

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Phase One</b>	<b>2</b>
<b>Network Topology - u2136685</b>	<b>2</b>
<b>Further work</b>	<b>4</b>
<b>WireGuard - u2136685</b>	<b>5</b>
<b>Configuration files</b>	<b>5</b>
<b>Two sample mobile workers</b>	<b>6</b>
<b>Further work</b>	<b>7</b>
<b>Certificate Authority - u2118859</b>	<b>7</b>
<b>Further work</b>	<b>11</b>
<b>Domain names and allocated IPs - u2136685, u2101239</b>	<b>11</b>
<b>Further work</b>	<b>13</b>
<b>Phase Two</b>	<b>14</b>
<b>Apache - u2136685</b>	<b>14</b>
<b>HTTPS</b>	<b>14</b>
<b>Further work</b>	<b>16</b>
<b>StrongSwan - u2101239</b>	<b>16</b>
<b>WireGuard - u2136685</b>	<b>17</b>
<b>Several sample mobile workers</b>	<b>17</b>
<b>Further work</b>	<b>18</b>
<b>Phase Three</b>	<b>19</b>
<b>DNSSEC - u2101239</b>	<b>19</b>
<b>Further Work</b>	<b>19</b>
<b>Evaluation of VPN Configurations</b>	<b>20</b>
<b>References</b>	<b>21</b>

# Phase One

## Network Topology - u2136685

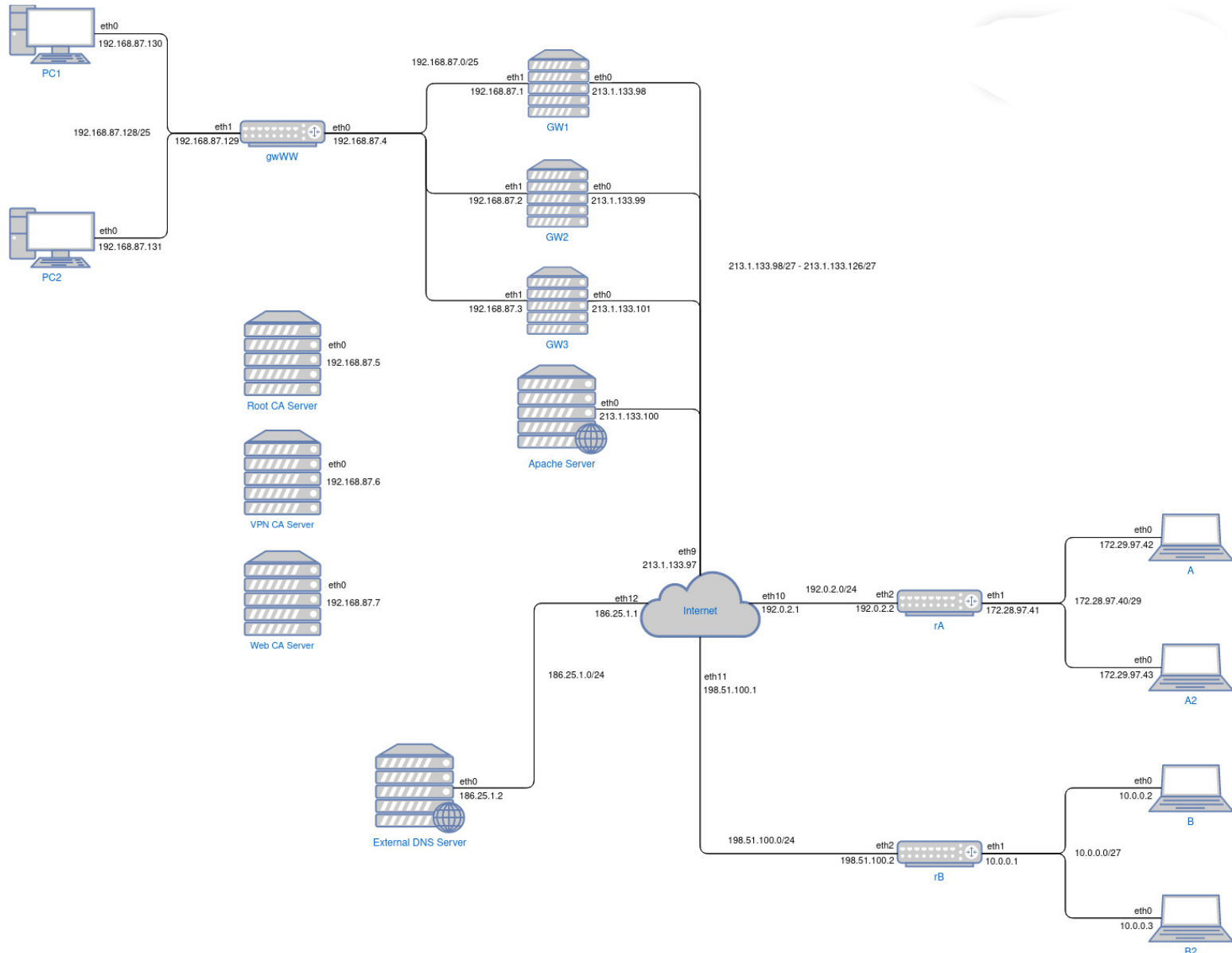


Figure 1 - Network Diagram

As seen in Figure 1, in the bottom right, there are two locations where workers could connect from, LAN A and LAN B. Both of the LANs have routers with an internet-facing ethernet card and a private ethernet card towards the workers.

To replicate all-purpose access points, such as a hotel's or any public network, the routers, rA and rB, have limited NATing, masquerading all of the network traffic incoming from the workers and towards the internet, as seen in Figure 2 and Figure 3 respectively.

```
#####
#
# rA.startup
#
#####

ip link set dev eth1 address 02:aa:aa:11:11:11
ip link set dev eth2 address 02:aa:aa:22:22:22

ip addr add 172.28.97.41/29 dev eth1
ip link set up dev eth1
ip addr add 192.0.2.2/24 dev eth2
ip link set up dev eth2

ip route add default via 192.0.2.1

iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Figure 2 - LAN A router NAT rule

```
#####
#
# rB.startup
#
#####

ip link set dev eth1 address 02:bb:bb:11:11:11
ip link set dev eth2 address 02:bb:bb:22:22:22

ip addr add 10.0.0.1/27 dev eth1
ip link set up dev eth1
ip addr add 198.51.100.2/24 dev eth2
ip link set up dev eth2

ip route add default via 198.51.100.1

iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Figure 3 - LAN B router NAT rule

The Warwick LAN consists of multiple internet-facing gateways, gw1, gw2 and gw3. Gw3 serves as the default gateway for normal traffic in and out of the LAN. Gw1 and gw2 serve as the LAN's VPN gateways, using WireGuard and an IPsec realisation of StrongSwan, respectively. Warwick's 192.168.87.0/24 subnet was split into two, 192.168.87.0/25 and 192.168.87.128/25, to facilitate the separation of the LAN's machines that have an internet-facing ethernet card, such as the internet-facing gateways, and the LAN's private machines, such as PC1 and PC2. The gateway gwWW was used to connect the private machines to the internet-facing gateways.

