

Our Fingerprints:

Joel: 7E79009B78AB6FA4233133CDCBE5AFD955854573

u2152233@live.warwick.ac.uk

Niall Noonan: FDE69348FBEC0080B658384D0D7977AF53ABC773

u2147214@live.warwick.ac.uk

Rosie Marlton: A21C 3345 0509 9AD6 4D24 C424 6C7F 48B6 7DFA
F9D9

rosie.marlton@warwick.ac.uk (u2139422)

MODULE TITLE: Implementing Secure Systems

MODULE CODE: WM242-24 (cw2)

TEAM NAME: Papa2

ID NUMBER: 2152233

PARTNER ID NUMBER(S): 2147214, 2139422

YOUR GPG Fingerprint:

7E79009B78AB6FA4233133CDCBE5AFD955854573

WM242: Implementing Secure Systems:

Network Implementation:

As part of the project, we were able to implement a fully functioning network for the needs of the office. Our network consists of six different subnets, these consist of:

→ intconn

- ◆ This subnet has the IP address range 192.168.3.0/24 - we chose to use this as it is a private address range (which is necessary as this is an internal subnet)
- ◆ We chose to use a subnet mask of /24 as this allows for 254 usable IP addresses and so therefore fits the needs of the office with room for expansion
- ◆ Furthermore, this subnet consists of the internal gateway ('int-gw'), wireguard host, strongswan host, internal dns, and two user machines (however this could be easily expanded to fit the needs of the network)

→ dmz

- ◆ This subnet has the IP address range 192.168.4.0/24 - again, a private IP address range was used as this is a 'demilitarised zone'
- ◆ We decided to implement a dmz as it adds an extra layer of security to the organisations internal LAN from untrusted traffic whilst still allowing internal services to access the internet
- ◆ We chose to use a subnet mask of /24 as this allows for 254 usable IP addresses and so therefore fits the needs of the office with room for expansion
- ◆ This subnet consists of the internal gateway, external gateway and apache2 web server

→ extconn

- ◆ This subnet has the IP address range 213.1.133.97/27 - this is because it is the subnet which contains the internet and internet facing services and so therefore needs a public address range
- ◆ Furthermore, the subnet mask /27 was used as it is unlikely that there would be a need for more than 32 usable IP addresses, and so this fits the needs of the network
- ◆ This subnet consists of the internet and the external gateway

→ ispA

- ◆ This subnet has the IP address range 100.100.101.0/24 - this is because a public address range was needed, this subnet holds mobile users connecting to a vpn service and so therefore needs to be able to access public facing services

- ◆ The subnet mask used was a /24 - this is because it is understood that the number of total mobile users is around 200 and therefore, a 254 usable IP addresses for each vpn gateway allows more than adequate room for expansion
- ◆ This subnet consists of mobile users connecting to a Wireguard VPN Gateway

→ ispB

- ◆ This subnet has the IP address range 200.200.200.0/24 - this is because a public address range was needed, this subnet holds mobile users connecting to a vpn service and so therefore needs to be able to access public facing services
- ◆ The subnet mask used was a /24 - this is because it is understood that the number of total mobile users is around 200 and therefore, a 254 usable IP addresses for each vpn gateway allows more than adequate room for expansion
- ◆ This subnet consists of mobile users connecting to a Strongswan VPN Gateway

→ dns

- ◆ This subnet has the IP address range 8.8.8.0/24 - this is because it was necessary for the external DNS services to be on an 8.8.8.8 address as this is the primary DNS server for Google DNS - this is a public address
- ◆ Google DNS is a public DNS service that is provided by Google with the aim to make the Internet and the DNS system faster, safer, secure, and more reliable for all Internet users and is available for anyone to use. (www.whatsmydns.com, n.d)
- ◆ We chose to use a subnet mask of /24 as this allows for 254 usable IP addresses and so therefore fits the needs of the office with room for expansion
- ◆ This subnet consists of external DNS services

DNS implementation:

In order to enclose all of the organisations online assets within the domain `papa2.cyber.test`, both the int-dns and ext-dns were configured, (as seen in the submission) and we resolved the hostnames `www.papa2.cyber.test`, `gw1.papa2.cyber.test`, `gw2.papa2.cyber.test` and `ca.papa2.cyber.test` all at their appropriate addresses, with internet facing addresses on the ext-dns and internal addresses on the int-dns. Evidence of these hostnames being resolved can be seen in 'dns.pcap'. Testing was accomplished by performing a dig on the various domain names.

