Web Apps – Hardening Too Hard

Nick Lauder

Me – nick@quantumsecurity.co.nz

What I do

- Penetration Tester
- Security Consultant for Quantum

Interests

- Wireless stuff
- Embedded dev
- Electronics

Thank You to Our Sponsors and Hosts!

OWASP
NEW
ZEALAND
owasp.org.nz

DATACOM

QUANTUM
SECURITY

INSOMNIA
SECURITY SPECIALISTS :: REST SECURED

myob

Pulse Security
www.pulsesecurity.co.nz

VOCUS

security initiative

SEQA™
Information Security

Without them, this Conference couldn't happen

Why are we here?

- Talking about general web application hardening
- Covering a wide range of topics
- Aimed at developers and security enthusiasts

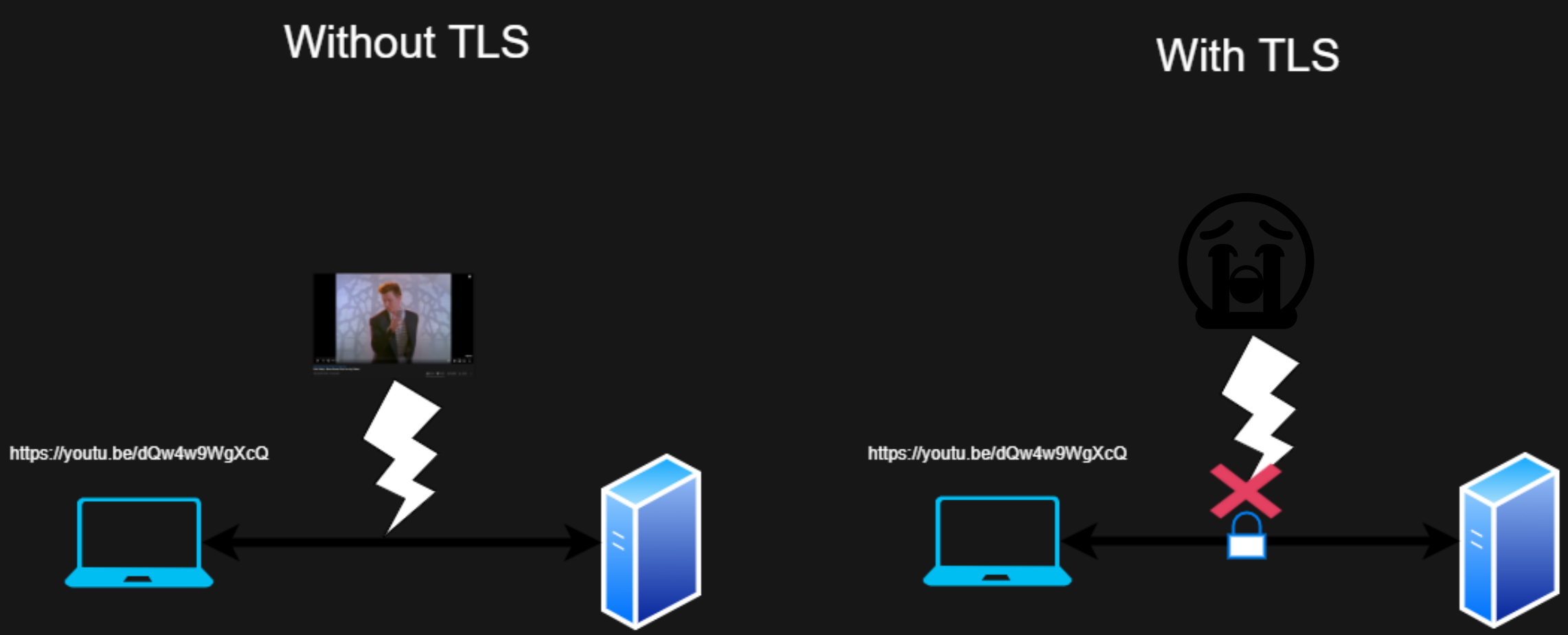
Why should you care?

