  - setTimeout()
  - setInterval()
  - Function()

https://github.com/ctavan/express-validator

Cross-Site Scripting (XSS)

- Run malicious JavaScript in victim's browser and targeted application
- Successful XSS can steal user information (cookies, session IDs, etc.), redirect them to malicious websites, act on users behalf, the list goes on and on...

Cross-Site Scripting (XSS)



Cross-Site Scripting (XSS)

- Validate and sanitize all user input!
- Encode output for specific contexts
- Set appropriate headers
  - Strict-Transport-Security
  - X-Frame-Options
  - X-XSS-Protection
  - X-Content-Type-Options
  - Content-Security-Policy

Cross-Site Scripting (XSS)

Validation

https://github.com/ctavan/express-validator

Setting Appropriate Headers

https://github.com/helmetjs/helmet

Cross-Site Request Forgery (CSRF)

- Forces the end user to execute unwanted actions on a web application in which they're currently authenticated
- Attackers usually target state-changing requests
- Possible to store CSRF attacks in IMG or IFRAME tags in fields that accept HTML

Cross-Site Request Forgery (CSRF)

- Include a random, unpredictable token in requests
- Add tokens to requests which mutate state

Protecting against CSRF

https://github.com/expressjs/csurf

Session Management

- Attacker uses leaks or flaws in the authentication or session management functions to impersonate other users
- Developers often build custom authentication and session management schemes
  - This is very hard to do correctly
  - Often have flaws in implementation

Session Management

- Don’t expose session tokens in URL
- Session tokens should timeout
- Recreate session tokens after successful login
- Use HTTPS for sending tokens
- Use appropriate permissions
- USE OAUTH 2.0 or SAML

Session Management

Building in Permissions

https://github.com/OptimalBits/node_acl

Passport: authenticate to anything (almost)

http://passportjs.org/docs/authenticate

Password Management

- Passwords are often handled incorrectly
- Passwords are left in plaintext or encrypted, not hashed with appropriate algorithm
- Salts on hashes are not used
- Example:
  - 2012 LinkedIn breach
    - Passwords were hashed, but failed to use a salt
    - Easily cracked using existing rainbow tables

Password Management

- Use bcrypt for hashing passwords
- Enforce strong passwords!
- Allow 2-factor authentication
  - Prefer TOTP or U2F
    - Google Authenticator
    - Authy
    - Yubico keys
  - Use SMS if nothing else

Password Management

Hashing Passwords with bcrypt

github.com/ncb000gt/node.bcrypt.js/

2-Factor Authentication

https://github.com/speakeasyjs/speakeasy

(or use Google Authenticator or Authy SDKs)

Handling Cookies

- For cookies, set the following flags:
  - secure: only send them over HTTPS
  - HttpOnly: do not allow the cookie to be accessed via JavaScript (prevent's XSS access of cookies)
- Enforce proper cookie scoping
  - domain: only accessible by certain domain
  - path: only accessible on certain path (and domain)
  - expires: expires after some amount of time

Handling Cookies

Setting Security Properties on Cookies

https://www.npmjs.com/package/cookie-session

https://www.npmjs.com/package/cookies

Signing and Verifying Data (Cookies/URLs)

https://www.npmjs.com/package/keygrip

Strict Mode

- Use strict mode in JavaScript when possible
- Include `use strict` to use a restricted variant of JavaScript, which:
  - Eliminates some silent errors and throws them at all times
  - Helps the JavaScript engine perform optimizations on code
  - Prevents “walking” the JavaScript stack via `arguments.caller` and similar methods

Information Disclosure

- Disclosing errors and excessive logging information can give away information helpful to hackers
- Limit logging and displayed errors in console within production
- Limit information that identifies technology used
  - i.e. Disable some headers like “X-Powered-By: Express”

Sensitive Data Exposure

- Don’t expose sensitive data! It’s a no brainer!
- Users put their trust in you, try not to break it

Sensitive Data Exposure

- Use SSL/TLS (HTTPS)
- Encrypt all sensitive data at rest and in transit
- Don’t store user’s sensitive data unnecessarily
- Disable caching on forms that store sensitive data
  - Often gets stored in local storage, can be accessed by XSS

Free Globally Trusted SSL Certificates!

