Designing, Building & Using APIs

(and a little bit about security for fun…)

Sensible Abstractions

```
void P1_0_high()
{
  P1DIR |= BIT1;
  P1OUT |= BIT1;
}

Void P1_1_high()
{ ... }
```

```
void P1_0_low()
{
  P1DIR |= BIT1;
  P1OUT |= BIT0;
}

Void P1_1_low()
{ ... }
```

What’s wrong with this?

- Very, very, very, very, very, very, very repetitive…
- Lots of scope to make mistakes and off-by-1 errors (P1.0 = BIT1 etc…)
- Leaky abstraction? Should we change a direction register?
- When you write complex code, think what else you might ~~change~~break
- More to the point, it’s a MESS
- Hard to maintain this code, and it will sprawl. Risk of rapidly exceeding the code size limit on an MSP430 with just your GPIO logic
- Do you fancy doing this on an FRAM SMD chip with 4 or 6 or more GPIO banks?

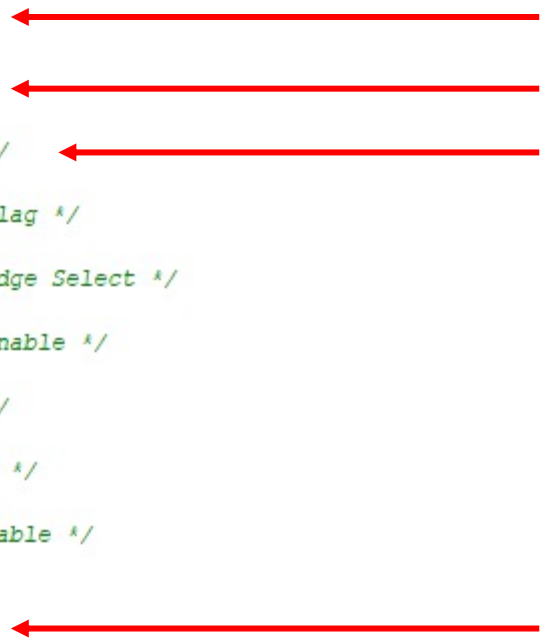
Let’s try again!

```
#define BASE_PORT_REG = 0x20 // this is the lowest port config register

// port is integer from 0 to 4
// pin is integer from 0 to 7
void gpio_out(uint8_t port, uint8_t pin, uint8_t state)
{
  // a bit mask for the selected pin
  uint8_t pinMask = 0x01 << pin;
  // Get address of PxDIR for port number `port`:
  uint8_t* portDirReg = BASE_PORT_REG + 2 + ((port - 1) * 0x08);
  uint8_t* portOutReg = BASE_PORT_REG + 1 + ((port - 1) * 0x08);
  portDirReg |= pinMask;
  if (state != 0) // using un-gated if’s to save space, don’t do this in your code!!!
    portOutReg |= pinMask;
  if (state == 0)
    portOutReg &= ~pinMask;
}
```

```
#define P1IN_               (0x0020u)  /* Port 1 Input */
READ_ONLY DEFC( P1IN          , P1IN_)
#define P1OUT_              (0x0021u)  /* Port 1 Output */
DEFC(   P1OUT             , P1OUT_)
#define P1DIR_              (0x0022u)  /* Port 1 Direction */
DEFC(   P1DIR             , P1DIR_)
#define P1IFG_              (0x0023u)  /* Port 1 Interrupt Flag */
DEFC(   P1IFG             , P1IFG_)
#define P1IES_              (0x0024u)  /* Port 1 Interrupt Edge Select */
DEFC(   P1IES             , P1IES_)
#define P1IE_               (0x0025u)  /* Port 1 Interrupt Enable */
DEFC(   P1IE              , P1IE_)
#define P1SEL_              (0x0026u)  /* Port 1 Selection */
DEFC(   P1SEL             , P1SEL_)
#define P1SEL2_             (0x0041u)  /* Port 1 Selection 2 */
DEFC(   P1SEL2            , P1SEL2_)
#define P1REN_              (0x0027u)  /* Port 1 Resistor Enable */
DEFC(   P1REN             , P1REN_)

#define P2IN_               (0x0028u)  /* Port 2 Input */
READ_ONLY DEFC( P2IN          , P2IN_)
#define P2OUT_              (0x0029u)  /* Port 2 Output */
DEFC(   P2OUT             , P2OUT_)
#define P2DIR_              (0x002Au)  /* Port 2 Direction */
DEFC(   P2DIR             , P2DIR_)
#define P2IFG_              (0x002Bu)  /* Port 2 Interrupt Flag */
DEFC(   P2IFG             , P2IFG_)
#define P2IES_              (0x002Cu)  /* Port 2 Interrupt Edge Select */
DEFC(   P2IES             , P2IES_)
#define P2IE_               (0x002Du)  /* Port 2 Interrupt Enable */
DEFC(   P2IE              , P2IE_)
#define P2SEL_              (0x002Eu)  /* Port 2 Selection */
DEFC(   P2SEL             , P2SEL_)
#define P2SEL2_             (0x0042u)  /* Port 2 Selection 2 */
DEFC(   P2SEL2            , P2SEL2_)
#define P2REN_              (0x002Fu)  /* Port 2 Resistor Enable */
DEFC(   P2REN             , P2REN_)
```