- We find these issues in *almost* every web application
- Make pentesters spend time finding the issues that have real impacts
- Helps to improve your security hygiene across your applications
- All this is learnt from pentesting and standing up my own web applications

Overview

- Issues
  - Weak TLS Configuration
  - Weak HTTP Security Header Configuration
  - Weak Cookie Configuration
  - Version Number Disclosure
  - Lack of CSRF Tokens
  - Sequential Object IDs
  - (More if we have time)
- Summary

What is TLS/SSL/HTTPS?



What makes up TLS?

Protocols

- SSLv2
- SSLv3
- TLSv1.0
- TLSv1.1
- TLSv1.2
- TLSv1.3

Ciphers

- Key Exchange
  - RSA
  - DHE
  - ECDHE
- Authentication
  - RSA
  - ECDSA
  - DSS
- Encryption
  - AES GCM/CCM/CBC
  - CHACHA20_POLY1305
- Hashing
  - SHA 1/2/3
  - MD5

What issues do we see?

Protocols

- TLSv1.0
  - POODLE
  - BEAST
- TLSv1.1
  - Not inherently insecure, but improvements have been made

Ciphers

- RSA
  - No forward secrecy
- AES CBC
  - Padding Oracle
  - POODLE
- SHA1
  - SHAppening
  - SHAttered

How to stop the hacker?

Fixing common TLS issues

Loads of free tools to help!

- Best Practice Guides
  - OWASP - https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/TLS_Cipher_String_Cheat_Sheet.md
  - IETF - https://tools.ietf.org/html/draft-ietf-uta-rfc7525bis-00
  - Qualys - https://www.ssllabs.com/projects/best-practices/
- Configuration
  - https://ssl-config.mozilla.org/
- Scanners
  - https://www.ssllabs.com/ssltest/index.html
  - https://testtls.com/

Fixing common TLS issues

My Recommendations

- Protocols
  - TLSv1.3
  - TLSv1.2
- Ciphers
  - ECDHE for key exchange
  - ECDSA for authentication
  - AES256 GCM
  - SHA384

- TLSv1.3
  - TLS_AES_256_GCM_SHA384
- TLSv1.2
  - TLS_ECDHE_ECDSA_AES_256_GCM_SHA384

Issues you may encounter

- Support for legacy clients?
  - Support for TLSv1.0 and TLSv1.1 is already gone!

- ECDSA for authentication?
  - Requires an ECDSA certificate authority (Not uncommon, but may not be default)

HTTP Security Headers

They activate what is already available!

- Modern
  - Content Security Policy (CSP)
  - Strict Transport Security (HSTS)
  - Cross Origin Resource Sharing (CORS)
- Aging
  - X-Content-Type-Options
  - X-Frame-Options
  - X-XSS-Protection
  - Referrer-Policy

Easy to fix

- HSTS
  - *Strict-Transport-Security: max-age=31536000; includeSubDomains*
- X-Content-Type-Options
  - *X-Content-Type-Options: nosniff*
- X-Frame-Options
  - *X-Frame-Options: DENY*
- X-XSS-Protection
  - *X-XSS-Protection: 1; mode=block*
- Referrer-Policy
  - *Referrer-Policy: no-referrer*

Some great resources

- Documentation
- Scanning tool
  - https://observatory.mozilla.org/ – Scans HTTP headers and provides rating

Context dependent headers

- CORS - What is it?
  - Used to prevent third-party applications retrieving content from your site

Common issues:

- *Access-Control-Allow-Origin: **
  - hmm😠
  - Needed on public APIs
- How to do properly:
  - *Access-Control-Allow-Origin: https://yourwebsite.link*
- Few more CORS headers but they are more specific to your use case

The hard one…

- Content Security Policy (CSP) – What does it do?
  - Bit of everything really – Built up of directives to activate browser protections
  - Obsoletes a lot of the previously mentioned headers
- Directives
  - default-src – sets a default values for all directives
  - frame-ancestors – chooses where the page can be loaded in a frame
  - form-action – chooses where forms can submit to
  - base-uri – specifies valid values for the *base* element
  - script-src – chooses where to load JavaScript from
  - object-src – chooses where to load objects from (object, embed, applet tags)
  - upgrade-insecure-requests – redirects http to https

What attacks can CSP prevent

- Cross-Site Scripting (XSS)
  - script-src – replaces X-XSS-Protection
  - object-src
  - worker-src
  - base-uri
- Clickjacking
  - frame-ancestors – replaces X-Frame-Options
  - child-src
  - frame-src
- Formjacking
  - form-action
  - base-uri

What can go wrong?