Firewall rules:

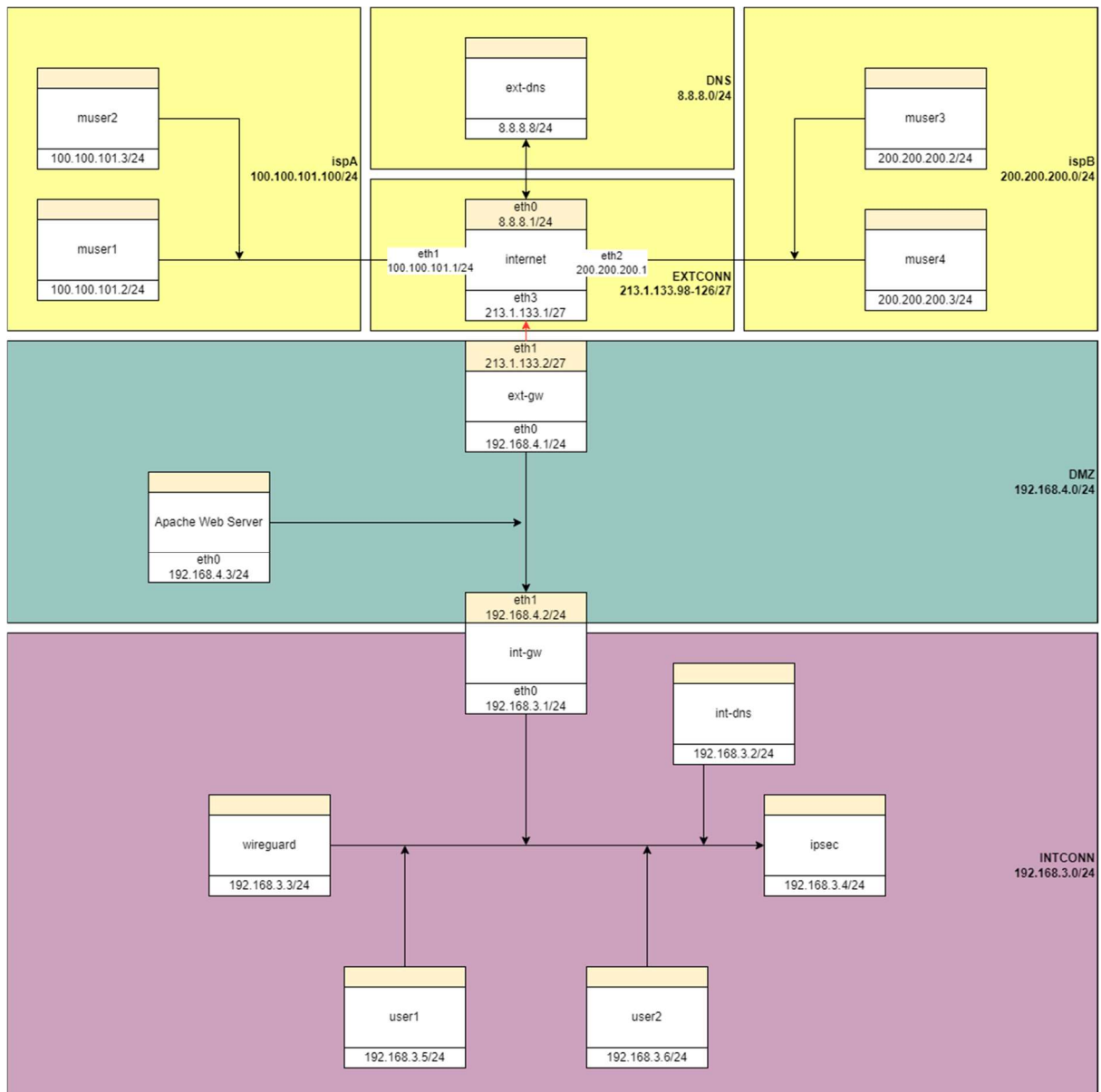
Ensuring connectivity across our network was achieved through the use of firewall rules which allowed us to control the flow of traffic and manage what traffic would and could not access the internal services.

