Network Security Assignment

Submitted for Andrew Paverd

December 5, 2023

HTTP/2-to-HTTP/1 Conversion Anomalies

a) In your own words, explain why HTTP/2-to-HTTP/1 conversions take place, and describe the technique used in this paper [12] to systematically study conversion anomalies.

HTTP/2-to-HTTP/1 conversions occur due to a lack of support for HTTP/2 in the back-ends of proxy servers. The problem is pervasive due to these proxy servers being used as CDN Edge Servers. The problem is persistent due to an unwillingness among vendors to rectify the lack of support for HTTP/2.

CDN Edge Servers cache static resources and sit in between the Origin server, which hosts the definitive copy of a website, and a client requesting content from that server. Edge Servers are globally distributed, so a client is usually much closer to an Edge Server than they would be to the Origin Server, providing lower latency responses. When the request cannot be satisfied by static resources alone, then the Edge Server behaves as a proxy and routes the traffic to the Origin Server, providing latency not much worse than if it weren't there.

Figure 1: *Request being intercepted by a CDN Edge Server, avoiding need for a round-trip to the origin[7].*

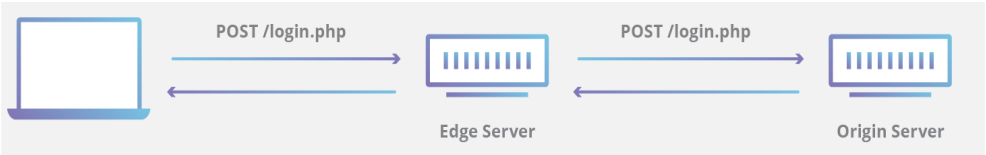

Figure 2: *The necessary round trip if the request requires more than just static resources to be returned[7].*

The traffic flow in Figure 2 has created a problem in practice. The Edge Server generally communicates with the client over HTTP/2. However, among vendors there is very little support for HTTP/2 for the proxy-to-origin leg. If the client request arrives via HTTP/2, and the server must forward it via HTTP/1, then it must translate the HTTP/2 request to an HTTP/1 request. Ie. An HTTP conversion.

The reasons for this lack of support are varied. Nginx cite the need for significant engineering work to implement it, and a lack of benefit, since despite HTTP/2 being multiplexed, it still moves over a single TCP connection, which isn't multiplexed[14]. Akamai seem to disagree, with their organisation openly championing the widespread adoption of HTTP/2[5]. With that said they appear not to live up to their word, with multiple forums suggesting they intend to support edge-to-origin HTTP/2, but the feature has seemingly yet to materialise[10][8]. This suggests a lack of a strong business incentive or shifting organisational priorities as the reason, rather than a lack of a technical benefit. Bucking the trend Envoy seem keen to tout their support for not only HTTP/2 but HTTP/3 and QUIC[9], suggesting perhaps nothing but industry inertia and time is holding other vendors back.

The paper[12] makes use of fuzzing as the technique of choice to systematically study the HTTP conversion anomalies that arise in a range of different proxy servers. That is to say, they produce a wide variety of malformed HTTP/2 requests and send them to each of the proxies they wish to study, then inspect the traffic that is converted to HTTP/1 and returned, and compare the traffic that has been returned to what they know the corresponding request ought to look like. Inconsistencies are flagged as anomalies.

To produce the HTTP requests required, the authors have defined a small context-free grammar[6] (CFG) for HTTP requests. A CFG in this case models an HTTP frame as an abstract syntax tree (AST), where child nodes are created from parent nodes according to 'production rules'. Taken together these rules comprise the grammar. This resolves to meaningful output when a production rule produces a leaf node, aka. a terminating symbol. When read left to right the terminating symbols make up the HTTP request that has been generated. The AST above them describes their structure and how they can be logically grouped together according to the grammar. Some of the production rules are nondeterministic. Therefore they produce something different every time they are invoked according to some fixed probabilities defined in the rule. In this way the authors have devised a tool to produce an arbitrary number of different valid HTTP/2 frames.

However, valid HTTP/2 frames aren't hugely useful in searching for security issues. Really the authors want to analyse the behaviour of the proxy servers when they are forced to convert *invalid* HTTP/2 frames. Once the Frameshifter tool has produced an HTTP/2 frame, it is 'mutated' a configurable number of times. A mutation manifests in one of two ways, either the string that forms a terminating symbol, ie. a header name, header value, request type, etc. is changed, or the structure of the request is changed. To alter the structure of the request the authors randomly edit a branch of the AST, either by adding a different subtree beneath it, or removing the nodes beneath it entirely and making it into a terminating symbol. By combining the approach of editing terminating symbols and editing the structure of the request the authors are able to produce large numbers of arbitrarily malformed HTTP/2 frames to feed to the proxy servers.

b) In your own words, briefly summarize the main security attacks reported in the paper, and for each attack, explain how one or more conversion anomalies could be used to mount the attack.