As stated above, gw3 is the default route for normal traffic. As seen in Figure 4 and 5, pinging the Apache web server placed on the internet from the local machine, pc1, proves that the traffic is in fact routed through gw3.

```
root@pc1:~# ping -c4 www.kilo2.cyber.test
PING www.kilo2.cyber.test (213.1.133.100) 56(84) bytes of data:
64 bytes from www.kilo2.cyber.test (213.1.133.100): icmp_seq=1 ttl=62 time=0.726
ms
64 bytes from www.kilo2.cyber.test (213.1.133.100): icmp_seq=2 ttl=62 time=0.886
ms
64 bytes from www.kilo2.cyber.test (213.1.133.100): icmp_seq=3 ttl=62 time=0.798
ms
64 bytes from www.kilo2.cyber.test (213.1.133.100): icmp_seq=4 ttl=62 time=0.848
ms

--- www.kilo2.cyber.test ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.726/0.814/0.886/0.059 ms
```

Figure 4 - Pinging the web server from the local machine, pc1

9	2023-02-15	21:20:14...	213.1.133.101	213.1.133.100	ICMP	98 Echo (ping) request	id=0x50a6, seq=1/256, ttl=62 (reply in 10)
10	2023-02-15	21:20:14...	213.1.133.100	213.1.133.101	ICMP	98 Echo (ping) reply	id=0x50a6, seq=1/256, ttl=64 (request in 9)
13	2023-02-15	21:20:15...	213.1.133.101	213.1.133.100	ICMP	98 Echo (ping) request	id=0x50a6, seq=2/512, ttl=62 (reply in 14)
14	2023-02-15	21:20:15...	213.1.133.100	213.1.133.101	ICMP	98 Echo (ping) reply	id=0x50a6, seq=2/512, ttl=64 (request in 13)
17	2023-02-15	21:20:16...	213.1.133.101	213.1.133.100	ICMP	98 Echo (ping) request	id=0x50a6, seq=3/768, ttl=62 (reply in 18)
18	2023-02-15	21:20:16...	213.1.133.100	213.1.133.101	ICMP	98 Echo (ping) reply	id=0x50a6, seq=3/768, ttl=64 (request in 17)
21	2023-02-15	21:20:17...	213.1.133.101	213.1.133.100	ICMP	98 Echo (ping) request	id=0x50a6, seq=4/1024, ttl=62 (reply in 22)
22	2023-02-15	21:20:17...	213.1.133.100	213.1.133.101	ICMP	98 Echo (ping) reply	id=0x50a6, seq=4/1024, ttl=64 (request in 21)

Figure 5 - PCAP file of gw3's internet-facing card, eth0

On-the-other-hand, when pinging a WireGuard machine from pc1, it is re-routed through gw2, as seen in Figure 6.

```

root@pc1:~# traceroute 10.90.90.3
traceroute to 10.90.90.3 (10.90.90.3), 30 hops max, 60 byte packets
 1 192.168.87.129 (192.168.87.129) 0.220 ms 0.112 ms 0.129 ms
 2 192.168.87.3 (192.168.87.3) 0.419 ms 0.359 ms 0.290 ms
 3 192.168.87.1 (192.168.87.1) 0.352 ms 0.285 ms 0.275 ms
 4 10.90.90.3 (10.90.90.3) 0.930 ms 1.266 ms 0.890 ms

```

Figure 6 - Traceroute of WireGuard worker showing redirection from .3 to .1, gw3 to gw1

In Figure 4, we show that the pc2 machine can ping the domain name of the Apache machine. This is possible through the external DNS machine on the internet, seen in Figure 1. In figure 7, Gw3 is querying the DNS server for the IP associated with the domain name, as a result of the pinging above.

1	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	80 Standard query 0x1716 A www.kilo2.cyber.test
2	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	80 Standard query 0xa0a0 AAAA www.kilo2.cyber.test
3	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	96 Standard query response 0x1716 A www.kilo2.cyber.test A 213.1.133.100
4	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	80 Standard query response 0xa0a0 Refused AAAA www.kilo2.cyber.test
7	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	86 Standard query 0x5c05 PTR 100.133.1.213.in-addr.arpa
8	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	120 Standard query response 0x5c05 PTR 100.133.1.213.in-addr.arpa PTR www.kilo2.cyber.test
11	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	86 Standard query 0x6f37 PTR 100.133.1.213.in-addr.arpa
12	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	120 Standard query response 0x6f37 PTR 100.133.1.213.in-addr.arpa PTR www.kilo2.cyber.test
15	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	86 Standard query 0x51e1 PTR 100.133.1.213.in-addr.arpa
16	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	120 Standard query response 0x51e1 PTR 100.133.1.213.in-addr.arpa PTR www.kilo2.cyber.test
19	2023-02-15	...	213.1.133.101	186.25.1.2	DNS	86 Standard query 0x79ee PTR 100.133.1.213.in-addr.arpa
20	2023-02-15	...	186.25.1.2	213.1.133.101	DNS	120 Standard query response 0x79ee PTR 100.133.1.213.in-addr.arpa PTR www.kilo2.cyber.test

Figure 7 - DNS queries from gw3 to the extDNS machine

The certificate authority machines sit on the Warwick LAN, but the root CA is offline and disconnected, as per the NCSC guidelines. The web and VPN CAs are up as they can be revoked and restored, if compromised, by bringing the root CA online. (National Cyber Security Centre, 2020b)

## Further work

In the future, we would bring the Apache web server into the Warwick LAN and implement NAT rules on the firewalls for port 443, as our web server uses HTTPS. Moreover, on the Apache machine, we would drop all packets and only allow the ones we deem necessary for the Apache machine to function correctly.

Moreover, we would configure an internal DNS as well, for the Warwick LAN, using the external DNS as an upstream DNS only if there are no records of the domain name in the internal one. The internal DNS would be more tightly secured with firewall rules and it would be less prone to DNS poisoning.

# WireGuard - u2136685

## Configuration files

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/%i.key
Address = 10.90.90.1/24
ListenPort = 51000

[Peer]
#machine a
PublicKey = V1whrReEUoFxxAcSIDOTrhB5xMGVRhltHrsvrUIFvDw=
AllowedIPs = 10.90.90.3/32

[Peer]
#machine a2
PublicKey = cF081iJ4XLSGFJj+tsc5eikmsgf4N0cqBM67I6zdtlk=
AllowedIPs = 10.90.90.4/32

[Peer]
#machine b
PublicKey = AfBM62fnhtRPCQcFLSqYHFScyumPKLF5SE40eRMxk0w=
AllowedIPs = 10.90.90.5/32

[Peer]
#machine b
PublicKey = YQOGTboXbd3tNbp3WZ49NvOjNy5Z3DilavDGHA/q82o=
AllowedIPs = 10.90.90.6/32
```

Figure 8 - GW1 WireGuard configuration file

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
ListenPort = 51000
Address = 10.90.90.5/24

[Peer]
PublicKey = Y1tCQHebAlpYo5xZz0PtE+m1Rd9j4tl0g044KcBnshI=
Endpoint = 213.1.133.98:51000
AllowedIPs = 10.90.90.0/24,192.168.87.0/25,192.168.87.128/25
```

Figure 9 - B WireGuard configuration file

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
ListenPort = 51000
Address = 10.90.90.6/24