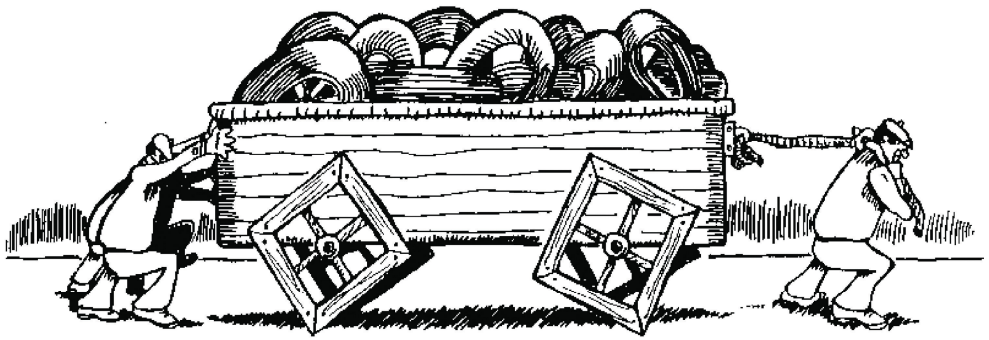
BASE_PORT_REG for me

BASE_PORT_REG + 1 = PxOUT

BASE_PORT_REG + 2 = PxDIR

BASE_PORT_REG +8 for P2…

But don’t reinvent the wheel



MSP430 driverlib basically does this sort of thing for you!

(you’ll struggle to use it on the G2553 though due to code space...)

But this shows you how an API works and what to think about!

Key Points of an API

1. Make it generic

Don’t make different APIs for P1 and P2 – understand the generic principle, and make an API that lets you access both!

Create meaningful, elegant abstractions. Think how much you require the user of your API to understand the hardware…

2. One API = One Action

Create meaningful abstractions!

Turning on GPIO P1.2 should not turn off any other unrelated GPIOs.

Turning on GPIO P1.2 should ensure it’s treated as an output

3. Deliver Consumable Functionality

Think about the functionality that is required, and deliver this in “easily consumable” chunks. Your API should be designed around the use-case, and designed to give you the access needed to implement the logic.

Sometimes a C API is not enough…

Functions in C are a type of API. Callable only within your code, or other code that links against yours…

You could pull in code from another library and use it (include the header file, and the C file)

What if you want to do something outside of your own board?

When you might need more than a raw C API

- If you need to integrate with code that is running elsewhere
- When the code using your functionality won't be running on the same device (i.e. microcontroller)
- Where you need to communicate between 2 separate processors
- Where you need to communicate/call functionality from a different location from where you and the microcontroller are located

Introducing HTTP

The protocol you use every day, and see every day, but maybe haven't thought of as an API in this way

(There are other ways to do this for special use-cases like 2 CPUs on the same PCB – like direct memory access (DMA), but this has security issues you want/need to understand first… Usually if you have 2 CPUs, there's a security reason behind it!)

About HTTP?

- A text-based protocol (basically serial text sent to a TCP socket on port 80 on a web server)
- You can make a (very basic) HTTP request yourself from telnet:

```
> telnet www.google.com 80
Trying 142.250.187.228...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.com

HTTP/1.1 200 OK
Date: Mon, 29 Nov 2021 00:08:07 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=511=UQmt936cnWlV8A3OVm6aU_XWI1dD-RmY9WCJ19CF80V1bBm_vqKIoYgc0olf_mIyRoBQ3iDSGYCDWZupW41sSZEE2HH5Yblmhu4LCVKju1VzStvblrQL
6JyY8ZrkM6myyuVm3TDUHdib4er1KSoYZBq7TYRwpaqsy0U1-nb08vA; expires=Tue, 31-May-2022 00:08:07 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked
```

Bullying inquiry head quits as PM backs Patel

Boris Johnson rejects standards adviser's findings on the behaviour of home secretary towards staff.