Functionality

- script-src
  - ‘example.com’
    - Prevents all inline code and any resources
  - ‘none’
    - Prevents loading of all JS
  - ‘nonce-<base64-value>’
    - Requires additional web app functionality
  - ‘<hash>’
    - Requires changing policy with every JS file change

Security

- script-src
  - ‘unsafe-inline’
    - Allows all inline code, even without hashes/nonce
  - ‘example.com’
    - Allows all content from example.com… May not be safe
  - ‘self’
    - Allows for self hosted files, but may be attacker uploaded?

How to do it right

Use the free resources!

- Header Evaluators
  - https://observatory.mozilla.org/ – header scanner with scores
  - https://github.com/GoSecure/csp-auditor – OWASP Zap/Burp Suite CSP plugin
  - https://cspscanner.com/ – In-depth CSP evaluator
- Configurators
  - https://report-uri.com/home/generate - Graphical, step-by-step CSP generator
- Best Practice Guides
  - https://owasp.org/www-project-secure-headers/ – OWASP on headers
  - https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html – OWASP on CSP
  - https://infosec.mozilla.org/guidelines/web_security - General web security

Cookies

What is a cookie 

- They track where you are (but sometimes in a good way)
- Set by the web server
- Often used to store authentication tokens

Baking a cookie

- *Set-Cookie* HTTP response header
- Flags
  - *Secure* – Only transmitted over HTTPS
  - *HTTPOnly* – Not accessible via JavaScript
  - *SameSite=None|Lax|Strict* – Prevents inclusion if request originates from a separate page

- *Set-Cookie: token=V2h5IGFyZSB5b3UgbGlrZSB0aGlzPw==; Secure; HTTPOnly; SameSite=Strict*

Ingredients

- *Secure*
  - Prevents against man-in-the-middle (MITM) attacks
- *HTTPOnly*
  - Prevents access to auth token during XSS attacks
- *SameSite*
  - *None – Send cookie with every request to the owning domain*
  - *Lax – Sends cookie only when redirecting to owning domain*
  - *Strict – Sends cookie only when originating from owning domain*
- Max-Age
  - Sets the lifetime of the cookie
- Domain
  - Sets the domain that the cookie is valid path
- Path
  - Sets the URL path that the cookie is valid for

Common issues

- Not setting the flags
  - Uncommon to see the flags breaking the application
- WAFs, Proxies, load balancers add their own cookies
- Generating cookies insecurely (Session Management issue)
  - “HackTheBox Special”

Set-Cookie: token=eyJ1c2VybmFtZSI6InVzZXIiLCJyb2xlIjoiYWRtaW4ifQ==; Secure; HTTPOnly; SameSite=Strict;

{"username":"user", "role":"user"}

Cooking tips

- Use all the flags
  - If this breaks something, review what broke rather then removing the control
- Generate cookies securely (random)
- Set restrictive scope and lifetimes
  - Short lifetimes for only the specific domains/paths you require

Version Number Disclosure

This includes:

- Web servers
- JavaScript libraries
- Web Application Firewalls (WAFs)
- PDF Generators

So what?

Well done, you discovered that we use a web server to host a website 👏

Increases your exposure ☹

- As an targeted attacker/pentester?
  - Big arrow saying exploit here
  - CVEs provide a nice list of potential vulnerabilities
  - Saves me a whole lotta time
- As a script kiddie?
  - Automated tools may identify your site as vulnerable (Shodan)
  - Attracts attention that may have passed by
  - Advertising your vulnerabilities when new exploits released?