[Peer]
PublicKey = Y1tCQHebAlpYo5xZz0PtE+m1Rd9j4tl0g044KcBnshI=
Endpoint = 213.1.133.98:51000
AllowedIPs = 10.90.90.0/24,192.168.87.0/25,192.168.87.128/25
```

Figure 10 - B2 WireGuard configuration file

In Figure 8, the gw1 WireGuard config file sets all of the peers allowed to connect to the gateway, using their public keys and IPs. The IPs are set with the /32 subnet to only allow one IP at a time. The gateway config file also has the information of LAN A, used to complete the phase two requirements.

All of the configuration set the “AllowedIPs” section to the WireGuard LAN, 10.90.90.0/24 and the entirety of the Warwick LAN, to ensure connectivity between a remote worker and the private LAN.

Moreover, the “PostUp” section sets the private key from a file, therefore preventing leakage of the private keys, as usually the private key is set in the “PrivateKey” section. (Ubuntu, 2022)

## Two sample mobile workers

An outside worker machine, b2, can connect through the WireGuard gateway, gw1, and access any local Warwick machine. For example, as seen in Figure 9, a worker can successfully ping a Warwick local machine and the traffic is encrypted when coming out of the gw1 gateway, as seen in Figure 8 showing a PCAP file from gw1’s internet-facing ethernet card, eth0.

1	2023-02-15	20:17...	c6:3e:6a:54:bf:a3	Broadcast	ARP	42 Who has 213.1.133.98? Tell 213.1.133.97
2	2023-02-15	20:17...	02:b2:02:b2:01:b2	c6:3e:6a:54:bf:a3	ARP	42 213.1.133.98 is at 02:b2:02:b2:01:b2
3	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	190 Handshake Initiation, sender=0xAEA4E4F6
4	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	134 Handshake Response, sender=0x4B3510C6, receiver=0xAEA4E4F6
5	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=0, datalen=96
6	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=0, datalen=96
7	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=1, datalen=96
8	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=1, datalen=96
9	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=2, datalen=96
10	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=2, datalen=96
11	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=3, datalen=96
12	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=3, datalen=96
13	2023-02-15	20:17...	02:b2:02:b2:01:b2	c6:3e:6a:54:bf:a3	ARP	42 Who has 213.1.133.97? Tell 213.1.133.98
14	2023-02-15	20:17...	c6:3e:6a:54:bf:a3	02:b2:02:b2:01:b2	ARP	42 213.1.133.97 is at c6:3e:6a:54:bf:a3
15	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=4, datalen=96
16	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=4, datalen=96
17	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0x4B3510C6, counter=5, datalen=96
18	2023-02-15	20:17...	213.1.133.98	198.51.100.2	WireGuard	170 Transport Data, receiver=0xAEA4E4F6, counter=5, datalen=96
19	2023-02-15	20:17...	198.51.100.2	213.1.133.98	WireGuard	74 Keepalive, receiver=0x4B3510C6, counter=6

Figure 8 - GW1 packet capture on eth0

```

root@b2:~# ping 192.168.87.129
PING 192.168.87.129 (192.168.87.129) 56(84) bytes of data:
64 bytes from 192.168.87.129: icmp_seq=1 ttl=62 time=3.55 ms
64 bytes from 192.168.87.129: icmp_seq=2 ttl=62 time=1.38 ms
64 bytes from 192.168.87.129: icmp_seq=3 ttl=62 time=1.37 ms
^C
--- 192.168.87.129 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.370/2.100/3.552/1.026 ms
root@b2:~# ping 192.168.87.130
PING 192.168.87.130 (192.168.87.130) 56(84) bytes of data:
64 bytes from 192.168.87.130: icmp_seq=1 ttl=61 time=1.49 ms
64 bytes from 192.168.87.130: icmp_seq=2 ttl=61 time=1.55 ms
64 bytes from 192.168.87.130: icmp_seq=3 ttl=61 time=1.67 ms
^C
--- 192.168.87.130 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.491/1.570/1.668/0.073 ms
root@b2:~#

```

Figure 9 - Worker, b2, pinging a Warwick local machine through WireGuard

Another worker from the same LAN, b, can also connect to the WireGuard gateway in the same way b2 can. B can ping any machine on the Warwick LAN, such as pc2, as seen in Figure 10.

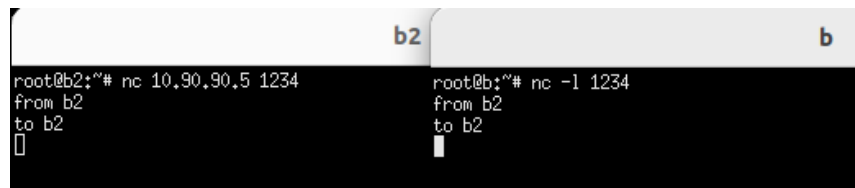
```

root@b:~# ping 192.168.87.131
PING 192.168.87.131 (192.168.87.131) 56(84) bytes of data:
64 bytes from 192.168.87.131: icmp_seq=1 ttl=61 time=4.00 ms
64 bytes from 192.168.87.131: icmp_seq=2 ttl=61 time=1.69 ms
64 bytes from 192.168.87.131: icmp_seq=3 ttl=61 time=1.58 ms
64 bytes from 192.168.87.131: icmp_seq=4 ttl=61 time=1.69 ms
^C
--- 192.168.87.131 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.582/2.237/3.998/1.017 ms

```

Figure 10 - Another worker, b, pinging a Warwick local machine through WireGuard

The Warwick LAN machines can also talk to the WireGuard machines, as seen in Figure 6, and the WireGuard machines can talk to each other, as seen in figure 11.



```
root@b2:~# nc 10.90.90.5 1234
from b2
to b2
[ ]

root@b:~# nc -l 1234
from b2
to b2
[ ]
```

Figure 11 - B2 connecting to a Netcat listener on b through WireGuard

## Further work

We would implement the PreSharedKey for the connection, as an extra layer of cryptographic security. It would add another layer of symmetric encryption to the WireGuard tunnel. (Ubuntu, 2022)

## Certificate Authority - u2118859

The root CA machine's job is to generate the root CA for the Warwick LAN. The root CA is a critical part of the public key infrastructure (PKI) and is responsible for issuing and managing digital certificates for all subordinate CAs and end entities within a trust domain.

```
# generate the CA key Using curve25519
#cd /etc/ipsec.d/
# generate root CA's private key
# openssl genpkey -algorithm ED25519 -out private/rootCA_Key.pem
#chmod 600 private/rootCA_Key.pem
# generate root CA's self-signed cert, related to the private key
#openssl req -key private/rootCA_Key.pem -new -x509 -days 3650 -out cacerts/rootCA_Cert.pem -subj "/C=UK/O=University of Warwick/OU=Cyber Security Centre (Teaching)/CN=Root CA"
# verify it looks right
#openssl x509 -in cacerts/CA_Cert.pem -text -noout
```

Figure 12 - Script to generate root CA on the root CA server

The reason why we chose elliptic curve 25519 over another encryption system such as rsa 2048 or 4096 is that Curve25519 is significantly faster than RSA 2048 (Muchtadi-Alamsyah and Tama, 2019). It is important for applications where performance is critical such as generating certificates. Another reason is key size. Curve25519 uses smaller key sizes than RSA 2048 while providing similar levels of security. This means that Curve25519 keys are faster to generate and use less storage space on the server. RSA 2048 is vulnerable to attacks by quantum computers, which do not exist yet but may be developed in the future. In contrast, Curve25519 is resistant to quantum computing attacks, making it a better alternative for future proofing.