22m | UK Politics

- **Summary of Patel bullying report**
- **Profile: Priti Patel**

LIVE
'Substantial differences' in Covid rates across England

8m Spain plans to vaccinate large part of population by mid-2021

23m Ugandan politician charged with breaking virus rules

34m Pfizer applies for US emergency authorisation

Flu jab push as Covid vaccine roll-out planned
4m | UK

Theresa May killing-threat man jailed
1h | Berkshire

McCann suspect's rape conviction appeal rejected
4h | Europe

Hip-hop star Rodney P admits ex-girlfriend attack
2h | UK

Christmas restriction warning if Wales' cases rise
1h | Wales

Biden wins Georgia recount as Trump setbacks mount
1h | US Election 2020

Probe after Pope's Insta account 'likes' model
3h | Europe

Infection rates levelling off in England and Scotland

ONS data suggests rates in school-age children are still rising, but falling among young adults.

5m | Health | 453

Sunak mulls public sector pay freeze for millions

Chancellor Rishi Sunak is trying to bolster public finances after huge spending to fight coronavirus.

2h | Business

Killer drug GHB 'should be reclassified'

GHB has been used in a series of murders but is currently in the same category as anabolic steroids.

3h | UK

Coroner says Kyle 'may have contributed to' death

Steve Dymond died days after taking a lie-detector test on the Jeremy Kyle Show.

6m | Hampshire & Isle of Wight

https://www.bbc.co.uk/news

Search

Let us know you agree to cookies

We use cookies to give you the best online experience.
Please let us know if you agree to all of these cookies.

Yes, I agree

No, take me to settings

BBC | Sign in | Home | News | Sport | Weather | iPlayer | Sounds | CBBC | More | Search

NEWS

Home | Coronavirus | US Election | UK | World | Business | Politics | Tech | Science | Health | Family & Education | Entertainment & Arts | Stories | In Pictures | More

England | N. Ireland | Scotland | Alba | Wales | Cymru | Local News

Bullying inquiry head quits as PM backs Patel

Boris Johnson rejects standards adviser's findings on the behaviour of home secretary towards staff.