For example, a large part of this was implemented on our external gateway, which was the bridge between the dmz and the internet. As part of this we implemented a series of NAT rules which allowed us to only accept outside traffic trying to access the Internal DNS Wireguard VPN, Strongswan VPN or Apache Web Server - this meant that unauthorised traffic wouldn't have access to our internal machines. For instance, as seen in 'ext-gw.startup', NAT Prerouting rules were used to route traffic trying to access the services listed above via their public IP addresses, and redirect this traffic to the internal, private IP address of this service. This was a significant part of keeping our internal services safe from outsider threats.

As well as this, rules were implemented to allow Wireguard and Strongswan VPN traffic to pass through into the internal network. These rules enabled us to accept both TCP and UDP traffic into the internal network and to the internal VPN machines.

Furthermore, only traffic that was 'RELATED' and 'ESTABLISHED' was accepted through the external gateway and into the dmz and then internal network - preventing unauthorised connections from being allowed.

Network Diagram:



Wireguard:

Wireguard is an 'extremely simple yet fast and modern VPN that utilises **state-of-the-art cryptography**. It aims to be faster, simpler, leaner, and more useful than IPsec'. It uses state-of-the-art cryptography such as the Noise protocol framework, Curve25519, ChaCha20, Poly1305, BLAKE2, SipHash24, HKDF, and secure trusted constructions and is very high-speed - which is perfect for the needs of the organisation. (Donenfield, 2015)

In order to generate the keys for gw1, muser1, muser2 and muser3 the command: '**umask 077; wg genkey | tee privatekey | wg pubkey > publickey**' was used; this allows for a private key to be generated and then a public key generated, derived from the private key.

Once the keys were generated for each client and the server a config file was created following the convention muserX-wg0.conf where X was either 1,2 or 3 and wireguard-wg0.conf for gw1's configuration file. These files were then stored within the /etc/wireguard folder for each machine. The structure of each of these files were as follows:

```
[Interface]
Address = 192.168.1.4/32
PrivateKey = CCaVdS4NyUW+Igc2I9wJsT18LjzDz/PXZJKdv0S9MUI=

[Peer]
PublicKey = SQqHbWZ/b71x2Y3HJQRQp6mZFSPM7ZTKDhSGSF11ATg=
AllowedIPs = 192.168.1.1/24, 192.168.3.0/24
Endpoint = 213.1.133.98:51820
```

Figure 1. Example of a peer config file

The interface section defines what address is being assigned to the client alongside the peer's generated private key, used to establish a connection between the interface and the server, which holds the public key. The peer section then defines the endpoint address and port number of the server alongside its public key and a set of allowed ips, the set of ips the local host should route to the remote host through the wireguard tunnel (stackoverflow, n.d)

When implementing wireguard we suffered from a bug which, when assigning multiple peers multiple subnets under the AllowedIPs statement the server's wg0.conf file would either not run at all or return a message saying the allowed ips for the other two clients were 'none' which was not the case but was caused by wireguard not setting up routing for the other peers due to the subnets overlapping. This was corrected by providing a specific ip under the allowed ips instead of an overlapping subnet e.g. AllowedIPs = 192.168.1.0/24 to: AllowedIPs = 192.168.1.2/32 (GitHub, n.d).

In the packet capture 'wireguard.pcap' from packet number 4 to 7 it is possible to see the wireguard tunnel in effect as it establishes a handshake between muser1 and gw1 (packets no. 4-5) and then proceeds to ping successfully between the two encrypting the icmp packets (packets no. 6-7). This can also be seen for the other two hosts (packet numbers: 8-9) for the handshake initiation for muser2 and (packet numbers: 10-11) for a successful ping between the two hosts. The same applies to muser3 (packet numbers: 14-15) for the successful handshake and (packet numbers: 16-17) for the successful ping between the two hosts i.e. muser3 and gw1.