```

csc@csc:~/Documents/cacerts$ openssl x509 -in rootCA_Cert.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      1c:aa:9f:b2:65:6d:f3:84:12:e3:4d:22:a0:d3:b2:ea:be:c0:c2:ce
    Signature Algorithm: ED25519
    Issuer: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Root CA
    Validity
      Not Before: Feb 13 21:54:35 2023 GMT
      Not After : Feb 10 21:54:35 2033 GMT
    Subject: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Root CA
    Subject Public Key Info:
      Public Key Algorithm: ED25519
      ED25519 Public-Key:
        pub:
          07:e2:63:be:3a:aa:9e:cc:c4:0a:23:43:02:eb:6f:
          e7:14:cc:02:e7:6d:6e:8c:4c:38:4f:86:e3:b0:60:
          99:01
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        D7:E3:67:7C:3F:F8:44:D8:0C:8F:3E:05:E0:42:3B:81:CE:14:A9:A9
      X509v3 Authority Key Identifier:
        keyid:D7:E3:67:7C:3F:F8:44:D8:0C:8F:3E:05:E0:42:3B:81:CE:14:A9:A9

      X509v3 Basic Constraints: critical
        CA:TRUE
    Signature Algorithm: ED25519
      fc:ec:52:0b:f3:22:62:fa:ea:ed:cc:93:a2:a5:3f:01:65:45:
      86:e6:79:48:29:3b:f1:c5:bd:de:20:c3:52:d9:b7:54:94:9f:
      01:c3:e7:67:3b:0a:93:84:ac:4b:95:f1:ea:c1:9f:b7:5d:1c:
      0f:69:16:79:64:97:ad:b2:e0:01
csc@csc:~/Documents/cacerts$

```

Figure 13 - Generated root CA

Two intermediate CA servers have been added to the Warwick LAN called vpnCA and webCA. This was chosen for multiple reasons. Reason one being Intermediate CAs allow organisations to create a hierarchical trust model for their digital certificates. This means that we can establish a root CA as the ultimate trust anchor, with intermediate CAs below it. The root CA can delegate trust to intermediate CAs, which can in turn issue certificates to end entities like the vpn servers and apache servers. This delegation of trust allows for a more flexible and scalable system for managing certificates. The second reason being by using an intermediate CA, we can separate the responsibilities of key management and certificate issuance. This means that the private keys used to sign digital certificates can be kept in a more secure location, while the intermediate CA can be used to issue certificates as needed. This can help to reduce the risk of compromise to the private keys. The third reason being intermediate CAs can be used to improve the management of certificate revocation. By using intermediate CAs, we can revoke a specific intermediate CA without revoking the entire hierarchy of certificates. This can help to reduce the impact of a security incident, and simplify the process of managing revocations (NCSC, 2020a).

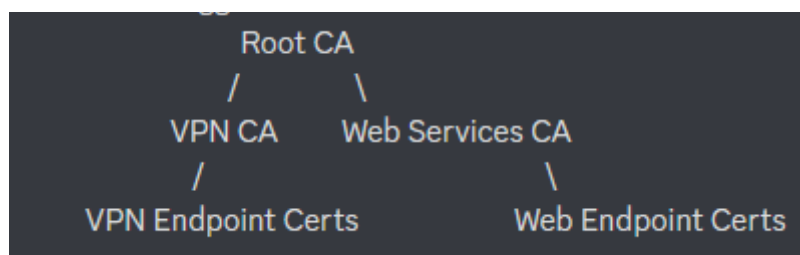


Figure 14 - Diagram of proposed x509 hierarchy



```
# cd /etc/ipsec.d/

# Generate a private key for the vpn CA server
# openssl genkey -algorithm ED25519 -out private/vpn_CAKey.pem

# Create a certificate signing request (CSR) for the vpn CA server
# openssl req -key private/vpn_CAKey.pem -new -out reqs/vpn_csr.pem -subj "/C=UK/O=University of Warwick/OU=Cyber Security Centre (Teaching)/CN=VPN CA"

# openssl x509 -req -in reqs/vpn_csr.pem -extfile ca.conf -extensions v3_ca -CA cacerts/rootCA_Cert.pem -CAkey private/rootCA_Key.pem -CAcreateserial -out cacerts/vpn_Cert.pem
```

Figure 15 - x509 script to generate and sign vpnCA certificate

```
cd /etc/ipsec.d/

# Generate a private key for the web CA server
# openssl genkey -algorithm ED25519 -out private/web_CAKey.pem

# Create a certificate signing request (CSR) for the web CA server
# openssl req -key private/web_CAKey.pem -new -out reqs/web_csr.pem -subj "/C=UK/O=University of Warwick/OU=Cyber Security Centre (Teaching)/CN=Web CA"

# openssl x509 -req -in reqs/web_csr.pem -extfile ca.conf -extensions v3_ca -CA cacerts/rootCA_Cert.pem -CAkey private/rootCA_Key.pem -CAcreateserial -out cacerts/web_Cert.pem
```

Figure 16 - x509 script to generate and sign webCA certificate

```
csc@csc:~/Downloads/ISS-CW2/ipsec/vpnca/etc/ipsec.d/cacerts$ openssl x509 -in vpn_Cert.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      19:ad:0e:1e:22:51:10:45:91:01:93:14:40:07:a7:7b:4b:c2:a7:7e
    Signature Algorithm: ED25519
    Issuer: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Root CA
    Validity
      Not Before: Feb 14 19:09:03 2023 GMT
      Not After : Mar 16 19:09:03 2023 GMT
    Subject: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = VPN CA
    Subject Public Key Info:
      Public Key Algorithm: ED25519
      ED25519 Public-Key:
      pub:
        0d:6b:7f:aa:27:c0:5c:dc:e3:3e:7a:d5:ed:41:72:
        66:8f:be:c8:7e:ba:59:92:a0:0e:7c:11:71:94:82:
        92:6c
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:TRUE
    Signature Algorithm: ED25519
      fb:fc:7f:76:c1:14:6e:44:6a:45:70:0a:28:f1:b1:1d:4c:b2:
      96:c3:27:ac:ab:3c:d5:e8:84:c4:73:0c:84:6c:e6:98:0d:3f:
      ef:af:e3:1e:74:6a:eb:0f:dd:21:6f:0a:15:f0:08:e1:4b:c9:
      2b:3b:95:97:23:76:aa:10:08:07
```

Figure 17 - Generated intermediate vpn CA

```
csc@csc:~/Downloads/ISS-CW2/ipsec/webca/etc/ipsec.d/cacerts$ openssl x509 -in web_Cert.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      39:d3:b5:5a:45:1b:7f:02:69:c8:b2:5e:34:a3:21:76:6c:90:d5:58
    Signature Algorithm: ED25519
    Issuer: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Root CA
    Validity
      Not Before: Feb 14 19:46:23 2023 GMT
      Not After : Mar 16 19:46:23 2023 GMT
    Subject: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Web CA
    Subject Public Key Info:
      Public Key Algorithm: ED25519
      ED25519 Public-Key:
      pub:
        f4:dc:10:95:26:c0:05:3b:e7:2b:bc:ca:d3:da:da:
        46:5e:fd:b7:d5:b6:91:92:6a:34:98:ee:bb:be:9c:
        df:27
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:TRUE
    Signature Algorithm: ED25519
      93:8b:2b:4f:11:13:cf:0b:4b:ae:f9:40:76:c7:ef:66:7c:2b:
      94:3d:0a:e5:5f:6b:4b:1b:e0:6e:96:ba:5e:ca:f6:b8:9c:f1:
      ea:0b:f6:29:a6:65:0a:8a:33:e2:b9:12:d6:62:1a:38:3c:93:
      c6:82:f5:dd:21:31:d8:01:82:08
```

Figure 18 - Generated intermediate web CA

The StrongSwan vpn and web server generates its own certificate client side based on the intermediate certificate it has been given.