23m | UK Politics

- **Summary of Patel bullying report**
- **Profile: Priti Patel**

LIVE

'Substantial differences' in Covid rates across England

9m Spain plans to vaccinate large part of population by mid-2021

24m Ugandan politician charged with breaking virus rules

35m Pfizer applies for US emergency authorisation

Flu jab push as Covid vaccine roll-out planned

4m | UK

Theresa May killing-threat man jailed

1h | Berkshire

McCann suspect's rape conviction appeal rejected

4h | Europe

Hip-hop star Rodney P admits ex-girlfriend attack

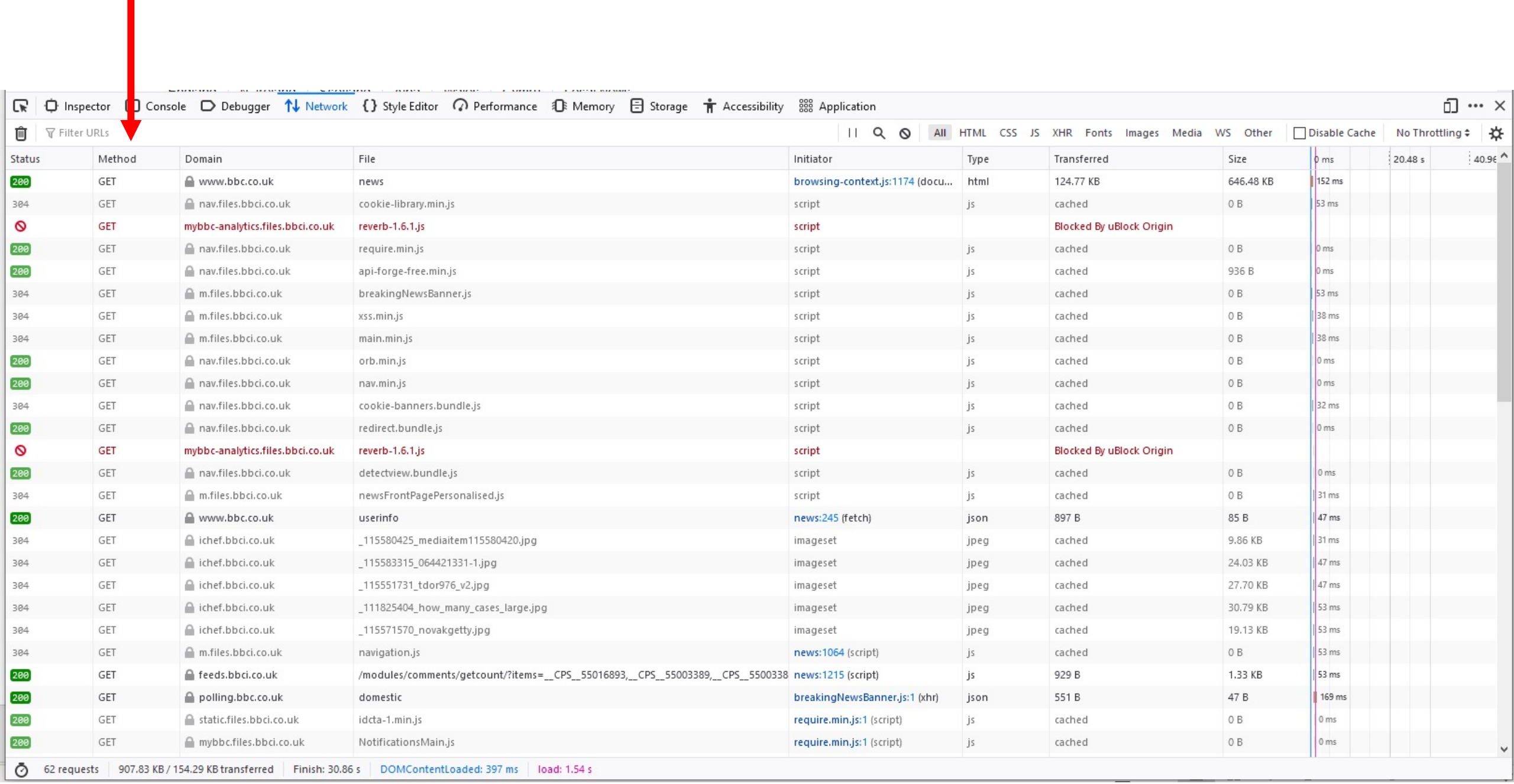
2h | UK

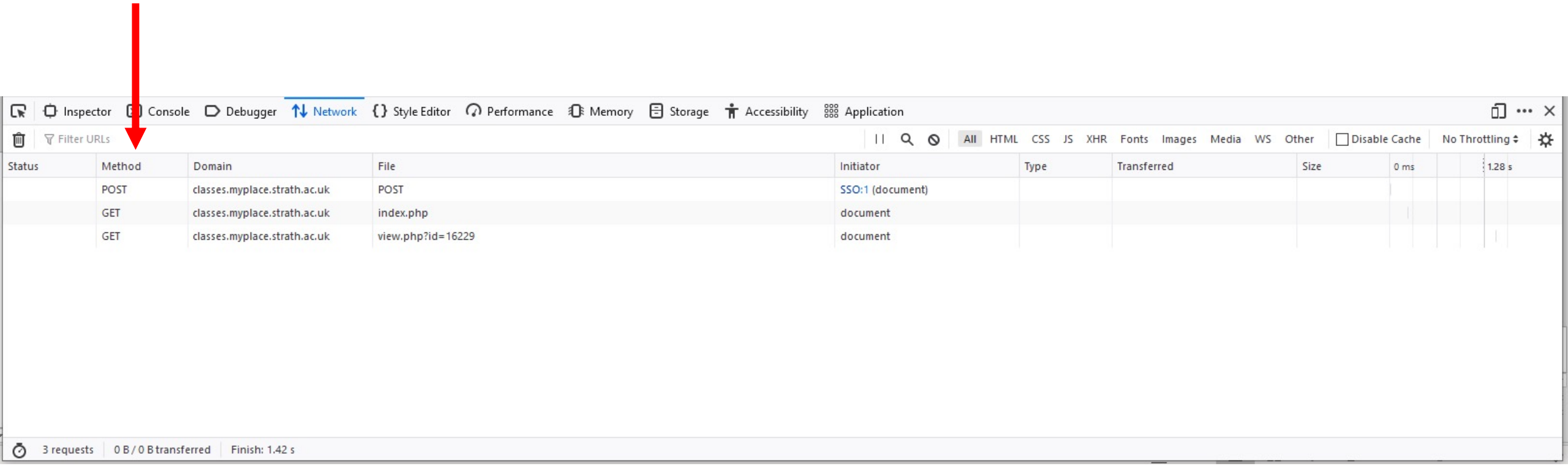
Inspector | Console | Debugger | Network | Style Editor | Performance | Memory | Storage | Accessibility | Application

Filter URLs | All | HTML | CSS | JS | XHR | Fonts | Images | Media | WS | Other | Disable Cache | No Throttling

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	1.28 s	2.5
200	GET	www.bbc.co.uk	news	browsing-context.js:1174 (docu...	html	124.77 KB	646.48 KB	152 ms		
304	GET	nav.files.bbci.co.uk	cookie-library.min.js	script	js	cached	0 B	53 ms		
	GET	mybbc-analytics.files.bbci.co.uk	reverb-1.6.1.js	script		Blocked By uBlock Origin				
200	GET	nav.files.bbci.co.uk	require.min.js	script	js	cached	0 B	0 ms		
200	GET	nav.files.bbci.co.uk	api-forge-free.min.js	script	js	cached	936 B	0 ms		
304	GET	m.files.bbci.co.uk	breakingNewsBanner.js	script	js	cached	0 B	53 ms		
304	GET	m.files.bbci.co.uk	xss.min.js	script	js	cached	0 B	38 ms		
304	GET	m.files.bbci.co.uk	main.min.js	script	js	cached	0 B	38 ms		
200	GET	nav.files.bbci.co.uk	orb.min.js	script	js	cached	0 B	0 ms		
200	GET	nav.files.bbci.co.uk	nav.min.js	script	js	cached	0 B	0 ms		
304	GET	nav.files.bbci.co.uk	cookie-banners.bundle.js	script	js	cached	0 B	32 ms		