How to fix?

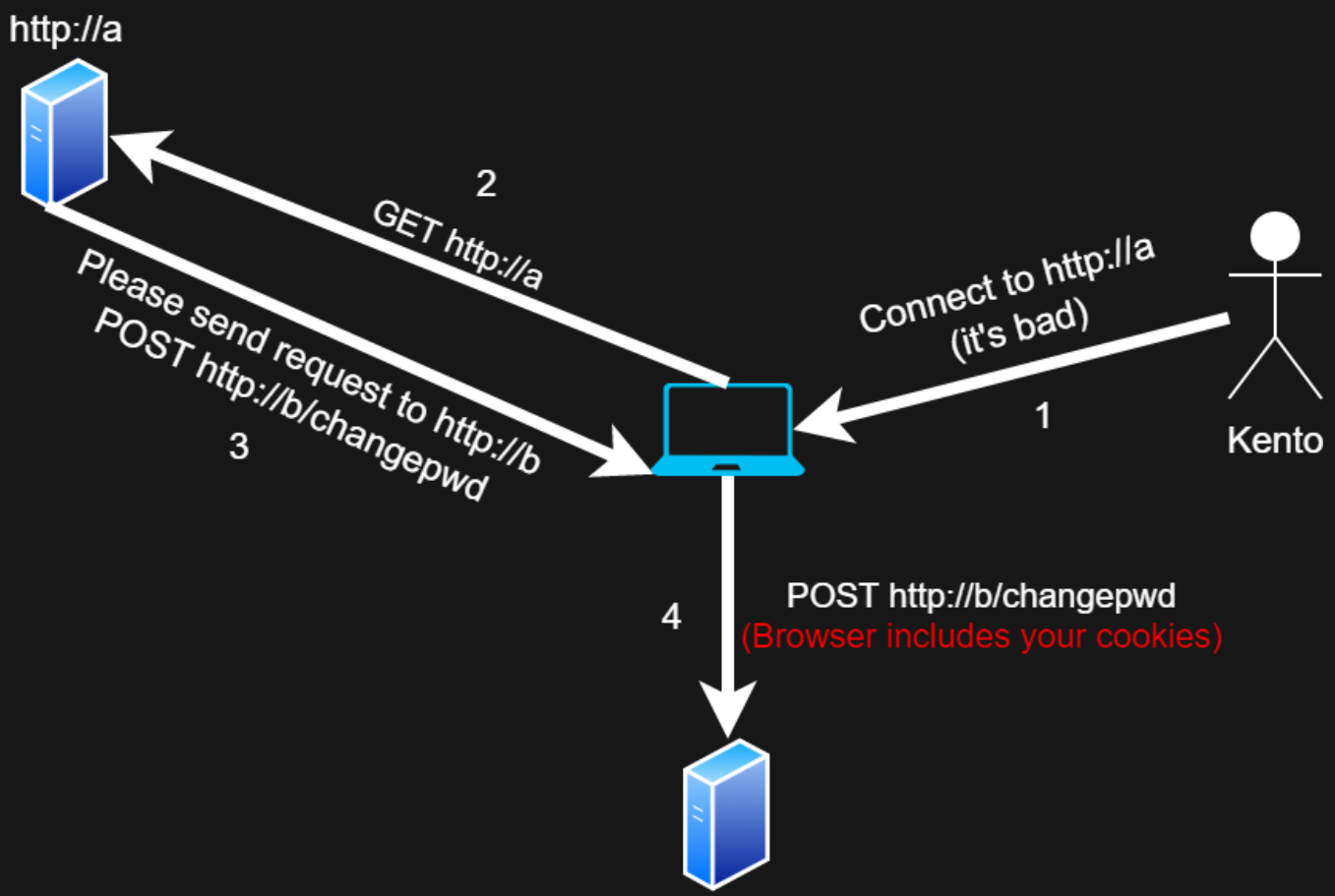
- Web servers
  - Stop returning the *Server* header
  - Stop returning stack traces as error messages
  - Stop using default error messages
- JavaScript libraries
  - Minify/Compress production files
  - Remove comments
- WAFs
  - Stop using headers to advertise the WAF product
- PDF Generators
  - Configure so that they don’t include information in the metadata

How to fix?