```
# generate the private key for gw2
# openssl genpkey -algorithm ED25519 -out private/vpn_gw2_CAKey.pem
# chmod 600 private/vpn_gw2_CAKey.pem

# generate the related cert signing request
# openssl req -key private/vpn_gw2_CAKey.pem -new -out reqs/vpn_gw2_csr.pem -subj "/C=UK/O=University of Warwick/OU=Cyber Security Centre (Teaching)/CN=Strongswan CA"

#####
# then get it signed / issued as a certificate by the CA
# note the common name CN or the subject alternate name (--san)
# MUST match the public domain name or ip addr of the server
# openssl x509 -req -in reqs/vpn_gw2_csr.pem -CA cacerts/vpn_cert.pem -CAkey private/vpn_CAKey.pem -CAcreateserial -days 1825 -out certs/vpn_gw2_Cert.pem -extfile <(echo -e "subjectAltName=IP:192.168.87.2")
```

Figure 19 - x509 script for CA generation for the strongswan vpn

```
# generate the private key for apache server
# openssl genpkey -algorithm ED25519 -out private/www.kilo2.cyber.test.pem
# chmod 600 private/www.kilo2.cyber.test.pem

# generate the related cert signing request
# openssl req -key private/www.kilo2.cyber.test.pem -new -out reqs/www.kilo2.cyber.test.pem -subj "/C=UK/O=University of Warwick/OU=Cyber Security Centre (Teaching)/CN=www.kilo2.cyber.test"

#####
# then get it signed / issued as a certificate by the CA
# note the common name CN or the subject alternate name (--san)
# MUST match the public domain name or ip addr of the server
# openssl x509 -req -extensions SAN -extfile <(printf "\n[SAN]\nsubjectAltName=IP:213.1.133.100") -in reqs/www.kilo2.cyber.test.pem -CA /etc/ipsec.d/cacerts/web_Cert.pem -CAkey /etc/ipsec.d/private/web_CAKey.pem -CAcreateserial -days 1825 -out certs/www.kilo2.cyber.test.pem

#cp private/www.kilo2.cyber.test.pem /hostlab/apache/etc/ipsec.d/private/
#cp certs/www.kilo2.cyber.test.pem /hostlab/apache/etc/ipsec.d/certs/
```

Figure 20 - x509 script for CA generation for the apache server

```
csc@csc:~/Downloads/ISS-CW2/ipsec/gw2/etc/ipsec.d/certs$ openssl x509 -in vpn_gw2_Cert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0b:f6:07:53:bd:d5:fb:20:d1:1a:61:78:95:44:31:3e:45:d3:43:3d
        Signature Algorithm: ED25519
        Issuer: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = VPN CA
        Validity
            Not Before: Feb 14 19:13:56 2023 GMT
            Not After : Feb 13 19:13:56 2028 GMT
        Subject: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Strongswan CA
        Subject Public Key Info:
            Public Key Algorithm: ED25519
            ED25519 Public-Key:
                pub:
                    44:db:e0:d5:e6:98:82:0a:cc:3b:4e:d6:4e:83:73:
                    8f:0c:1d:5b:b0:92:1a:9b:d5:29:fa:9d:36:83:40:
                    8d:ed
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                IP Address:192.168.87.2
        Signature Algorithm: ED25519
        7b:aa:b1:a8:dc:47:82:07:1e:42:8a:45:83:ae:df:b5:4f:5f:
        50:95:93:72:56:bb:fa:97:38:cd:4e:77:57:97:a7:1a:16:77:
        93:b4:84:46:5b:a5:80:4e:45:d0:94:a9:3b:a6:ad:6e:6c:40:
        f8:a9:b8:8d:bc:e6:c1:ec:30:06
```

Figure 21 - Generated StrongSwan CA

```

csc@csc:~/Downloads/ISS-CW2/ipsec/apache/etc/ipsec.d/certs$ openssl x509 -in www.kilo2.cyber.test.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      7f:88:6b:6d:72:81:2e:38:8b:d0:3c:86:ba:7f:f7:01:fc:00:54:71
    Signature Algorithm: ED25519
    Issuer: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = Web CA
    Validity
      Not Before: Feb 14 22:06:47 2023 GMT
      Not After : Feb 13 22:06:47 2028 GMT
    Subject: C = UK, O = University of Warwick, OU = Cyber Security Centre (Teaching), CN = www.kilo2.cyber.test
    Subject Public Key Info:
      Public Key Algorithm: ED25519
      ED25519 Public-Key:
        pub:
          86:1d:da:94:30:1a:a6:32:5d:bb:79:ec:83:db:51:
          3a:e2:1d:81:9e:8c:73:9a:13:ea:86:3e:47:1c:eb:
          5b:44
    X509v3 extensions:
      X509v3 Subject Alternative Name:
        IP Address:213.1.133.100
    Signature Algorithm: ED25519
    09:68:47:69:dd:a1:be:3e:b7:93:58:d6:28:9e:2a:6f:25:64:
    b0:9e:18:d6:a0:c2:f5:45:1c:4d:a3:c6:b2:09:a5:42:12:8e:
    ee:ed:63:b4:96:5b:8d:ac:00:9a:69:cd:17:72:70:0c:99:d8:
    0c:97:76:31:4d:c4:fa:46:ba:05

```

Figure 22 - Generated apache CA

## Further work

In the future we would like to implement a system that would auto distribute the root CA to the intermediate servers through encrypted methods such as FTPS or SFTP instead of having the Root CA offline.

## Domain names and allocated IPs - u2136685, u2101239

The coursework brief specified that we set the following domain names to the following IPs:

1. gw1.kilo2.cyber.test at 213.1.133.98:

```

### gw1 Startup File ###

ip link set dev eth0 address 02:b2:02:b2:01:b2
ip addr add 213.1.133.98/27 dev eth0
ip link set up dev eth0

```

Figure 23 - Gw1 being assigned its appropriate IP

2. gw2.kilo2.cyber.test at 213.1.133.99:

```

### gw2 Startup File ###

ip link set dev eth0 address 02:a2:02:a2:02:a2
ip addr add 213.1.133.99/27 dev eth0
ip link set up dev eth0

```

Figure 24 - Gw2 being assigned its appropriate IP

3. www.kilo2.cyber.test at 213.1.133.100:

```
### apache Startup File ###

ip link set dev eth0 address 0e:01:ee:01:ee:01
ip addr add 213.1.133.100/27 dev eth0
ip link set up dev eth0
```

Figure 25 - Apache being assigned its appropriate IP

Their domain names are set in the “db.cyber.test” file on the external DNS machine, as seen in Figure 26. This file is used by BIND9 to create an authoritative DNS server. We have added the other internet-facing machines as well, the external DNS, rA, rB and gw3.

```
;
; BIND data file for local loopback interface
;
$TTL 604800
@      IN      SOA      dns.cyber.test. admin.cyber.test. (
                                3      ; Serial
                                604800 ; Refresh
                                86400  ; Retry
                                2419200 ; Expire
                                604800 ) ; Negative Cache TTL
;

      IN      NS       dns.cyber.test.

dns.cyber.test. IN      A       186.25.1.2
rA.cyber.test.  IN      A       192.0.2.2
rB.cyber.test.  IN      A       198.51.100.2
gw1.kilo2.cyber.test. IN      A       213.1.133.98
gw2.kilo2.cyber.test. IN      A       213.1.133.99
gw3.kilo2.cyber.test. IN      A       213.1.133.101
www.kilo2.cyber.test. IN      A       213.1.133.100
```

Figure 26 - “db.cyber.test” file with A records for public-facing services

The “named.conf.local” file was also modified to create a new zone, *cyber.test*, comprising the above records (see Figure XXX below). Notice the final two lines, which are included to facilitate the use of DNSSEC, configured in phase three. Setting “auto-dnssec” to “maintain” causes new signatures to be created for the zone when they expire, and specifying “inline-signing” allows BIND9 to create these signatures transparently (without manual user input).