61 requests | 907.79 KB / 154.29 KB transferred | Finish: 1.81 s | DOMContentLoaded: 397 ms | load: 1.54 s


Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter URLs
All
HTML
CSS
JS
XHR
Fonts
Images
Media
WS
Other
Disable Cache
No Throttling
Status Method Domain File Initiator Type Transferred Size
200 GET www.bbc.co.uk news browsing-context.js:1174 (docu... html 124.77 KB 646.48 KB
304 GET nav.files.bbci.co.uk cookie-library.min.js script js cached 0 B
GET mybbc-analytics.files.bbci.co.uk reverb-1.6.1.js script Blocked By uBlock Origin
200 GET nav.files.bbci.co.uk require.min.js script js cached 0 B
200 GET nav.files.bbci.co.uk api-forge-free.min.js script js cached 936 B
304 GET m.files.bbci.co.uk breakingNewsBanner.js script js cached 0 B
304 GET m.files.bbci.co.uk xss.min.js script js cached 0 B
304 GET m.files.bbci.co.uk main.min.js script js cached 0 B
200 GET nav.files.bbci.co.uk orb.min.js script js cached 0 B
200 GET nav.files.bbci.co.uk nav.min.js script js cached 0 B
304 GET nav.files.bbci.co.uk cookie-banners.bundle.js script js cached 0 B
200 GET nav.files.bbci.co.uk redirect.bundle.js script js cached 0 B
GET mybbc-analytics.files.bbci.co.uk reverb-1.6.1.js script Blocked By uBlock Origin
200 GET nav.files.bbci.co.uk detectview.bundle.js script js cached 0 B
304 GET m.files.bbci.co.uk newsFrontPagePersonalised.js script js cached 0 B
200 GET www.bbc.co.uk userinfo news:245 (fetch) json 897 B 85 B
304 GET ichef.bbci.co.uk _115580425_mediaitem115580420.jpg imageset jpeg cached 9.86 KB
304 GET ichef.bbci.co.uk _115583315_064421331-1.jpg imageset jpeg cached 24.03 KB
304 GET ichef.bbci.co.uk _115551731_tdor976_v2.jpg imageset jpeg cached 27.70 KB
304 GET ichef.bbci.co.uk _111825404_how_many_cases_large.jpg imageset jpeg cached 30.79 KB
304 GET ichef.bbci.co.uk _115571570_novakgetty.jpg imageset jpeg cached 19.13 KB
304 GET m.files.bbci.co.uk navigation.js news:1064 (script) js cached 0 B
200 GET feeds.bbci.co.uk /modules/comments/getcount/?items=__CPS__55016893,__CPS__55003389,__CPS__5500338 news:1215 (script) js 929 B 1.33 KB
200 GET polling.bbc.co.uk domestic breakingNewsBanner.js:1 (xhr) json 551 B 47 B
200 GET static.files.bbci.co.uk idcta-1.min.js require.min.js:1 (script) js cached 0 B
200 GET mybbc.files.bbci.co.uk NotificationsMain.js require.min.js:1 (script) js cached 0 B
0 ms
20.48 s
40.96
152 ms
53 ms
0 ms
0 ms
53 ms
38 ms
38 ms
0 ms
0 ms
32 ms
0 ms
0 ms
31 ms
47 ms
31 ms
47 ms
47 ms
53 ms
53 ms
53 ms
53 ms
169 ms
0 ms
0 ms
62 requests
907.83 KB / 154.29 KB transferred
Finish: 30.86 s
DOMContentLoaded: 397 ms
load: 1.54 s


Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter URLs
All
HTML
CSS
JS
XHR
Fonts
Images
Media
WS
Other
Disable Cache
No Throttling
Status
Method
Domain
File
Initiator
Type
Transferred
Size
0 ms
1.28 s
POST
classes.myplace.strath.ac.uk
POST
SSO:1 (document)
GET
classes.myplace.strath.ac.uk
index.php
document
GET
classes.myplace.strath.ac.uk
view.php?id=16229
document
3 requests
0 B / 0 B transferred
Finish: 1.42 s

HTTP Verbs

GET, POST (and many, many more)

Happen behind the scenes of every request

When you visit a website, that’s a GET request

HTTP Verbs

The actual verb you use doesn't matter, but there are some conventions

GET = retrieve data (reading only)

HEAD = just fetch the headers for a page (not the body)

DELETE = delete the resource referred to by the URL visited

POST = submit new data (you will know what you're making but not the URL yet)

PUT = submit data to a URL (either new, or updating existing)

PATCH = partially update an existing value (i.e. certain parameters)

Key Points

- The logic behind verbs is not implemented by default – when you build a web service, you get told the URL and the verb…
- You can implement logic that does anything – you CAN make a GET request trigger deletion of something in your application i.e.