- Web servers
  - Stop returning the *Server* header
  - Stop returning stack traces as error messages
  - Stop using default error messages
- JavaScript libraries
  - Minify/Compress production files
  - Remove comments
- WAFs
  - Stop using headers to advertise the WAF product
- PDF Generators
  - Configure so that they don't include information in the metadata

Side Note:
Update your underlying software…
9/10 times we find that disclosed software is out of date

Cross-Site Request Forgery



Cross-Site Request Forgery

What's the issue?

- When requesting a domain, cookies are automatically included
- Request becomes authenticated
- Makes the outcome of the request the same as if you did it

- Last example showed a change password request, resulting in the malicious website changing Kento's password

But we've already fixed 😠

Cookies with *SameSite* set well help to prevent this!

- Applies as a blanket across entire site
  - May break functionality
  - Have to manage trusted vs untrusted resources

What is dis?

CSRF/XSRF Token

- Randomly generated nonce
- Included with every page load
- Sent with every submission (POST)
- Token verified server side against what was provided
- Supported by lots of frameworks

How to implement

- Existing Solutions
  - Java – OWASP CSRF Guard or Spring Security
  - PHP – CSRFProtector Project
  - AngularJS – XSRF Protection
- Manually
  - Generate nonce server side and store along side session token
  - Send token in hidden HTML form field
  - On form submission, compare provided token with stored value

Resources

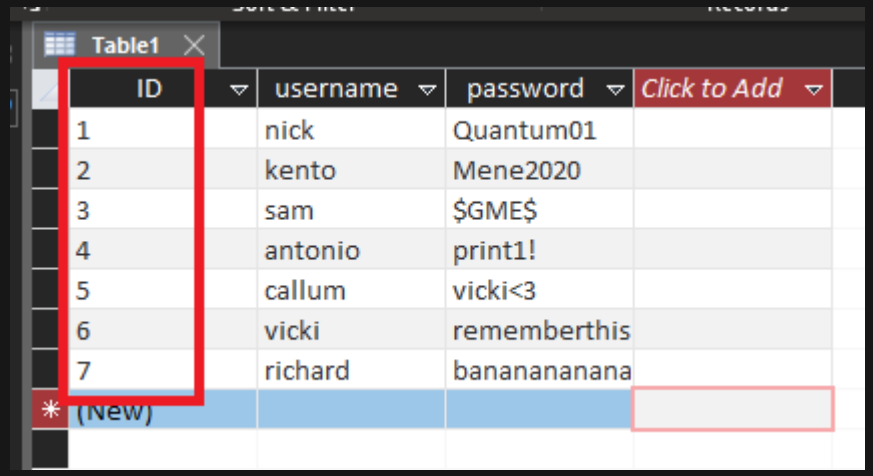
- OWASP Cheat Sheet!
  - https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- PortSwigger
  - https://portswigger.net/web-security/csrf/tokens
- Your framework documentation!

Sequential Object IDs

Unique identifiers used to call specific objects

- Users
- Posts
- Pictures
- Uploads
- Groups

What causes?



http://myapi/api/v1/users/1/profile

Problem?