```
zone "cyber.test" {
    type master;
    file "/etc/bind/zones/db.cyber.test";
    auto-dnssec maintain;
    inline-signing yes;
};
```

Figure XXX - Definition of the *cyber.test* zone

To verify that these domain names have been successfully assigned, we ran “nslookup” for each of them:

**1. gw1.kilo2.cyber.test at 213.1.133.98:**

```
root@pc1:~# nslookup gw1.kilo2.cyber.test
Server:      186.25.1.2
Address:     186.25.1.2#53

Name:   gw1.kilo2.cyber.test
Address: 213.1.133.98
```

Figure 27 - Nslookup for gw1.kilo2.cyber.test

**2. gw2.kilo2.cyber.test at 213.1.133.99:**

```
root@b:~# nslookup gw2.kilo2.cyber.test
Server:      186.25.1.2
Address:     186.25.1.2#53

Name:   gw2.kilo2.cyber.test
Address: 213.1.133.99
```

Figure 28 - Nslookup for gw2.kilo2.cyber.test

**3. www.kilo2.cyber.test at 213.1.133.100:**

```
root@a:~# nslookup www.kilo2.cyber.test
Server:      186.25.1.2
Address:     186.25.1.2#53

Name:   www.kilo2.cyber.test
Address: 213.1.133.100
```

Figure 29 - Nslookup for www.kilo2.cyber.test

## Further work

The organisation should consider implementing an internal DNS server with which domain names would be allocated to each internal resource (e.g. *pc1.kilo2.cyber.test*). Such a configuration would have been relatively simple to implement given more time to complete the project.

A reverse DNS zone should be implemented to complement the *cyber.test* zone, allowing clients to lookup domain names from IP addresses.

## Phase Two

### Apache - u2136685

#### HTTPS

In the `apache.startup` file, the Apache web server enables the SSL module, which allows it to enable the “[www.kilo2.cyber.test-ssl.conf](http://www.kilo2.cyber.test-ssl.conf)” file, therefore enabling HTTPS on the web server.

```
### enabling ssl on apache ###

a2enmod ssl
systemctl restart apache2
a2ensite www.kilo2.cyber.test-ssl.conf
systemctl reload apache2
systemctl restart apache2
```

Figure 30 - Enabling SSL and the HTTPS website config file

The “[www.kilo2.cyber.test-ssl.conf](http://www.kilo2.cyber.test-ssl.conf)” specifies the SSL certificates used in order to use HTTPS, shown below. The Apache certificate is issued by the intermediate Web CA. The Web and Root CA certificates and path are also specified.

```
# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/ipsec.d/certs/www.kilo2.cyber.test.pem
SSLCertificateKeyFile /etc/ipsec.d/private/www.kilo2.cyber.test.pem

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convenience.
SSLCertificateChainFile /etc/ipsec.d/cacerts/web_Cert.pem

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
SSLCACertificatePath /etc/ipsec.d/cacerts/
SSLCACertificateFile /etc/ipsec.d/cacerts/rootCA_Cert.pem
```

Figure 31 - Enabling SSL certificates

The Apache certificate can be seen in Figure 22, whereas the Web CA certificate and Root CA certificate can be seen in Figure 18 and 13, respectively.

The HTTPS website can be accessed by any machine. We can verify this by running the command “`lynx https://www.kilo2.cyber.test`” and analyzing the packets leaving gw3. The website displays an HTTPS ASCII art, as seen below.



Figure 32 - “lynx <https://www.kilo2.cyber.test>” output

We can verify that the traffic was encrypted by looking at the PCAP file generated.

4	2023-02-16	186.25.1.2	213.1.133.101	DNS	136 Standard query response 0xb7a0 AAAA www.kilo2.cyber.test SOA dns.cyber.test
5	2023-02-16	da:5f:16:74:8c:80	Broadcast	ARP	42 Who has 213.1.133.100? Tell 213.1.133.101
6	2023-02-16	0e:01:ee:01:ee:01	da:5f:16:74:8c:80	ARP	42 213.1.133.100 is at 0e:01:ee:01:ee:01
7	2023-02-16	213.1.133.101	213.1.133.100	TCP	74 53154 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1404214904 TSecr=0 WS=16
8	2023-02-16	213.1.133.100	213.1.133.101	TCP	74 443 → 53154 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3191447176 TSecr=1404214904 WS=16
9	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 53154 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0 TSval=1404214905 TSecr=3191447176
10	2023-02-16	213.1.133.101	213.1.133.100	TLSv1.3	468 Client Hello
11	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 443 → 53154 [ACK] Seq=1 Ack=403 Win=64768 Len=0 TSval=3191447217 TSecr=1404214945
12	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	1350 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
13	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 53154 → 443 [ACK] Seq=403 Ack=1285 Win=64096 Len=0 TSval=1404214955 TSecr=3191447226
14	2023-02-16	213.1.133.101	213.1.133.100	TLSv1.3	72 Change Cipher Spec
15	2023-02-16	213.1.133.100	213.1.133.101	TCP	66 443 → 53154 [ACK] Seq=1285 Ack=409 Win=64768 Len=0 TSval=3191447284 TSecr=1404214967
16	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	403 Application Data, Application Data
17	2023-02-16	213.1.133.100	213.1.133.101	TCP	66 443 → 53154 [ACK] Seq=1285 Ack=746 Win=64432 Len=0 TSval=3191447284 TSecr=1404215013
18	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	337 Application Data
19	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	337 Application Data
20	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 53154 → 443 [ACK] Seq=746 Ack=1827 Win=64096 Len=0 TSval=1404215016 TSecr=3191447286
21	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	457 Application Data
22	2023-02-16	213.1.133.100	213.1.133.101	TLSv1.3	90 Application Data

Figure 33 - GW3 PCAP HTTPS capture

As seen in Figure 33, the TLSv1.3 protocol is used, which is the Transport Layer Security cryptographic protocol used to secure HTTPS. Also, TLS is the successor to SSL, which is deprecated.

If a user connects to the website through HTTP, they will be greeted by the same style of ASCII art, but it only spells out HTTP instead.



Figure 34 - “lynx <http://www.kilo2.cyber.test>” output

Analysing this PCAP file from gw3’s eth0 card, in Figure 33, we can clearly see the difference, as the HTTP protocol is clearly used.

1	2023-02-16	213.1.133.101	186.25.1.2	DNS	80 Standard query 0x65c7 A www.kilo2.cyber.test
2	2023-02-16	213.1.133.101	186.25.1.2	DNS	80 Standard query 0x3d9e AAAA www.kilo2.cyber.test
3	2023-02-16	186.25.1.2	213.1.133.101	DNS	96 Standard query response 0x65c7 A www.kilo2.cyber.test A 213.1.133.100
4	2023-02-16	186.25.1.2	213.1.133.101	DNS	136 Standard query response 0x3d9e AAAA www.kilo2.cyber.test SOA dns.cyber.test
5	2023-02-16	213.1.133.101	213.1.133.100	TCP	74 80 → 37634 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1404892722 TSecr=0 WS=16
6	2023-02-16	213.1.133.100	213.1.133.101	TCP	74 80 → 37634 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3192124994 TSecr=1404892722
7	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 37634 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0 TSval=1404892723 TSecr=3192124994
8	2023-02-16	213.1.133.101	213.1.133.100	HTTP	307 GET / HTTP/1.0
9	2023-02-16	213.1.133.100	213.1.133.101	TCP	66 80 → 37634 [ACK] Seq=1 Ack=242 Win=64928 Len=0 TSval=3192124995 TSecr=1404892724
10	2023-02-16	213.1.133.100	213.1.133.101	HTTP	432 HTTP/1.1 200 OK (text/html)
11	2023-02-16	213.1.133.100	213.1.133.101	TCP	66 80 → 37634 [FIN, ACK] Seq=367 Ack=242 Win=64928 Len=0 TSval=3192124998 TSecr=1404892724
12	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 37634 → 80 [ACK] Seq=242 Ack=367 Win=64096 Len=0 TSval=1404892727 TSecr=3192124998
13	2023-02-16	213.1.133.101	213.1.133.100	TCP	66 37634 → 80 [FIN, ACK] Seq=242 Ack=368 Win=64096 Len=0 TSval=1404892731 TSecr=3192124998
14	2023-02-16	213.1.133.100	213.1.133.101	TCP	66 80 → 37634 [ACK] Seq=368 Ack=243 Win=64928 Len=0 TSval=3192125002 TSecr=1404892731

Figure 33 - GW3 PCAP HTTP capture

In Figure 34, we can even see the HTML file transferred to the client.