  GET /light_switches/id/12345/delete

- But just don’t!!! It’s bad practice for many reasons… Not least because you might confuse others!

DELETE /light_switches/id/12345 is more sensible to do!

How do I use an HTTP API?

- There's software for PC (all platforms) like Insomnia (a REST client… hence the software-nerd pun), or Python requests library
- This will let you pick the request type (GET/DELETE/POST etc.)
- A web browser can make a GET request if needed
- You can also use telnet (yes, really) – telnet to port 80 for a plain HTTP web server – that is how you'll likely do it on a microcontroller!
- In HTTP, you open a connection, say what you want, get a response, and the server generally closes the connection immediately – nice and simple to handle!

Things to think about…

- Versioning APIs is a good idea (in the real world) – hard to get all client devices updated to latest version. Impossible when dealing with the general public (see people still using Windows 7/XP!!)
- This helps you evolve your API without breaking compatibility
- Design your APIs to be stable and consistent – think to the future, but don’t over-complicate the present.
- i.e. should it support one device, or multiple devices?

Designing an API

- Let’s make an API for checking if a door is locked or unlocked…
- What verb should we use?
- GET…
- GET /door/status ?
- This works, but what if we have a front and back door?
- GET /door/front/status ?
- Better! This assumes “front” is globally unique – what if 2 houses?
- GET /door/12/status
- Much better! If we swap doors around, we don’t have to find a way to change the door name, the door ID can remain the same and we can change the mapping to an ID in the user interface or software

Designing an API

- Now let's make an API for locking and unlocking the door!
- What verb should we use? GET? POST? PUT? PATCH?
- PUT is probably the best verb here (but PATCH/POST ok too I guess)
- PUT /door/12/status (in the request body, put locked=1 etc.)

- Notice from our URL we are GET'ing and PUT'ing to /status in both cases – this is fine, one reads it; one writes it

- If you design a lock/unlock API though, what would you need to think about?

API Security

Many people get this wrong… Huge numbers of major breaches happen as a result of this – being able to request any item from the API, and getting access to it…

You need to think about security – access control and authentication – understand who is making the request, and what they are permitted to do

Step 1 is to use HTTPS – encryption and authentication of connections to/from the server. Little point in securing it if you don’t do this!

Securing APIs

- When an HTTP request is made, you can send a header – commonly used is “Authorization”, but you can use what you like – “API_KEY”?
- The application on the server-side can make decisions based on this header’s value (and whether it exists or not) – link doors to API keys?
- Be sure to think about authenticating messages from the door to the server, as well as the user to the server – both need to be secure!
- Don’t use the same API key for both (!)
- If users control via a web browser, use a session cookie instead of an HTTP header, and this will be sent for you automatically
- Don’t allow GET verb requests to do anything dangerous!! Why?...

Securing APIs (a bit more)

- GET requests are the default made by a web browser. If you trick a user (via phishing email etc.) to click a link to:
- https://your-server.com/api/v1/door/12/unlock
- The door will unlock if they're logged in, as their cookies are sent!
- Similarly, don't have https://your-server.com/api/v1/door/12/status return anything like a "door unlock pin code"…..
- I'm looking at you, cheap IoT device makers who don't get it!!!
- Make sure users with access to door 12 don't have access to others!
- If you are interested in more about this, look into CORS (Cross-origin and same-origin policy (stops another website asking the browser to visit /api/v1/door/12/unlock or whatever…)

Handling Errors

You can return HTTP status codes to indicate whether something worked or not

Convention: 200 = OK, 201 = Created, 403 = Forbidden, 404 = URL not found

Spot a trend?

2xx = success

3xx = redirects

4xx = client error

5xx = server error

420 = enhance your calm…

418 = I’m a teapot… (can’t brew coffee, as opposed to a coffee machine)

Handling Errors

Document every error your system can generate, and communicate this to users of your API!

Handle errors elegantly in your API client! You can’t easily go in and look at it to find out what’s going on!

Design your server to accept as broad an input as possible – you want to make it easy for people to talk to you!

Overall Design

Ensure everything talks through the APIs – the “Amazon API mandate”

You don’t want parts of your system talking to each other through other means

If you were going to do that, why bother with an API in the first place?!