letsencrypt.org

Some Useful Tools

Scan Node/JS projects for vulnerable modules

https://github.com/retirejs/retire.js/

Audit your applications with the Node Security Platform CLI

https://github.com/nodesecurity/nsp

Some Useful Tools

Isolate your Node codebases from the rest of the system

https://github.com/docker/docker

Lock down dependency versions with NPM Shrinkwrap

https://docs.npmjs.com/cli/shrinkwrap

Takeaways

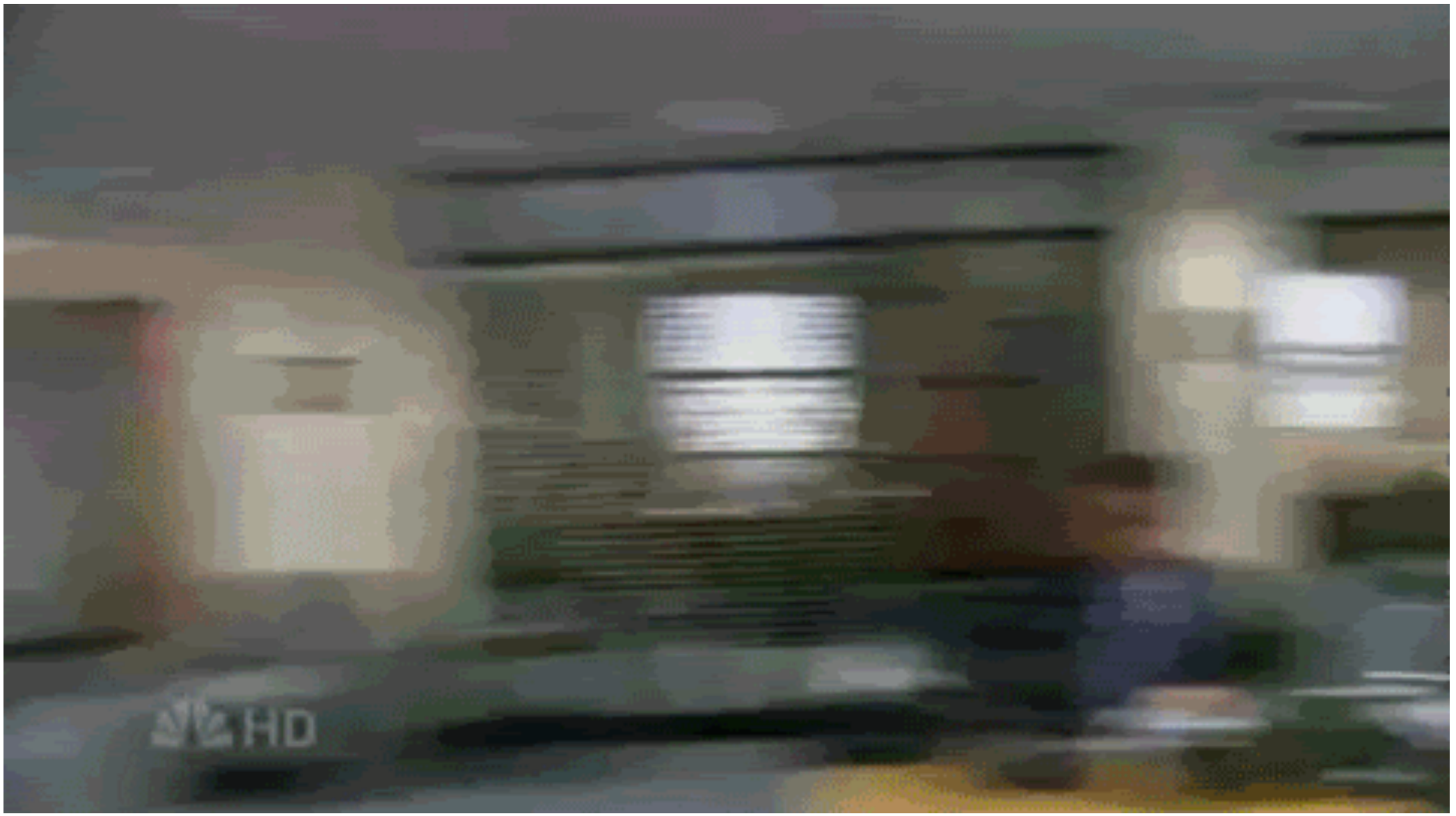
Takeaways

- Acknowledge that security is worth spending money and time on
- Understand application security at a high level
- Have the mindset of an attacker when building and testing your systems
- Use security best practices and well-tested modules to add security properties to your codebases
- Promote a positive security culture at your company

"If you spend more on coffee than on IT security you will be hacked. What's more, you deserve to be hacked."

–RICHARD CLARKE

FORMER NATIONAL COORDINATOR FOR SECURITY, INFRASTRUCTURE PROTECTION AND COUNTER-TERRORISM FOR THE UNITED STATES


HD

Questions? Reach out!

Jared M. Smith

jaredmichaelsmith.com

jared@jaredmichaelsmith.com

jaredmichaelsmith

jaredthecoder

jaredmichaelsmith

Backup Slides

GitHub is Awesome

- https://github.com/sbilly/awesome-security
- https://github.com/Hack-with-Github/Awesome-Hacking
- https://github.com/PaulSec/awesome-sec-talks
- https://github.com/meirwah/awesome-incident-response
- https://github.com/enaqx/awesome-pentest
- https://github.com/jaredmichaelsmith/awesome-vehicle-security

Sites to check out

- OWASP Top 10 Security Vulns - https://www.owasp.org/index.php/Top_10_2013-Top_10
- https://haveibeenpwned.com/ - check if your account has been compromised
- https://www.shodan.io/ - search devices on the internet (computers, connected TV's, cars, IoT devices, etc.)
- https://www.eff.org/https-everywhere - Encrypt all the data!
- https://panopticlick.eff.org/ - See how well your browser stands up to fingerprinting
- https://www.schneier.com/ - Bruce Schneier's blog, prolific cryptographer
- Browse the interwebs privately - https://www.torproject.org/projects/torbrowser.html.en

Notable Podcasts

- http://securityweekly.com/ - Weekly Interview with security expert, security news of the week, and usually a technical segment. Now has a dedicated podcast each week for startup security news and topics and also a podcast for enterprise security news
- http://softwareengineeringdaily.com/ - Interviews with people like Jeff Atwood (creator of Stack Exchange), creators of Spark and Hadoop, creators of ReactJS and Redux, lead engineers at well-known software companies, etc.

Further Reading

- Black Hat Python: Python Programming for Hackers and Pentesters - Amazon
- AngularJS Security: https://www.troyhunt.com/introducing-angularjs-security/
- Troy Hunt's security links: https://www.troyhunt.com/troys-ultimate-list-of-security-links/
- Penetration Testing: A Hands-On Introduction to Hacking - Amazon
- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software - Amazon
- Hacking: The Art of Exploitation - Amazon