Incomplete Body Denial Of Service: The authors pursue a strategy of using conversion anomalies that produce frames with incomplete bodies to induce denial of service (DoS). This strategy yields three results:

If the body is incomplete, then the header will suggest the frame is larger than it is, so the server will wait for the remaining bytes that will never arrive until timeout. If the timeout is long enough, then only a relatively small number of these malformed packets need to be sent in order to exhaust the resources of the server. This **Timeout-Based DoS** was found to be viable for several reverse proxy/origin server combinations. The anomalies that consistently lead to DoS were 'incomplete content-length with body', 'incomplete

content-length without body', and 'missing last chunk'. For a particular combination the anomalies 'missing chunk data' and 'missing chunk data termination' were also found to be effective.

Another consequence observed for a few specific combinations of proxy and origin server when the incomplete body anomaly was 'incomplete content-length without body' was **Request Blackholing**. After the malformed request is sent, subsequent unrelated requests are then treated as being part of the incomplete body of the malformed frame, and are accordingly either dropped as invalid, or lost altogether, ie. blackholed. The authors note that in other scenarios this could lead to a more serious attack, however since the only conversion anomaly that produces this forwards a request with no body, it is only good for DoS.

For one particular reverse proxy (Caddy) the timeout logic seems not to have been implemented properly, causing the request with an incomplete body to never timeout at all. This means the malformed request, when sent once per persistent connection, will act as a **Query of Death**, leaving each connection in an unrecoverable hung state. The result is the proxy becoming completely unresponsive, even to system commands. The conversion anomalies that were found to exploit this vulnerability were 'incomplete content-length with body', 'incomplete content-length without body', and 'missing last chunk'. This is a more serious DoS than the previous ones, since they rely on an attacker being able to continuously send malicious packets to sustain the DoS. This offers a one-and-done DoS, with likely just a few dozen packets to take the server down permanently.

Cache-Poisoned Denial of Service: The principle here is to manipulate the cache of a reverse proxy to inaccurately store an HTTP error response for a page/resource that does in fact exist. The objective is to find a conversion anomaly with a request to a legitimate page that is handled and forwarded by the proxy but causes an error to be returned by the origin. The authors found that when proxying through ATS (Apache Traffic Server) the conversion anomaly 'repeating header value' caused HTTP 501 errors to be returned by the origin. A 501 response code is cacheable by default[11], therefore in principle having this error returned for an HTTP method ought to poison the cache for that server and method for all resources. This attack was found not to be viable for any of the proxy servers tested. However, it remains an attack reported on, and negative results are legitimate and useful. The authors speculate that the error would only have been cached if the poisoned request was the first one made with that HTTP method against that server. Since results with that method had already been cached for that server subsequent errors are being ignored.

Response Queue Poisoning: This attack relies on a form or request smuggling. Which is to say, making multiple HTTP requests look like just one. The scenario presented in this attack is that a conversion anomaly causes multiple HTTP/1 requests to be generated from one HTTP/2 frame. The principle being that this presents a possibility of request smuggling, and therefore causing an off-by-one error in the request-response matching. The conversion anomaly that could exploit this vulnerability is 'multiple forwarded requests'. While an interesting idea overall the authors were unable to observe a response to a victim request being sent to an attacker. The authors speculate that this was due to lack of scale. This is potentially wishful thinking, since this attack is highly reliant on assumptions about the servers back-end data structures. If the proxy implements any kind of isolation between client connections, eg. by creating a separate buffer per client with their own context, or by opening a new connection with the upstream server per client, this would also cause this attack to fail. Either of these are also plausible explanations for why the authors attack was not successful.

HTTP Request Smuggling: An attack that relies on ambiguous parsing of an HTTP requests body, that causes two HTTP servers to disagree on where the body ends and the next request begins. An example of how this can be exploited is provided by Portswigger[15]: Bypassing Front-End Access Control. Briefly, if

the reverse proxy has been entrusted with access control for subdirectories, with the origin server trusting requests sent to it implicitly, then an HTTP request to an unprivileged subdirectory that successfully smuggles another request to a privileged subdirectory can retrieve content it was not authorised to. In principle, a conversion anomaly that impacts any of the headers that are often associated with HTTP request smuggling, notably headers related to body parsing (eg. content-length, transfer-encoding), give rise to potential request smuggling. Candidates to this effect were the 'invalid header value' and 'repeating header name' conversion anomalies. The authors opted for the same starting point as they did for their attempted Response Queue Poisoning, which is to say looking for requests where sending one request resulted in two responses being returned. While the authors did discover apparent request smuggling issues, they concluded that none of them had arisen as a consequence of a conversion anomaly, and were in fact just more generic request smuggling vulnerabilities. They reported them all the same since the paper is just as much about presenting their fuzzing tool, as it is about conversion anomalies. Sadly however this means these vulnerabilities cannot be discussed in the context of a conversion anomaly, and therefore leaves them beyond the scope of this question.

Public WiFi Security Scenario

Bob is using his laptop in a coffee shop to browse the Web. His laptop is connected to the Internet via the coffee shop's Wi-Fi network, which uses WPA2 in pre-shared key (PSK) mode with a strong password that is only given to customers. Bob is not using a VPN, but he takes care to only visit HTTPS websites. He observes all warnings provided by his Web browser, and his laptop's software is fully up to date with security updates. Eve is also a customer in the coffee shop, and thus she also has the password for the coffee shop's Wi-Fi network. Eve is proficient in offensive network security.

a) Can Eve ascertain which websites Bob is visiting using only network-based attacks?