If you implement a web interface to control your server, it should use the standard API!! Don’t add secret undocumented functionality (ahem, Google)

You probably don’t want random client devices reconfiguring settings that the user is meant to be setting – security & access control

A little bit about security…

(This class doesn't have an exam… If it did, this bit wouldn't be in the exam… But it is very relevant to any kind of professional software development, and it's the sort of thing you will be expected to know or learn if you develop software "for real"!)

Through confined
inputs and is making
a sound lit. Your
pote 23…

We've been building closed-loop systems

- Heavily constrained inputs and outputs. You could "exhaustively validate" your code fairly easily!
- What do we mean by validate? We could build a test matrix with every possible combination of inputs, and work out in our head what we expect to happen… Then sit and test it manually
- We could automate this to some extent by "scripting" our code to respond to given stimulus. (That wouldn't test our hardware though)
- Think truth tables of a logic gate – fairly manageable in complexity!
- Even your lab 5 exercise has fairly easily validated/verified logic.

Delightful things we handle in the real world

- Text that users have supplied us… (Users are BAD NEWS!)
- Text that the internet has supplied us (THE INTERNET IS EVIL!)
- Stuff that other devices are sending us (usually as some kind of binary stream, or strings – like a CAN bus on a vehicle)

- When we talk to other people or things, we have to think how they might screw up, or do things we don’t expect them to do!

A Common Design Pattern (DO NOT DO THIS!)

```
void process_http_response(char* http_body) {
char* response = char[256];
int len = 0;
// strcpy(dst, src) is the syntax for strcpy. Copies from src to dst
len = strcpy(response, http_body); // strcpy copies memory until it finds a null byte
//...
}
```

- What's the issue here?
- What if http_body has 1500 bytes? You copy it over response
- And then some…. As you copy it across all sorts of other stuff!

How bad could it be???

Quite bad…

How could someone weaponise this?

- Think about when you wrote code in assembly
- Where in memory was response* created?
  - It’s stack memory (as it’s allocated here in a function, with a fixed, known size)
- If it was allocated dynamically (malloc/calloc), you’d get heap memory
- Heap memory is accessible “globally” as it’s just an address
- Stack memory is “scoped” to the function you are in
- Jump into your disassembly, look at heap/stack addresses
- Heap memory is always there. Stack memory will be created (based on the current value of SP, the stack pointer)

How could someone weaponise this?

- Whenever we call a function (CALL in asm, invoking a function in C)
  - We decrement the stack pointer (SP) by 2 (Remember, stack counts “up”!)
  - We write the “return address” (current PC value) to the (now moved) SP
  - We set the program counter to the operand to CALL
- This is how interrupts work behind the scenes – that’s how you end up back in the middle of your code after calling RETI!
- If you want to see it happen yourself, do a function call (or set an interrupt), then watch the stack and look at the SP value change, then go into memory view and watch an address appear… Go back to that address, see what it is, then notice you return there :)

How could someone weaponise this?

- How do we pass parameters when we call a function?
- Well – we actually pass them around on the stack! Your “calling convention” (C compilers have one) dictates exactly how this works – expect to see PUSH and POP instructions to do this.
- Then we can put some other variables into the stack (like we did with response).
- BUT if these overlap existing stack memory, we can overwrite other memory… Including things like the stored PC (i.e. return address)!
- So we could dump some bytes that are assembled instructions into memory, and adjust the stored PC to jump to them on return!
- And now we’ve only changed some “data” input, but have introduced arbitrary ASM into the program!

OK so what?

This is called a “buffer overflow” or “stack overflow” in particular.

Not to be confused with…

(who named themselves after this phenomenon)

CVE-2015-0235 – GHOST: glibc gethostbyname **buffer** **overflow** (http://www.openwall.com/lists/oss-security/2015/01/27/9)
531 points | martius | 7 years ago | 241 comments

Glibc getaddrinfo stack-based **buffer** **overflow** (https://googleonlinesecurity.blogspot.com/2016/02/cve-2015-7547-glibc-getaddrinfo-stack.html)
502 points | 0x0 | 6 years ago | 439 comments

Heap-based **buffer** **overflow** in Sudo (https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-**overflow**-sudo.txt)
400 points | ptype | 10 months ago | 77 comments

StarCraft: Remastered – Emulating a **buffer** **overflow** for fun and profit [pdf] (http://0xeb.net/wp-content/uploads/2018/02/StarCraft_EUD_Emulator.pdf)
249 points | jsnell | 4 years ago | 33 comments

Avast Antivirus Remote Stack **Buffer** **Overflow** with Magic Numbers (https://landave.io/2017/06/avast-antivirus-remote-stack-**buffer**-**overflow**-with-magic-numbers/)
245 points | landave | 4 years ago | 54 comments

CVE-2015-8126: Multiple **buffer** **overflow**s in libpng (https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-8126)
205 points | pjmlp | 6 years ago | 84 comments