The second half of the packet capture tests for connectivity over the subnet, in this case user1 on the int-conn subnet, which can be seen in (packet numbers: 51-54) where, in packet 51, muser1 connects to the tunnel and then, in packet 52 sends a ping request to user1 and receives a reply in packet 53, the data is then transported back through the wireguard tunnel on the internet transporting the icmp response. This can also be seen for muser2 (packet numbers: 43-46) and muser3 (packet numbers: 37-42) respectively.

Strongswan:

StrongSwan is a comprehensive implementation of the Internet Key Exchange (IKE) protocols that allows securing IP traffic in policy- and route-based IPsec scenarios from simple to very complex (strongswan, n.d).

Strongswan itself has many security features which as a team we planned to implement - some of these include:

- NAT Traversal (uses UDP to encrypt headers and payloads to maintain authenticity)
- IPComp (which compresses ip datagrams to increase performance)
- MOBIKE (allows a user to maintain VPN connection whilst changing IP addresses associated with IKEv2)
- Integrity Testing Support (used to detect non-malicious file manipulations)

(strongswan, n.d)

Unfortunately, as a group we were not able to achieve a fully robust strongswan realisation of an ipsec gateway due to time constraints - however, we were able to start implementing the VPN gateway (please see /ipsec/etc for configuration files, keys and certificates created). We were able to decide that our best option would be a 'Roadwarrior Case' configuration - this is because it makes use of the newer swanctl, rather than the depreciated ipsec.conf format. Furthermore, this configuration would allow our gateway to serve an arbitrary number of remote VPN clients who would usually have dynamic IP addresses (this in turn would make it easier to implement features such as MOBIKE). This configuration would consist of one 'moon', ie. our gw2, and one 'roadwarrior', ie. a mobile user. Furthermore, this would also allow us to make use of the virtual IP feature of StrongSwan as well as the extended authentication protocol (EAP), which would mean users would have to authenticate themselves using a password, making the VPN more secure, and so therefore protecting our internal services. Finally, if given more time, we would have aimed to further increase the security of this ipsec gateway by making use of Perfect Forward Secrecy (which refers to an encryption system that changes the keys used to encrypt and decrypt information frequently and automatically). (StrongSwan, n.d)

Key Generation & Signing:

As a part of this project, we were required to submit a GPG key that is consistent with our University of Warwick emails and valid up until at least September 2025.

We were able to generate these keys using the following:

```
$ gpg --full-generate-key
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection?
```

As a group, we all opted for RSA keys, with a length of 4096 bits (as this provides a higher level of security) and with a validity of at least 3 years.

Furthermore, as a group, we were all able to have our submitted public keys signed by at least three other students' submitted public keys and also have correctly used the private key associated with our submitted public key to sign the submitted public keys of at least three other students in the class.

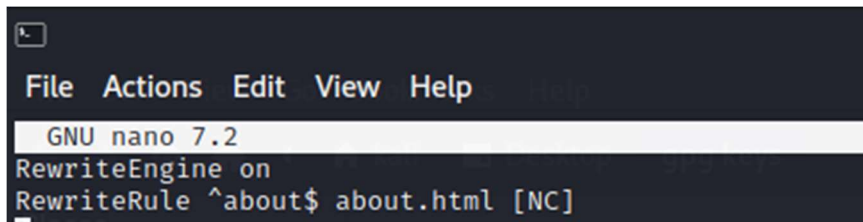
Apache Web Server:

As part of this project, we have implemented a robust HTTPS configuration of an apache2 web server.

The web server itself includes several features to help improve its security. For example, OpenSSL allows us to create HTTPS connections, this is important as HTTPS uses the SSL/TLS protocol to encrypt communications so that attackers can't steal data. This also allows us to confirm that a website server is who it says it is, preventing impersonation. This would be vital for the office webserver to ensure that communications are safe from malicious attackers through encryption. (CloudFlare, n.d)

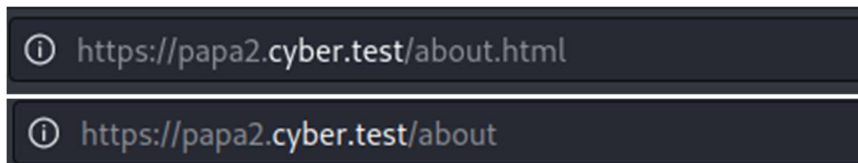
Furthermore, in the security configuration, it was ensured that server tokens were configured with the 'prod' option. This means that only 'Apache' is returned in the server header and therefore prevents the software version, operating system modules installed etc. from being broadcasted. Not broadcasting this information makes our web server less of a target to attackers as there is less available information that can be taken advantage of. In addition to this, server signatures were turned off - this follows on and ensures that less information is publicly available which could be exploited. Server signatures can give away information that can be used by threat actors to exploit a system.