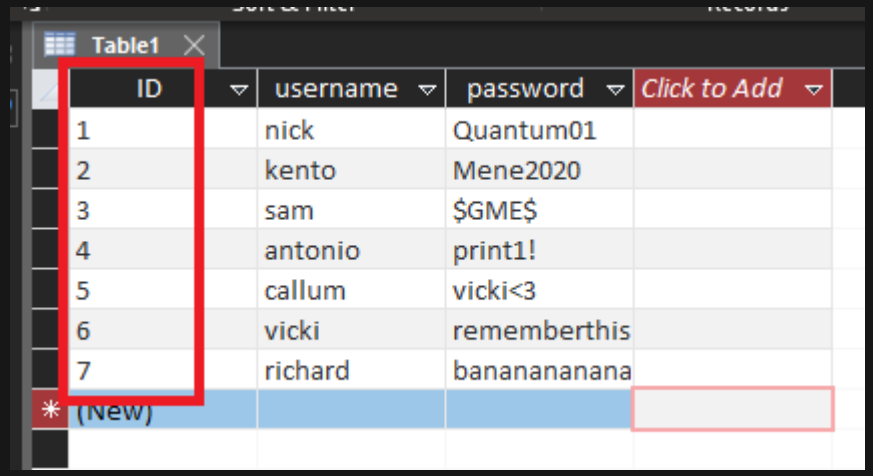


Table1

ID	username	password	Click to Add
1	nick	Quantum01	
2	kento	Mene2020	
3	sam	GME	
4	antonio	print1!	
5	callum	vicki<3	
6	vicki	rememberthis	
7	richard	bananananana	
(New)			

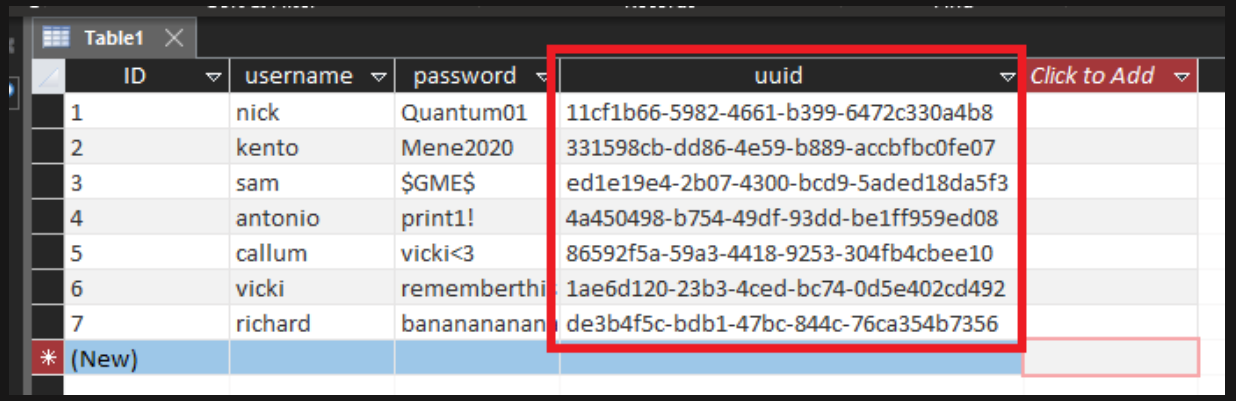
http://myapi/api/v1/users/1/profile

If my profile is at the link:
http://myapi/api/v1/users/1/profile

Then maybe I can access:
http://myapi/api/v1/users/2/profile

This allows for broken access control to be exploited

But I like my numbers ☹



http://myapi/api/v1/users/11cf1b66-5982-4661-b399-6472c330a4b8/profile

Sorry about your API

Speedrun

- Error Handling
  - Use custom messages – no stack traces or default pages
- Password policies
  - 14 min limit with no max limit and at least one non-letter plz
- User enumeration
  - Return generic messages on forgot password page
- Other services
  - Stop running SSH on your web server
- XSS
  - Validate and sanitise user input everywhere
- No MFA
  - MFA isn’t hard anymore, at least do it for admins
- No brute-force protections
  - Account lockout
  - Rate limiting
  - CAPTCHA

Summary

- So many free scanning tools
- Scan and fix your own stuff before a pentest
- Resources and documentation are everywhere
- Pentesters want to find the big issues
- OWASP has an article on everything

Actual takeaways

- Make the pentesters work hard
  - More value in 2 highs than 10 lows
- Genuinely may not be able to fix them all
  - But knowing your issues is great proof of knowledge
- Really satisfying when your webapp is done right
  - Big grin when my internal apps were pentested

Thanks for coming

Come and talk to me about

- Working at Quantum
- Issues I’ve encountered with fixing these issues
- Unique ways to rick roll someone

Questions?