A detail that is important in every attack that follows that we omit there but will mention here is as follows: In order to make use of the data from any of these attacks it is important for us to identify which traffic is coming from Bob and which is associated with other users, otherwise we will not know who we have spied on! This will likely be more of an art than a science. We need to correlate the behaviour of customers in the shop with metadata associated with their network traffic. The most obvious thing we can do is to take a note of what time each customer sat down and what time they left, then look for which MAC addresses traffic fits best into that interval. If there is ambiguity we can collect more metadata, for instance taking note of any moments that customers went to the toilet or went to order another coffee. This data may be noisy/sparse and difficult to analyse real-time, making it difficult for Eve to be sure they are targeting Bob at the time. To get around this there is no reason she cannot simply run her attacks in a scripted fashion against every user on the network. For simplicity we will assume that this is how she proceeds, with her analysing the findings and deanonymising Bob using other metadata after the fact.

Deauth and Decrypt
WPA2-PSK makes use of strong encryption (AES-CBC)[1] for which there is no known, viable, conventional-hardware attack. Therefore we must obtain the symmetric key being used to encrypt traffic between the router and Bobs laptop:

1. Buy a coffee, obtain the WiFi password, take a seat.
2. Configure a packet sniffing tool on a device with a suitable network card to capture all ambient traffic. Eg. with Wireshark this can be done by enabling 'monitor' mode.
3. Capture the 4-way handshake[16] between Bob and the router:
   (a) If Eve arrives before Bob and is already capturing packets this is trivial, just comb through packets captured and identify the packets that comprise the 4-way handshake.
   (b) If Eve arrives after Bob, or for any reason fails to capture his 4-way handshake when he connects, then she can force Bob to repeat the exchange by sending an 802.11 disassociation management frame to Bob, spoofing the router in order to do so, that informs his device that it has been deauthenticated. This will be interpreted as a unilateral action by the access point prompting no response from Bob. This will prompt Bob to reauthenticate, and give Eve another opportunity to capture the handshake. This may require Bob to actively reconnect, we reasonably assume that he would do so if he was informed he had been disconnected. A tool such as `aireplay` from the `aircrack-ng`[4] suite would be suitable for this.
4. Compute the Pairwise Transient Key (PTK) that is in use as the symmetric key for the packets being exchanged between the router and Bob. The PTK is computed deterministically as a function of the

Pairwise Master Key (PMK), which in this case will be the WiFi password that Eve obtained when she purchased a coffee, a nonce computed by the router (ANonce), a nonce computed by Bob (SNonce), and the MAC addresses of the router and Bobs laptop. With the exception of the PMK, all the information required to compute the PTK is exchanged during the four-way handshake. The detail of precisely how to compute this is well documented elsewhere[2]. For her purposes Eve can again make use of `aircrack-ng` or a similar tool to compute the PTK.

5. Decrypt the `.pcap` files captured. With the PTK in hand this is just a matter of appropriate tooling. Both `airdecap-ng`[3] (from the `aircrack-ng` suite) and Wireshark[17] actually roll this and the previous step into one, as long as the PCAP you want to decrypt is the same one that contains the 4-way handshake.

With this layer of encryption removed there are at least four ways the privacy of Bobs traffic is compromised:

1. DNS queries and responses will be visible in plaintext. This will reveal the domains of any sites Bob browses to that aren't in his browsers local DNS cache.

2. Initial HTTP GET requests that Bob might make by typing in a URL directly into his search bar/making use of any bookmarks will be visible in plaintext, revealing not only the domain he browsed to, but a full query URL.

3. The IP addresses that Bobs TCP traffic goes to will be visible. Though due to the widespread use of CDNs, cloud providers and load balancers this may be of limited use. Knowing that traffic is bound to an IP address owned by AWS isn't hugely useful in trying to compromise the privacy of a users traffic.

4. OCSP requests and responses will be visible in plaintext. These do not directly reveal browsing data, however they contain the serial number of a certificate, and the certificate issuer. Together these can be used to look up the certificate, which will contain a domain/subdomain.

Man In The Middle

An alternative to the deauth and decrypt attack that achieves the same result is for Eve to position herself as a Man In the Middle (MITM).

1. Bring another 'rogue' router concealed in a rucksack with her into the coffee shop

2. Once the WiFi password is obtained, configure the rogue router to have the same password as the legitimate one, make sure to also name it appropriately, with something like: `<insert legitimate router name> 5GHz`. This will make users more likely to connect to it.

3. If Bob arrives after and connects right away to the rogue router then we are done, otherwise we have two options:

   (a) We can send deauth 802.11 management signals to Bob repeatedly with a tool like `aireplay-ng`, to convince him that his connection is unstable, and prompt him to consider connecting to the alternative WiFi.

   (b) We can use ARP spoofing to force Bob to connect to the rogue router instead of the legitimate one. Using a tool like `arpspoof` we can broadcast that the IP of the router is now associated with the MAC address of the rogue router. If we begin to see unintelligible encrypted traffic arriving we will know that we have successfully poisoned an ARP cache. At this point we deauth as before, then a new 4-way handshake ought to be initiated, as Bob now connects to the rogue router while still thinking he is connecting to the legitimate one.