Furthermore, Mod_rewrite was used to enable us to use human-readable URLs - this refers to the idea that if the web server was to have multiple pages, they could be searched for through: www.papa2.cyber.test/about rather than www.papa2.cyber.test/about.html This in itself does not affect the security of the system but is a more clean and professional look for an office - this was enabled through the '.htaccess' file which contains:



```
File  Actions  Edit  View  Help
GNU nano 7.2
RewriteEngine on
RewriteRule ^about$ about.html [NC]
```

In terms of future expansion, this file can be updated to include more pages following this same RewriteRule, as they all follow a similar format. An example of how this may look to users is:



```
https://papa2.cyber.test/about.html
https://papa2.cyber.test/about
```

As you can see, this is a much cleaner look and more in line with what would be expected of an office.

The apache web server itself is located at 213.1.133.100 and uses the domain name 'www.papa2.cyber.test' - this was implemented through the use of dns configurations.

DNSSEC:

DNS Security Extensions strengthen authentication in DNS through the use of digital signatures based on public key cryptography - this refers to the idea that the DNS data itself is signed by the owner of the data. The DNS zone owner uses the zone's private key to sign DNS data and generate digital signatures and the public key can be used to validate the authenticity of DNS data. Furthermore, DNSSEC adds data origin authentication and data integrity protection to the DNS protocol. (ICANN, 2016)

Under the time constraints given, as a group we were not able to implement a robust and compelling DNSSEC configuration. However, given more time, we would have aimed to do this through the following:

- Creating and implementing our DNS zones - this can be done with services such as the Google Cloud DNS Console - alternatively, they could have been implemented through the zone configuration such as:

```
papa2.cyber.test. IN SOA ns1.papa2.cyber.test. u2139422@warwic.ac.uk (
                                2023021401      ; Serial
                                10800             ; Refresh after 3 hours
                                3600              ; Retry after 1 hour
                                604800           ; Expire after 1 week
                                300 )            ; Negative Response TTL
```

(Jaimes, 2012)

- Generating KSK and ZSK keys for our DNS zones, this can be done in terminal through the use of 'dnssec-keygen'

- We would then store these keys in the zone file and sign this zone
 - Reconfiguring the zones and enabling DNSSEC support in the BIND configuration
 - Providing the zone parent with the DS record
- (www.linuxjournal.com, n.d)

X509 Certificate Authority:

Hierarchy and explanation:

We have created a 2-tier hierarchy, with a root CA at the top as the base of the chain of trust, followed by two intermediate CA's which are used to issue certificates to appropriate end-entities. By implementing intermediate CA's the security of the PKI is increased, as "separating certificate authorities in this way will reduce the impact of compromise of a single CA and provide a separation of duty between each CA" (NCSC, 2023).

This also means the root CA can be kept offline, which significantly reduces the chances of the whole chain of trust being compromised. The ICAs are trusted by root and kept online, so able to issue certificates to end entities.

We decided to create a VPN CA and a WWW CA, as these are the only two types of services that will need to be issued certificates. The NCSC states that the "the intermediate CA will often be organised to issue certificates to a certain function", which is what we have done here with VPN and web services. By separating them with their own respective ICA, these two different functionalities can be kept separate from each other, which provides a level of segmentation between different use cases for the PKI. This will help to reduce the 'blast radius' of a compromised CA (NCSC, n.d)

Our original plan was to create an internal ICA and an external ICA, however this makes less sense as here each ICA would have to deal with a mix of functionalities, which goes against the guidelines issued by the NCSC.