Interactive guide to **Buffer** **Overflow** exploitation (https://nagarrosecurity.com/blog/interactive-**buffer**-**overflow**-exploitation)
169 points | bordplate | 2 years ago | 30 comments

How security flaws work: the **buffer** **overflow** (http://arstechnica.com/security/2015/08/how-security-flaws-work-the-**buffer**-**overflow**/)
154 points | guardian5x | 6 years ago | 45 comments

Buffer **overflow** when pwfeedback is set in sudoers (https://www.sudo.ws/alerts/pwfeedback.html)
142 points | masklinn | 2 years ago | 160 comments

Baron Samedit: Heap-based **buffer** **overflow** in Sudo (CVE-2021-3156) (https://www.openwall.com/lists/oss-security/2021/01/26/3)
120 points | jwilk | 10 months ago | 34 comments

Cisco **buffer** **overflow** vulnerability with remote code execution (https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20160210-asa-ike)
111 points | silenteh | 6 years ago | 20 comments

OS X 10.11 **buffer** **overflow** with deep filesystem hierarchy (https://cxsecurity.com/issue/WLB-2015100149?)
109 points | ivank | 6 years ago | 43 comments

Heap-Based **Buffer** **Overflow** in Sudo (https://blog.qualys.com/vulnerabilities-research/2021/01/26/cve-2021-3156-heap-based-**buffer**-**overflow**-in-sudo-baron-samedit)
94 points | lima | 10 months ago | 210 comments

Sudo **buffer** **overflow** explained, and how to patch (WSL and Linux) [video] (https://github.com/mbcrump/CVE-2021-3156/blob/main/README.md)
90 points | mcrump | 10 months ago | 37 comments

Bitdefender Anti-Virus: Heap **Buffer** **Overflow** via 7z LZMA (https://landave.io/2017/08/bitdefender-heap-**buffer**-**overflow**-via-7z-lzma/)
83 points | landave | 4 years ago | 58 comments

Multiple Heap **Buffer** **Overflow**s In the Windows DNS Client (https://www.bishopfox.com/blog/2017/10/a-bug-has-no-name-multiple-heap-**buffer**-**overflow**s-in-the-windows-dns-client/)
83 points | dijit | 4 years ago | 30 comments

Nginx SPDY heap **buffer** **overflow** (affects 1.3.15 – 1.5.11) (http://nginx.org/en/security_advisories.html?1.5.12)
77 points | jvt | 8 years ago | 29 comments

BSD libc contains a **buffer** **overflow** vulnerability (https://www.kb.cert.org/vuls/id/548487)
74 points | kumaranvpl | 5 years ago | 13 comments

CVE-2014-6273: **Buffer** **overflow** vulnerability vulnerability in apt-get (https://lists.debian.org/debian-security-announce/2014/msg00219.html)
72 points | anon1385 | 7 years ago | 36 comments

Python socket.recvfrom_into() remote **buffer** **overflow** (http://pastebin.com/raw.php?i=GHXSmNEg)
71 points | casca | 8 years ago | 19 comments

How to explain **buffer** **overflow** to a layman (http://security.stackexchange.com/q/53878/36538)
70 points | egsec | 8 years ago | 31 comments

VLC media player 3.0.6 and earlier: Read **buffer** **overflow** and double free (https://www.videolan.org/security/sa1901.html)
70 points | qndev | 2 years ago | 12 comments

How can we fix it?

- We can do input validation
- When working with annoying structures (C strings!!) we need to ensure we use “safe” functions to limit how many bytes we copy
- strcpy() is UNSAFE. strncpy() is safe, as it requires you specify a “max length” beyond which it won’t copy.
- It will truncate your string, but won’t overwrite other memory!
- Or think about other, safer programming paradigms (if appropriate) – maybe make a struct combining a C string, and len_t value for length
  - But… what if someone makes length longer than the C string is?
  - Oopsie, you could get an arbitrary memory read… Which could let you read secrets or keys out the program!
- Careful design and understanding of assumptions is key!

This is just one type of vulnerability!

There are dozens or hundreds more… Some won't be discovered yet (!)

I alluded to a second one there (arbitrary memory read)

Integer buffer overflows are another good one

Loop variable iterator modifications

Logic bugs in input handling/validation on edge-cases (i.e. len=zero)

Writing secure code is a whole topic in itself! But being aware of these kinds of things is the best starting point!

Big Danger Areas

- Dynamic memory allocation (malloc/calloc, free)
  - Use-after-free attacks
  - Double-free (null pointer dereference)
- String handling (variable length strings in particular)
- Untrusted external input handling
- Any serialisation/deserialisation of data (i.e. encoding/decoding from binary structures or JSON or similar)
- Type confusion (is that integer signed or unsigned? Does it matter?)
- Database/file access and error condition handling