Website Fingerprinting
Website Fingerprinting is an attack that uses the size, distribution, timing, and other metadata of packets to deanonymise users browsing the internet and correlate them across multiple sites. Our scenario reverses this. We know who the user is, and it is the websites they are browsing that we want to deanonymise. This area is relatively under-researched, since the use-case is quite niche. It would only make sense if the adversary controls neither the network, nor the users endpoint, and is satisfied with a probabilistic result rather than an authoritative one. Never-the-less it is conceptually plausible for the same reason that conventional Website Fingerprinting attacks are.

1. To mount this attack Eve should procure a device as similar to Bobs as possible, and connect it to the coffee shop WiFi. She should make a careful note of what the PTK is.

2. On her own device, she should begin a packet capture that will capture everything that goes over the network.

3. On the Bob imitation device, she should run a script that queries the 1000 most popular websites, and any other websites of interest, and performs basic activity on those websites programmatically via a testing framework, eg. Selenium. The script should log the key that is agreed upon in TLS to encrypt the TCP streams.

4. The packet captures from this exercise should be parsed into a columnar database, with the number of the packet as the primary key, and with several columns of metadata associated with each packet.

5. The packets should be decrypted, first at the WPA2 layer, then at the TLS layer, using the encryption keys logged. From the decrypted data, the website that was being browsed to should be parsed for each packet from the HTTP frames, and added to the columnar database. In this way we now have a labelled dataset.

6. Machine learning model development follows. Derived features (eg. time between packets) should be computed. Dimensionality reduction, PCA and manual feature selection should be considered. The data should be split by training, validation and testing, and fed to the appropriate state-of-the-art ML algorithm (whichever one that happens to be this week).

7. Eve should now parse the metadata from her packet capture of Bobs traffic and feed it to her classifier. If the model development was successful and generalises well, she should be able to accurately classify Bobs packets with some percentage certainty.

Comparing the attacks listed, we should more or less immediately discount the proposed **Website Fingerprinting** attack. While conceptually fun it is not proven to work, and would be prohibitively difficult. It would also not provide us with authoritative answers to Bobs browsing activity, only statistically likely ones. This leaves the **Man in the Middle** and **Deauth and Decrypt** attacks. They achieve fundamentally the same thing, decryption of packets associated with DNS, OCSP, initial GET requests, and IP ranges. So the only basis on which to compare them is their likelihood of success, ease of implementation and stealth. On all three **Deauth and Decrypt** is the clear winner.

1. **Likelihood of Success:** Deauth and Decrypt is virtually guaranteed to succeed, the only risk is that a single deauth causes Bob to become frustrated and instantly get up and leave. By contrast the MITM approach carries risk of failure in either of the two scenarios: Repeated deauthing trying to get him to connect to a different router is far from guaranteed to work, he is just as likely to get up and leave, or connect to his phone serving as a WiFi hotspot. ARP spoofing is more promising, however it is still not guaranteed. The legitimate router may regularly broadcast ARP signals itself, causing Bobs ARP

cache to frequently switch back and forth. This again risks Bob becoming frustrated and leaving as the instability may be visible to him.

2. **Ease of Implementation:** A clear win for Deauth and Decrypt. For that Eve only needs a device capable of listening to ambient traffic and sending 802.11 deauth frames. For MITM she will need a rogue access point that has been specially configured to capture and forward traffic.

3. **Stealth:** Both attacks involve sending 802.11 deauth frames. However, MITM involves sending a great deal more malicious traffic on top of that. The ARP spoofing, rogue access point itself (which could be discovered by the shop owner at any time), and copious additional deauth frames all make this a significantly less subtle attack than Deauth and Decrypt.

b) Bob is concerned that someone in the coffee shop might be able to monitor his Web browsing. What network-based defences could he use to prevent or reduce the impact of the attacks identified above?

arpwatch: This tool provides a detective control against ARP spoofing. A preventive control is difficult to implement against ARP spoofing since the protocol does not have any notion of authentication built into it. However this detective control can watch the network for multiple responses to ARP requests, ie. multiple actors with different MAC addresses both claiming to have a particular IP address. This provides a good defence against the version of the **MITM** attack that relies on ARP spoofing to redirect traffic to Eve's rogue access point. While it does not give any way of enabling Bob to browse confidentially, it will very likely inform him of the event that he is currently subject to a MITM attack. A limitation of the tool is that it is likely to throw a great deal of false positives, a notable case in which it will be confounded is for any users on the network making use of a virtual machine (VM). This can cause a device with just one MAC address appear to be associated with multiple IP addresses. This would be legitimate activity, but may be flagged by `arpwatch` as suspicious. Another limitation is that the results may be quite difficult for anyone who is not an expert in offensive network security to interpret.

Virtual Private Network: Use of a VPN would be a strong mitigation for both the **MITM** and the **Deauth and Decrypt** attacks, since this will provide a layer of encryption between the VPN server and Bobs VPN client on his machine. This will entail providing an additional layer of encryption for the DNS, initial HTTP, and OCSP traffic that was previously compromising the privacy of Bobs browsing. The VPN will also obscure the IP ranges that Bobs packets are destined for. All that Eve will be able to see is that his packets are being routed to a known VPN server. This will likely tell her which VPN service Bob is using, but absolutely nothing else. All she will be able to glean is metadata about the packets that Bob is sending.