```

Transmission Control Protocol, Src Port: 80, Dst Port: 80
Hypertext Transfer Protocol
Line-based text data: text/html (6 lines)
<pre>\n
      \n
      ||h |||t |||t |||p ||      \n
      ||_|||_|||_|||_|||_||      \n
      |/_\|/_\|/_\|/_\|/_\|      \n
</pre>\n

```

Figure 34 - HTML file seen in WireShark packet

## Further work

We would add a way to revoke certificates in case of a breach that compromises the web server. Moreover, we would add stateful firewall rules specific to the 80 and 443 ports, to tightly control network traffic.

## StrongSwan - u2101239

The Netkit configuration facilitates remote access to the internal Warwick network via a StrongSwan realisation of an IPsec gateway, *gw2*. Clients from both LAN A and LAN B can tunnel to *gw2* provided that they trust the certificate authorities involved and possess the required secrets. Henderson, 2023 was primarily used as an assisting resource in the configuration of StrongSwan.

The IPSec responder is configured to use the connection identified by *ikev2-lan-b*, shown in Figure 35:

```
conn base
  fragmentation=yes
  dpdaction=clear
  dpddelay=350s
  ike=chacha20poly1305-sha512-curve25519-prfsha512!
  esp=chacha20poly1305-sha512!
  keyingtries=%forever
  leftid=213.1.133.99
  leftcert=vpn_gw2_Cert.pem

conn ikev2-lan-b
  also=base
  leftsendcert=always
  keyexchange=ikev2
  leftsubnet=192.168.87.0/24
  rightsourcexp=10.2.2.0/25
  rightsendcert=never
  right=%any
  rightid=%any
  type=tunnel
  rightauth=eap-mschapv2
  eap_identity=%identity
  auto=start
```

Figure 35 - The *ikev2-lan-b* StrongSwan connection and its dependencies

The VPN is configured to use the IKEv2 protocol, with only one cryptographically secure cipher suite being negotiable. Perhaps this is excessively restrictive and may prevent remote workers with



older devices not supporting such algorithms from connecting. Notwithstanding, this configuration ensures that only strong cryptographic algorithms are employed. Additionally, users must authenticate themselves with an identity and password associated with that identity over the EAP-MSCHAPv2 protocol.

The StrongSwan gateway identifies itself using a certificate, signed by the intermediate VPN CA. Remote peers are sent this certificate before the tunnel is established, allowing them to decide whether or not to trust the source.

Virtual IPs from the subnet 10.2.2.0/25 are allocated to authenticated users, and a route to this network via *gw2* has been specified in *gwWW* in order to allow internal systems to respond to requests originating from peers in this subnet.

The following pcap (ipsec-b-pc1.pcap) excerpt shows encrypted traffic between the *pc1* and *b* machines:

4	14.385179	198.51.100.2	213.1.133.99	ESP	162	ESP (SPI=0xcb953ecf)
5	14.385518	213.1.133.99	198.51.100.2	ESP	162	ESP (SPI=0xc85be675)
6	15.460217	198.51.100.2	213.1.133.99	ESP	162	ESP (SPI=0xcb953ecf)
7	15.460737	213.1.133.99	198.51.100.2	ESP	162	ESP (SPI=0xc85be675)
8	16.461932	198.51.100.2	213.1.133.99	ESP	162	ESP (SPI=0xcb953ecf)

Figure 36 - Encrypted ping requests between *pc1* and *b*, a StrongSwan client

Notice that “ESP” (Encapsulating Security Payload) has been identified as the protocol used. This suggests that the traffic was encrypted over an IPsec tunnel.

## WireGuard - u2136685

### Several sample mobile workers

As was demonstrated in phase one, the WireGuard workers were from the same LAN, LAN B. For phase two, we implemented WireGuard for LAN A as well. The WireGuard gateway, *gw1*, can be seen allowing peers connecting from LAN A in Figure 8.

The LAN A workers also needed configuration files to be able to connect to the gateway and to talk to the Warwick local machines. The following Figures 37 and 38 show the same configuration as in Figures 9 and 10, along with the private key leakage prevention.

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
ListenPort = 51000
Address = 10.90.90.3/24

[Peer]
PublicKey = Y1tCQHbAlpYo5xZz0PtE+m1Rd9j4tL0gO44KcBnshI=
Endpoint = 213.1.133.98:51000
AllowedIPs = 10.90.90.0/24,192.168.87.0/25,192.168.87.128/25
```

Figure 37 - Worker “a” *wg0.conf* file

```

[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
ListenPort = 51000
Address = 10.90.90.4/24

[Peer]
PublicKey = Y1tCQHbAlpYo5xZz0PtE+m1Rd9j4tl0g044KcBnshI=
Endpoint = 213.1.133.98:51000
AllowedIPs = 10.90.90.0/24,192.168.87.0/25,192.168.87.128/25

```

Figure 38 - Worker “a2” wg0.conf file

The workers from LAN A have the same abilities as the workers from LAN B, being able to ping machines in the Warwick LAN, being pinged by machines in the Warwick LAN and communicating with each other, even across LANs, as seen in Figure 41.

```

root@a:~# ping 192.168.87.130
PING 192.168.87.130 (192.168.87.130) 56(84) bytes of data:
64 bytes from 192.168.87.130: icmp_seq=1 ttl=61 time=1.28 ms
64 bytes from 192.168.87.130: icmp_seq=2 ttl=61 time=93.7 ms
64 bytes from 192.168.87.130: icmp_seq=3 ttl=61 time=1.55 ms
64 bytes from 192.168.87.130: icmp_seq=4 ttl=61 time=1.44 ms
^C
--- 192.168.87.130 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.280/24.484/93.676/39.947 ms

```

Figure 39 - Pinging a Warwick local machine from worker “a”