The script of instructions:

The files and folders that my script creates are all stored within the CA_stuff folders located within each directory that requires it. There are 3 scripts that are used to create all of the certificates, 'make_CA_certs.sh', 'makin_VPN_certs.sh', 'makin_WWW_certs.sh' and all of these are run sequentially through the script 'initialise_CA.sh'.

We have created multiple scripts to initialise the CA hierarchy within our netkit implementation. First of all, the necessary folders are created for the root CA, where then an RSA key is created and stored in the 'private' folder, which is then set to be read-only, and only readable by the owner. This should help to prevent the private key being stolen and used for malicious purposes like issuing certificates to untrustworthy services, such as what happened with Stuxnet*. (Enterprise Systems, 2023)

Then the root CA's self-signed cert will be generated, using root CA's private key. This certificate will have a 20-year lifespan and will be saved in a folder named `certs` within the `root_CA` directory. We chose this amount of time as it seemed typical for most root CA certs to expire after a period of 10 to 20 years. This seemed a good balance to us, as any longer could pose some security risks, but shorter would only mean the PKI has to be re-implemented sooner.

Next, the script will verify the root certificate to make sure it was created properly, then set the permissions to read only to ensure it cannot be tampered with. It is important that the certificate is able to be read. Now that the root CA has been successfully set up, the script moves on to instantiating the ICAs.

A very similar process is followed for both `WWW_ICA` and `VPN_ICA`, starting with creating all the necessary folders and generating their respective private keys. However, it comes with the extra step of sending a certificate signing request (`csr`) to the root CA. It is also worth noting that the script sets the lifespan of ICA certificates to be only 10 years.

The `csr` is sent to the root CA (copied from the ICA directory into the `unsigned` folder in the root CA directory), and then proceeds to get signed by the root CA. This process comes with the output of a signed certificate for the ICA in question, which is stored in the `'signed'` folder in the root CA directory, and then also copied to the `'certs'` folder within the correct ICA. Once again permissions are set to read only for this certificate. The final step in establishing the CAs is to create a certificate chain, which is outputted to the `'certs'` folder of the ICA in question.

The next scripts that will be run work almost exactly the same, but instead generate keys and certificates for `ipsec` and `apache`, which are then signed by the appropriate ICA, instead of being signed directly by the root CA.

Each entity has their own `openssl.cnf` file, which details how their certificates should be generated. This has helped to achieve consistency with all certificates, as it makes it easy to ensure they are all formatted the same.

We chose RSA-4096 over ed25519 mainly as I was more familiar with how it works, and as RSA is the most widely used algorithm for SSH keys, it is the most supported and has the most compatibility. RSA should also provide slightly more security due to the significantly longer key length, though this does sacrifice some speed/ efficiency, as the larger key requires more time to generate. However, with modern computer systems, it is unlikely there will be any noticeable difference that can be perceived by a user. RSA-4096 is expected to remain unbreakable for the foreseeable future, making it a good option for scalability, as the keys will not need to be changed for a long time. It has also been claimed that RSA is faster than ECC for signature verification (Yubico, 2015).

We chose to use RSA-4096 over RSA-2048, as 2048 keys are only expected to stay secure until 2030. This would not be sufficient, as our root CA has a validity period of 20 years.

The structure of the CA hierarchy also provides good options for scalability. It would be easy to modify the script and add in additional ICAs should new types of services need to be

implemented. It is also a very simple process to get a new end-entity's certificate signed by an ICA.

We have not implemented sub-keys for sake of maintaining some simplicity, however they could be a useful upgrade in the future as they provide an enhancement of security. Sub-keys can be revoked independently of primary keys ([wiki.debian.org, n.d](https://wiki.debian.org/n.d)) meaning that if a sub-key is compromised, a new one can easily be created from the original primary key, without having to create a new primary key. Sub-keys make key management easier, which would become more useful as the scale of an implementation increases.

In the future we would also plan to get wireguard to work with the CA, however unfortunately we were unable to achieve this in the given time frame.