Random Bidirectional Padding: While the **Website Fingerprinting** attack is unlikely and potentially prohibitively difficult to mount, it has the benefit that neither a VPN nor even a Tor-based browser alone directly mitigates it. However a technique called Random Bidirectional Padding (RBP) has been shown to be an effective defence against conventional website fingerprinting[13]. The principle will be the same as with defending against conventional website fingerprinting attacks: obscure and mutate the traffic until the features have become useless to the model and could no longer possibly explain the variance. More precisely, we inject noise into the traffic to confound models and destroy Inter-Arrival Time (IAT) features. This could be implemented as a browser extension and made available to Bob. It could also be bundled as an additional feature implemented by a VPN client he can use. Since the attack is theoretical so are the defences, but if we entertain the possibility of such a website fingerprinting attack then this is the appropriate defence.

c) Daniella, the owner of the coffee shop, is actually a highly proficient network security researcher. She has configured the coffee shop's Wi-Fi router to record all traffic passing through

the router for subsequent analysis using tools like Wireshark. What would she look for in the captured traffic to detect the attacks identified in part a)? Are there any attacks she would not be able to detect using only the captured traffic?

To identify the **Deauth and Decrypt** attack, ideally Daniella would have configured the router to record all traffic in something equivalent to monitor mode in Wireshark, so that she would capture ambient traffic that was not necessarily destined for the router. This would allow her to log the activity of the `aireplay-ng` tool as it sends the dissociation frame to Bobs device, spoofing the router in the process. Unfortunately, supposedly Daniella is only logging traffic that moves through the router, and because Bob will not have sent any kind of response to the spoofed dissociation frame, Daniella will see no record of the malicious activity. However, Daniella will see something that ought to be a very strong indicator of such a deauthentication attack. She will see Bob attempting to reconnect with a fresh 4-way handshake, in spite of the fact that the last time he exchanged traffic with the router was less time ago than the session key timeout, and as far as Daniella can see no dissociation frames were exchanged. This won't be definitive, for instance it could indicate that Bobs device died for instance, and he has just begun recharging it, and when it died his device lost the session key. As we discussed above however, it is quite likely that Eve will not have time to work out precisely whose traffic she is intercepting, and is therefore running this attack against every device on the network. If Daniella observes a pattern of every device on the network needing to reconnect in spite of the session key timeout not having elapsed, she can be reasonably confident that someone was performing a deauthentication attack.

Detecting the **ARP Spoofing** will be much more straightforward. This attack entails broadcasting malicious ARP responses that announce that the IP of the router is associated with a MAC address that is not that of the actual router, thus looking to poison the ARP caches of devices on the network. Due to the broadcast nature of ARP this will be plainly visible to Daniella inspecting the recorded traffic in Wireshark. She can simply filter for ARP messages, and then filter again for ones associated with the IP of the router itself, and look for any that do not correlate with the application logs of the router, or that mention a MAC address that is unknown to her.

Detecting the rogue access point itself could be very easy, or quite difficult, depending on how much effort Eve went to in obfuscating her attacks. If Eve was lazy, and as we've already discussed, she attacked everyone in the shop in the same fashion in order to get to Bob, and forwarded everyone's traffic straight to the legitimate router to enable it to reach the internet, then the attack will be blatantly obvious. Daniella will recall serving a dozen customers with laptops at that particular moment, yet the logs will clearly show only one device connected, yet streaming a volume of packets comparable to that that a dozen devices would generate. Equally suspicious would be if Eve had opted to stream the traffic straight to the internet herself from her rogue access point via a mobile internet hotspot or similar device. At that point Daniella would be able to observe nothing but minimal baseline traffic moving through her router, in spite of a large number of customers who requested her WiFi password sitting in her coffee shop and using internet connected devices. One thing Eve could have done that might have made the traffic volumes and distribution appear normal would have been to run a compute cluster on her device, with her rogue access point serving as a load balancer, distributing the traffic to a number of different virtual machines, each with their own network interfaces, and therefore their own MAC addresses and IPs, which only *then* forward the traffic onto Daniella's router. This would create the impression of multiple distinct devices exchanging internet traffic, all of which would have performed normal 4-way handshakes with the router. If Eve's setup was configured such that there would be one VM provisioned per customer she was Man in the Middle-ing, then Daniella would be none the wiser. At this stage the only thing Daniella might have been able to do would be to have connected a known device to her own router, and observe that though traffic associated with that device is

clearly present, her MAC address is nowhere to be seen.

The **Website Fingerprinting** attack is entirely passive, and therefore impossible for Daniella to spot from only observing her own passively obtained logs.

Biotech Company Network Design

a) Identify security weaknesses and propose alternative design.

The security weaknesses identified with the proposed network architecture are as follows:

1. *The use of WPA2 in pre-shared Key mode (PSK).* An attacker who obtains the password need only intercept the 4-way handshake between a client and the router in order to decrypt the traffic moving over the air. This compromises a valuable layer of the defence of the network.

2. *Lack of network segregation between the WLAN used by guests and the WLAN used by staff and critical internal assets.* In most organisations a compromise of the guest WiFi network would be considered a minor breach, that is almost to be expected, since the password is effectively shared with the public. With this architecture however, access to the guest WiFi provides network layer access to the organisations most critical assets, notably the database with the patient medical data. This leaves a potentially useful layer of defence at the network layer compromised by design.