1	2023-02-16 02:53:40...	192.0.2.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0xDA62B47C, counter=3, datalen=96
2	2023-02-16 02:53:40...	213.1.133.98	192.0.2.2	WireGuard	170 Transport Data, receiver=0xFE98E6CA, counter=2, datalen=96
3	2023-02-16 02:53:41...	192.0.2.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0xDA62B47C, counter=4, datalen=96
4	2023-02-16 02:53:42...	213.1.133.98	192.0.2.2	WireGuard	170 Transport Data, receiver=0xFE98E6CA, counter=3, datalen=96
5	2023-02-16 02:53:42...	192.0.2.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0xDA62B47C, counter=5, datalen=96
6	2023-02-16 02:53:42...	213.1.133.98	192.0.2.2	WireGuard	170 Transport Data, receiver=0xFE98E6CA, counter=4, datalen=96
7	2023-02-16 02:53:43...	192.0.2.2	213.1.133.98	WireGuard	170 Transport Data, receiver=0xDA62B47C, counter=6, datalen=96
8	2023-02-16 02:53:43...	213.1.133.98	192.0.2.2	WireGuard	170 Transport Data, receiver=0xFE98E6CA, counter=5, datalen=96
9	2023-02-16 02:53:45...	02:b2:02:b2:01:b2	02:00:00:09:09:09	ARP	42 Who has 213.1.133.97? Tell 213.1.133.98
10	2023-02-16 02:53:45...	02:00:00:09:09:09	02:b2:02:b2:01:b2	ARP	42 213.1.133.97 is at 02:00:00:09:09:09

Figure 40 - The ICMP packets are hidden in the WireGuard tunnel

a2	b
root@a2:~# nc 10.90.90.5 1234	root@b:~# nc -l 1234
from a2	from a2
from b	from b
□	■

Figure 41 - The workers talking across LANs through WireGuard

## Further work

As stated in phase one, to further increase the security of the WireGuard tunnel, we would implement pre-shared keys for the connection to the gateway as an extra layer of cryptographic security. It would add another layer of symmetric encryption to the WireGuard tunnel, making the network traffic harder to decrypt. (Ubuntu, 2022)

## Phase Three

### DNSSEC - u2101239

The external DNS server running on the machine with name “extDNS” supports the use of DNSSEC when making DNS requests. This was accomplished using the inline signing feature provided by BIND9 (Stirnemann, 2014), a software suite to manage and interact with the domain system.

Evidence of such functionality can be produced by using *dig* with the “dnssec” flag set. The following Figure 42 shows the output of this command when querying *www.kilo2.cyber.test* on the *rB* machine:

```
root@rB:~# dig www.kilo2.cyber.test +dnssec
; <<> Dig 9.16.15-Debian <<> www.kilo2.cyber.test +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20967
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: do: udp: 1232
;; COOKIE: b4d5430849f4a2e3010000063ed64bab8a8596fe79da66a (good)
;; QUESTION SECTION:
;www.kilo2.cyber.test.      IN      A
;; ANSWER SECTION:
www.kilo2.cyber.test.      604800  IN      A          213.1.133.100
www.kilo2.cyber.test.      604800  IN      RRSIG      A 15 4 604800 20230306181459 20230215215918 43507 cyber.test. FHVP/iud9vHu/X5z/8j3DeNraVj0R+9hcbZZxCHEQLfnySbX97u1
Rvig cgmj3R2wDxvsp65uPH/MiruKKxIsDw==
;; Query time: 10 msec
;; SERVER: 186.25.1.2#53(186.25.1.2)
;; WHEN: Thu Feb 16 00:17:58 UTC 2023
;; MSG SIZE rcvd: 199
```

Figure 42 - Output of *dig www.kilo2.cyber.test +dnssec*

Notice that the “ANSWER SECTION” reveals an additional “RRSIG” record, containing the signature used to sign the DNS response. Using this signature, the client can verify that their DNS requests are being served by the intended source.

The elliptic curve ED25519 algorithm was used to create the signing keys on the “extDNS” machine due to its increased performance and security when compared to more traditional alternatives, such as RSA. It also maintains consistency across the organisation, which relied on the same algorithm to generate the certificate hierarchy.

## Further Work

Although clients can now verify the authenticity of the nameserver, the particularly security-conscious of users may wish to preserve the confidentiality of their DNS requests. Hence, an appropriate DNS-over-HTTPS implementation should be considered, whereby user’s requests are encrypted using TLS, preventing packet sniffers from revealing the specific domains that were requested.

# Evaluation of VPN Configurations

WireGuard's source code is less than 4000 lines, compared to StrongSwan's code at 400 000, making it 100 times more lightweight than StrongSwan. (WireGuard, 2018)

WireGuard's setup is much easier, requiring only keys, whereas StrongSwan requires certificates to be generated by certificate authorities, making the setup more cumbersome and time-consuming. The depth of configuration offered by StrongSwan is also far greater than the options presented by WireGuard, making it more complex to implement. However, allowing fine-grained control of the VPN may result in stricter security if configured by a user with advanced knowledge.

One drawback to allowing extensive configuration is that insecure configurations are permitted, whereas WireGuard does not allow the use of legacy functions and is more secure than StrongSwan when operated with default parameters.

An IPSec-based VPN is compatible with a wider range of devices, since it is a much older protocol compared to the novel WireGuard architecture. Support for legacy functionality further extends the range of supported devices, making StrongSwan the better choice for those intending to facilitate connections with clients on older devices.

In terms of network performance, an experiment found that StrongSwan's packet goodput was consistently better than that of WireGuard when specific cipher suites were used, with goodput being a measure of the speed and accuracy with which packets are communicated across the virtual network. Among the most performant ciphers was the "ChaCha20-Poly1305" algorithm, which we employ in our own implementation of StrongSwan. (Dekker and Spaans, 2020)

WireGuard aims to replace older VPN models such as IPSec, being simpler, more lightweight and easier to use and setup. (Dekker and Spaans, 2020; WireGuard, 2018).

# References

Dekker, E. and Spaans, P. (2020). *A Performance Comparison of the VPN Implementations WireGuard, StrongSwan and OpenVPN in a One Gbit/s Environment*. [online] Available at: <https://rp.os3.nl/2019-2020/p71/presentation.pdf> [Accessed 16 Feb. 2023].

Henderson, T. (2023). *Install and Configure StrongSwan on Ubuntu 20.04*. [online] Linode. Available at: <https://www.linode.com/docs/guides/strongswan-vpn-server-install/> [Accessed 16 Feb. 2023].

Muchtadi-Alamsyah, I. and Tama, Y.B.W. (2019). *Implementation of Elliptic Curve25519 in Cryptography*. [online] [www.intechopen.com](http://www.intechopen.com). IntechOpen. Available at: <https://www.intechopen.com/chapters/68653> [Accessed 15 Feb. 2023].

National Cyber Security Centre (2020a). *Certificate Authority hierarchy*. [online] [www.ncsc.gov.uk](http://www.ncsc.gov.uk). Available at: <https://www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/introduction-to-public-key-infrastructure/ca-hierarchy> [Accessed 15 Feb. 2023].

National Cyber Security Centre (2020b). *Keep the root CA offline and be unavailable for use*. [online] [www.ncsc.gov.uk](http://www.ncsc.gov.uk). Available at: <https://www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/pki-principles/keep-the-root-ca-offline-and-be-unavailable-for-use> [Accessed 15 Feb. 2023].

Stirnemann, D. (2014). *DNSSEC signing your domain with BIND inline signing*. [online] Switch. Available at: <https://securityblog.switch.ch/2014/11/13/dnssec-signing-your-domain-with-bind-inline-signing/> [Accessed 15 Feb. 2023].

Ubuntu (2022). *WireGuard VPN - Security Tips*. [online] Ubuntu. Available at: <https://ubuntu.com/server/docs/wireguard-vpn-security> [Accessed 15 Feb. 2023].

WireGuard (2018). *Linux Plumbers Conference*. [online] Available at: <https://www.wireguard.com/talks/lpc2018-wireguard-slides.pdf> [Accessed 16 Feb. 2023].