Bibliography:

262588213843476 (n.d.). *How to setup your own CA with OpenSSL*. [online] Gist. Available at: <https://gist.github.com/Soarez/9688998> [Accessed 15 Feb. 2023].

andres.jaimes.net. (2012). *Create a DNS zone file*. [online] Available at: <https://andres.jaimes.net/90/how-to-create-a-dns-zone-file/> [Accessed 15 Feb. 2023].

baeldung (2021). *Creating a Self-Signed Certificate With OpenSSL | Baeldung*. [online] www.baeldung.com. Available at: <https://www.baeldung.com/openssl-self-signed-cert>.

Cloud Flare. (n.d.). *Why use HTTPS?* [online] Available at: <https://www.cloudflare.com/en-gb/learning/ssl/why-use-https/> [Accessed 15 Feb. 2023].

developers.yubico.com. (n.d.). *Generating keys using OpenSSL*. [online] Available at: https://developers.yubico.com/PIV/Guides/Generating_keys_using_OpenSSL.html

Enterprise Systems. (2023). *Best Practices for Securing Private Keys and Code-Signing Certificates -- Enterprise Systems*. [online] Available at: <https://esj.com/articles/2011/06/28/best-practices-securing-private-keys.aspx?m=1> [Accessed 15 Feb. 2023].

GitHub. (n.d.). *Bug? allowed ips: (none) (is configured!) · Issue #80 · WireGuard/wireguard-vyatta-ubnt*. [online] Available at: <https://github.com/WireGuard/wireguard-vyatta-ubnt/issues/80> [Accessed 15 Feb. 2023].

ICANN (2016). *DNSSEC – What Is It and Why Is It Important? - ICANN*. [online] [icann.org](https://www.icann.org). Available at: <https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en>.

SSL.com. (n.d.). *Manually Generate a Certificate Signing Request (CSR) Using OpenSSL*. [online] Available at: <https://www.ssl.com/how-to/manually-generate-a-certificate-signing-request-csr-using-openssl/> [Accessed 15 Feb. 2023].

Stack Overflow. (n.d.). *What is the difference between Endpoint and AllowedIPs fields in Wireguard config file?* [online] Available at: <https://stackoverflow.com/questions/65444747/what-is-the-difference-between-endpoint-and-allowedips-fields-in-wireguard-confi> [Accessed 16 Feb. 2023].

strongSwan project (2012). *strongSwan - IPsec VPN for Linux, Android, FreeBSD, Mac OS X, Windows*. [online] Strongswan.org. Available at: <https://www.strongswan.org/>.

strongSwan project (2012). *strongSwan - Configuration Quickstart*. [online] Strongswan.org. Available at: <https://www.strongswan.org/>.

wiki.debian.org. (n.d.). *Subkeys - Debian Wiki*. [online] Available at: <https://wiki.debian.org/Subkeys> [Accessed 15 Feb. 2023].

www.ncsc.gov.uk. (n.d.). *7. Use a separate intermediate CA per technology or organisation function*. [online] Available at: www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/pki-principles/use-a-separate-intermediate-ca-per-technology [Accessed 15 Feb. 2023].

www.ncsc.gov.uk. (n.d.). *Certificate Authority hierarchy*. [online] Available at: <https://www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/introduction-to-public-key-infrastructure/ca-hierarchy#:~:text=A%20CA%20hierarchy%20starts%20with> [Accessed 15 Feb. 2023].

www.openssl.org. (n.d.). */docs/man1.1.1/man5/config.html*. [online] Available at: <https://www.openssl.org/docs/man1.1.1/man5/config.html> [Accessed 15 Feb. 2023].

www.linuxjournal.com. (n.d.). *DNSSEC Part II: the Implementation | Linux Journal*. [online] Available at: <https://www.linuxjournal.com/content/dnssec-part-ii-implementation> [Accessed 15 Feb. 2023].

www.whatsmydns.net. (n.d.). *8.8.8.8 - DNS Articles - What's My DNS?* [online] Available at: <https://www.whatsmydns.net/articles/8-8-8-8.html#:~:text=8.8%20is%20the%20primary%20DNS> [Accessed 15 Feb. 2023].

Yubico. (2015). *The Big Debate, 2048 vs. 4096, Yubico's Position*. [online] Available at: <https://www.yubico.com/blog/big-debate-2048-4096-yubicos-stand/> [Accessed 15 Feb. 2023].