3. *Authentication over TCP for databases (potentially).* This means that credentials to authenticate against the organisations most valuable assets could be being sent over the air, with only the encryption provided by the WLAN to conceal them. While it is possible authentication is actually happening at layer 7 and there is SSL encryption concealing traffic end to end this is not explicitly stated, and therefore should be assumed not to be the case.

4. *No MFA for access to databases.* This means that credentials to obtain access to critical data can be phished, and subsequently used by an actor from any location, at any time, in any context.

5. *Every employee has credentials for both databases.* Access to sensitive data should be needs-based, it is unlikely that every employee needs this.

6. *Employee access to databases is not scoped to their role.* While true that it isn't explicitly stated that employees credentials are powerful, it is not stated that they aren't. Therefore we must assume that employees access to the databases is not done on a least privilege basis. Users should only have as much privilege as they need to perform their roles and no more. For instance, it is very unlikely that many, if any of them, will need the ability to drop tables, or write to the database manually in an ad hoc fashion.

7. *Use of a home broadband style router in an enterprise setting.* Particularly when considering that this would be coupled with a static IP address, this represents a security weakness since it provides very little scope for the implementation of protection against Denial of Service attacks against the network.

With these weaknesses in mind I propose the following alternative design, that looks to avoid the most serious among them, while maintaining a very simple, comprehensible, and maintainable architecture.

Cloud Architecture: Each cloud provider is different, and therefore so are the architectures that would make sense for each of them. This makes it difficult to discuss them in the abstract. Therefore here we specifically target AWS as our choice cloud provider. The requirements of the cloud portion of the company are quite vague, so we provide a generic architecture that is secure be default but should be easily extensible.

- We assume that the company is used to and would prefer to use a relational database, therefore we provide an RDS instance.

- The RDS instance has a security group that only admits traffic from an EC2 instance.

- The RDS is also configured to only allow user interaction based on AWS IAM roles. It cannot be authenticated to via username/password.
- We assume that direct interaction with the database will be relatively unusual, and that for the most part it will serve as a database to store the data for workloads that will be scheduled on the EC2 instance. In the event that a user does wish to interact directly with the database it will still be possible, but not straightforward. They will need to SSH to the EC2 instance via AWS SSM in the console, or from the on-prem network, and from there pass a short lived ephemeral token to the RDS endpoint to authenticate with the AWS CLI. This process is cumbersome by design, direct database access is poor practice and should ideally only be done in emergencies.
- The EC2 instance also has a security group. This only permits traffic to flow between it and the RDS instance, and it and a Transit Gateway.
- All of these resources (except the Transit Gateway) exist inside an AWS VPC, providing a private subnet for the resources that offers network isolation.
- An AWS Transit Gateway serves as a bridge, providing connectivity from the EC2 instance to an on-prem VPN server. This provides layer 7 encryption for all traffic that flows to and from the VPC, and has made it such that the VPC is only externally routable via the on-prem VPN.
- A caveat to the carefully laid network layer isolation is that all of these resources are still accessible and configurable via AWS APIs. Access to this will be controlled via SSO and MFA with the organisations chosen identity provider.

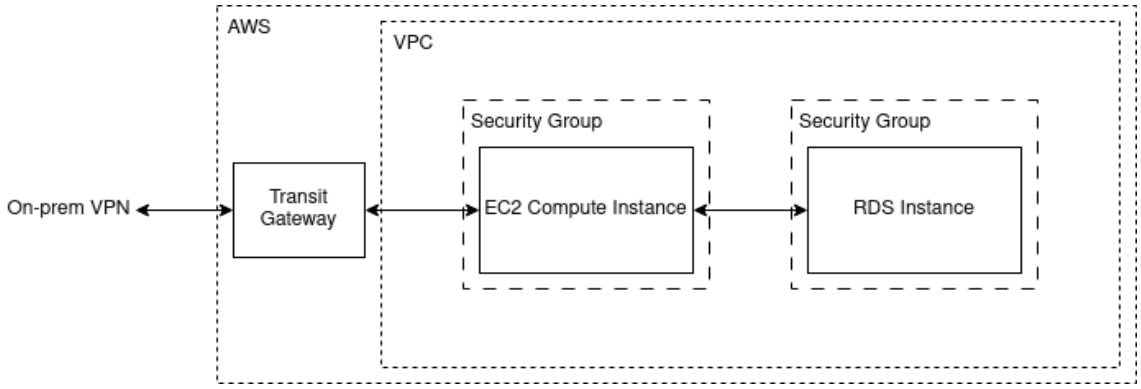

Figure 3: *AWS cloud architecture.*

On-Prem Architecture: While more fleshed out our revised architecture for the on-prem setup is only marginally more sophisticated. There is not as much network segregation as there could be. We wanted it as simple as possible. Motivated by a need to make the setup as maintainable as possible, we avoid adding security features for the sake of it, particularly where they add significant complexity for end users/network administrators and offer little real security benefit in return.

- The personal-use grade broadband router has been discarded in favour of an Enterprise Grade Router. This will enable Enterprise WPA2 to be the means of connecting to the WLAN, allowing per user authentication and enabling us to configure MFA based authentication to the WLAN. Since the authentication flow is handled by something closer to a web app, it will now not be possible to capture and recover the symmetric key in use for WPA2 encryption.

- The new router also puts us in a stronger position to mitigate the risk of Denial of Service attacks against the router. It will come with greater ability to handle large volumes of traffic out the box, and will likely also include features for traffic filtering and rate limiting.
- A final positive of the upgraded router will be a builtin VPN server, that will facilitate the secure connection with the AWS Transit Gateway. At present it is only configured to admit traffic to and from the Transit Gateway, however it opens up the potential to also serve as a secure gateway for users looking to connect to the office network from beyond the reach of the WLAN. This may be rendered necessary if they were to be forced to work from home due to the emergence of surprise pandemic for instance.
- For simplicity everything connected to the router is on the same VLAN. There is no network layer segregation between these components. This is because there is a legitimate basis for each component in the VLAN to interact with every other component, and because the alternative would introduce significant complexity for the network administrator while yielding relatively little security benefit.
- The heavy purple lines in Figure 4 indicate Ethernet. This aspect of the design is largely unchanged, with the exception of the introduction of a layer 2 switch to manage the number of physical Ethernet connections. An Enterprise Grade Router ought to have enough ports for the initial 10 employees, however a layer 2 switch serves as nice future-proofing for the event of company expansion.
- While the on-prem database remains reachable via TCP, the authentication will now take place on layer 7. Allowing for role based access control and authentication via SSO with MFA through the company's choice identity provider.
- A personal-use grade broadband router will be retained for the purposes of serving as a guest WiFi. It will have no routing connection to anything on the internal network. It will be a physically separate device. We considered simply implementing it as a separate VLAN on the main routers WLAN, however this overlooks a potential useful secondary function this WiFi can serve. In the event of a network outage of some kind, having continued access to the internet is extremely useful, therefore this second router will also serve to provide redundancy for the event that the main one fails. It will not be able to adequately replace it, however it will allow the network administrator to continue to research solutions, and potentially connect to one or two specific desktop machines, the on-prem database, or even the AWS console to enable emergency triage work.

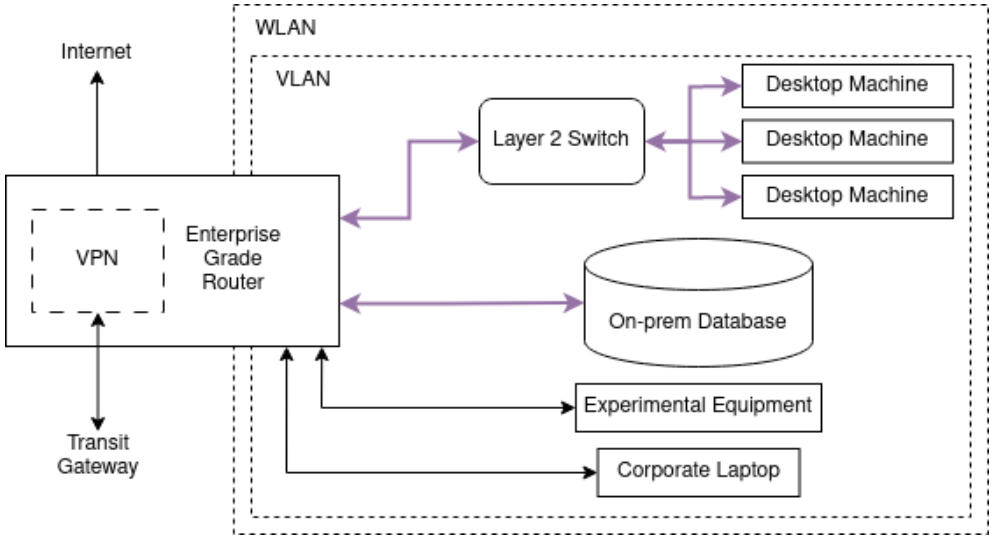

Figure 4: *Network layer architecture for on-prem setup.*

b) Discuss the advantages and disadvantages in terms of both efficiency and security of having employees connect to cloud assets routed via a proxy in an office versus direct connection to the cloud provider.

Efficiency advantage: If employees in an organisation are distributed around southeast England, with an office in London, but the primary data center they make use of for their cloud compute is in Ireland, then there is an advantage to be had from forcibly routing user traffic via an office proxy. The AWS-specific reasoning is that a Direct Connect service can be set up between the office and the data center, providing a high throughput, low latency connection between those two nodes. This will provide better latency for anyone who routes their traffic through the office proxy rather than trying to reach Ireland via the internet. This is not a concept that is unique to AWS' offering. GCP has an equivalent named Dedicated Interconnect. Azure has ExpressRoute. This is not guaranteed to always provide a noticeable difference to end-users, but it will certainly prove it's worth for any occasions where a significant amount of data transfer that will strain the bandwidth of a normal connection is needed.

Efficiency disadvantage: The Singaporean government has strict rules around whether and how data that pertains to Singaporeans can ever leave Singapore. So if our Biotech organisation hopes to offer services to Singaporeans, they must run their operation from a Singaporean data center. It is not at all a far fetched hypothetical to suggest that a company doing this kind of business might be based in London, and implemented the proposed proxy routing to simplify their network security setup before they signed the Singaporean client. The company is now in an awkward position. The customer environment and its data are in Singapore per the law, however that environment is orchestrated from, and only routable via, the corporate HQ in London. A support engineer in Singapore, or even the client themselves, in order to access an environment that is physically located less than 10 miles away from them, must route their traffic via a proxy on the other side of the globe, causing the traffic to take a round-trip that is absolutely unbearable for a number of basic support tasks. The obvious solution is to have a proxy per office, and for the traffic to be dynamically routed via whichever one is closest. However this is a significantly more complex setup, and a departure from the original proposal, that would entail significant engineering effort to migrate to.

Security advantage: Routing via a proxy that forces all traffic to move via an office could significantly reduce the organisations network attack surface. Where previously they may have had to implement Internet Gateways and have public IP addresses, they can now be completely concealed at the network layer within their VPCs. This massively hinders an attackers efforts at enumeration, and greatly simplifies the blue teams threat modelling, since they no longer need to consider any cloud assets to be a potential point of entry from a network security standpoint.

Security disadvantage: A single point of failure has been created for the security incident response team. If the office based proxy is taken offline for any reason, it becomes prohibitively difficult to access and protect cloud assets. A well resourced threat actor would know therefore that anything they can do to disrupt the proxy will cripple the organisations ability to respond to security alerts and support the product. As a result the impact of a DDoS attack becomes catastrophic. The mitigation for this is to invest heavily in anti-DDoS measures. This could include having the VPN hosted by a third party that specialises in this, such as Cloudflare. The router that serves the VPN and any infrastructure behind it could be upgraded to simply try and have the bandwidth to absorb the attack. Software can be purchased to automatically detect attacks and use strategies like dynamic packet filtering, rate limiting and null routing.

In summary, there are trade-offs. Depending on what exactly the Biotech company needs it could make different decisions and the reasoning would be sound either way. A well resourced organisation that is confident in their ability to afford the countermeasures to defend themselves against a massive DDoS may well opt to route all their traffic via an office based proxy. By contrast, an organisation whose data in the cloud are not as valuable to money-motivated criminal gangs and state backed groups, and/or has more of a focus on running a fast moving, globally distributed company, may well favour having its cloud environment routable, and trusting that the layers of authentication it has configured higher up the OSI stack are adequate protection.

References

[1] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1205–1209, 2012.

[2] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, page 1263, 2012.

[3] aircrack-ng. Airdecap-ng. https://www.aircrack-ng.org/doku.php?id=airdecap-ng/, 2009.

[4] aircrack-ng. Cracking WPA. https://www.aircrack-ng.org/doku.php?id=cracking_wpa/, 2019.

[5] Akamai Technologies Inc. Akamai Helps Drive Adoption of Faster and more Secure Web Protocol. https://www.prnewswire.com/news-releases/akamai-helps-drive-adoption-of-faster-and-more-secure-web-protocol-300179712.html/, 2015.

[6] Bahruz Jabiyev. Frameshifter. https://github.com/bahruzjabiyev/frameshifter/blob/main/conf/experiment_conf/, 2022.

[7] Cloudflare. Origin Servers. https://www.cloudflare.com/en-gb/learning/cdn/glossary/origin-server/, 2023.

[8] Community Migration. Timeline on HTTP/2 support from Edge to Origin. https://community.akamai.com/customers/s/question/0D50f00005RtrBmCAJ/timeline-on-http2-support-from-edge-to-origin?language=en_US/, 2016.

[9] Envoy Project Authors. HTTP/3 Overview Envoy Proxy. https://www.envoyproxy.io/docs/envoy/latest/intro/arch_overview/http/http3/, 2023.

[10] Eric Elbaz. Does Akamai Support HTTP/2 from origin to edge? https://community.akamai.com/customers/s/question/0D50f00006Qr9DqCAJ/does-akamai-support-http2-from-origin-to-edge?language=en_US/, 2019.

[11] Internet Engineering Task Force (IETF). RFC 9110 HTTP 501 Error. https://www.rfc-editor.org/rfc/rfc9110#name-501-not-implemented/, 2022.

[12] B. Jabiyev, S. Sprecher, A. Gavazzi, T. Innocenti, K. Onarlioglu, and E. Kirda. FRAMESHIFTER: Security implications of HTTP/2-to-HTTP/1 conversion anomalies. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1061–1075, Boston, MA, Aug. 2022. USENIX Association.

[13] T. Luo, L. Wang, S. Yin, H. Shentu, and H. Zhao. Rbp: a website fingerprinting obfuscation method against intelligent fingerprinting attacks. *Journal of Cloud Computing*, 10(1):29, May 2021.

[14] Maxim Dounin. HTTP/2 Gateway. https://mailman.nginx.org/pipermail/nginx/2015-December/049445.html/, 2015.

[15] Portswigger. Exploiting HTTP Request Smuggling. https://portswigger.net/web-security/request-smuggling/exploiting/, 2023.

[16] wifi-professionals.com. 4-way Handshake. https://www.wifi-professionals.com/2019/01/4-way-handshake/, 2019.

[17] Wireshark. How to Decrypt 802.11. https://wiki.wireshark.org/HowToDecrypt802.11/, 2020.