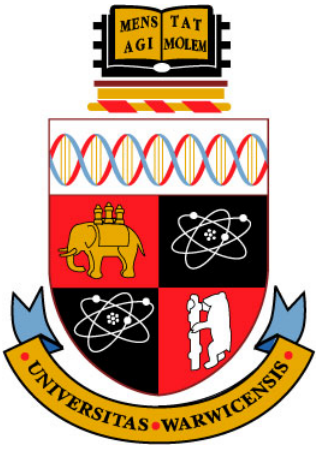

Post Module Assignment Submission Form

MODULE TITLE: Security Architecture and Network Defence

MODULE LECTURER: Peter Norris

MODULE CODE: ES94P-15

MODULE DATES: 11/1/21 – 22/1/21

STUDENT ID NUMBER: 2086937

Contents

List of Figures

List of Tables

Abbreviations

Access Control List	ACL
Demilitarised Zone	DMZ
Denial of Service	DoS
Distributed Denial of Service	DDoS
Dynamic Host Configuration Protocol	DHCP
Generic Receive Offload	GRO
Internet Engineering Task Force	IETF
Internet Protocol	IP
Internet Service Provider	ISP
Intrusion Detection System	IDS
Intrusion Prevention System	IPS
Large Receive Offload	LRO
Lightweight Directory Access Protocol	LDAP
Local Area Network	LAN
Midland Endpoint Mobile Enterprise	MEME
National Institute of Standards and Technology	NIST
Network Interface Card	NIC
Network Intrusion Detection System	NIDS
Network System Monitoring	NSM
Packet Capture file	PCAP
Pluggable Authentication Module	PAM
Secure Shell	SSH
Security Information and Event Management	SIEM
Security Operations Centre	SOC
Virtual LAN	VLAN
Virtual Private Network	VPN
Wide Area Network	WAN

1 Phase 1

The organisation Midland Endpoint Mobile Enterprise (MEME) is planning to move to offices in Coventry and requires the design of a secure network architecture to facilitate their organisational needs. MEME has a workforce of 1000 who are mainly nonsalaried volunteers, but does have system administrators and security operators who are salaried in-house specialists. In addition to this, post-COVID most employees work online and their is only a few technical specialists working on site. MEME works closely with an Australian collaborator and is looking to expand significantly within the next three years. It is the task of this report to model MEME's future network architecture using the User Mode Linux Netkit software. This architecture must include zones of trust (section 1.1), IP addressing (section 2.2), traffic filters (section 1.2), and augmentations that enhance the organisation's security posture (sections 2.3, 3.1, & 3.2). In addition to these features, verification both of connectivity (1.1.2 & 1.2.7) and the prevention of connectivity when unwanted (2.1) are also provided by this report.

1.1 Architecture – Zones of Trust

1.1.1 Design

To provide MEME with a secure network architecture this model segments the organisation's network into zones of trust. These zones contain assets which have been grouped based on their patterns of interaction, access control constraints, and the danger they pose to the organisation if compromised. Each zone is granted some level of trust which determines the network resources that assets in that zone are allowed to reach (Gilman and Barth, 2017:28). This network segmentation allows MEME to monitor and filter traffic which is traversing between zones and provides a strong defence-in-depth model. For MEME the following zones of trust have been established:

- ◇ Demilitarised Zone (DMZ) – contains assets which are public facing and are likely to be targeted by adversaries, thus have a higher risk of compromise
- ◇ Server Zone – contains assets which are accessed by organisation employees internally and need to be monitored.
- ◇ Remote Access Zone – contains VPN endpoints which allow secure remote connections into the organisation for both employees (Wireguard VPN) and executives (OpenVPN). This is particularly relevant due to the post-COVID environment where many employees are working from offsite.
- ◇ Internal Zone – contains assets local to the organisation which onsite employees use. This is further segmented into:
  - ☐ Executive Internal Zone – assets which executive members of the organisation access, which may contain sensitive organisation data.
  - ☐ Employee Internal Zone – assets which general employees use.
  - ☐ Secured Internal Zone – assets used by local employees which need to be more closely monitored as they will do significant damage to the organisation if compromised.
- ◇ Admin Zone – assets which the onsite system administration and security operators will use to monitor and maintain firewalls. This zone makes up the Management Network which is an augmentation discussed in section 3.1 designed to enhance the security of MEME's network.

These zones can be visualised in the figure 1.1 below, which shows how traffic comes into the organisation from the Internet and is then routed by the Perimeter Firewall to one of MEME's zones of trust.

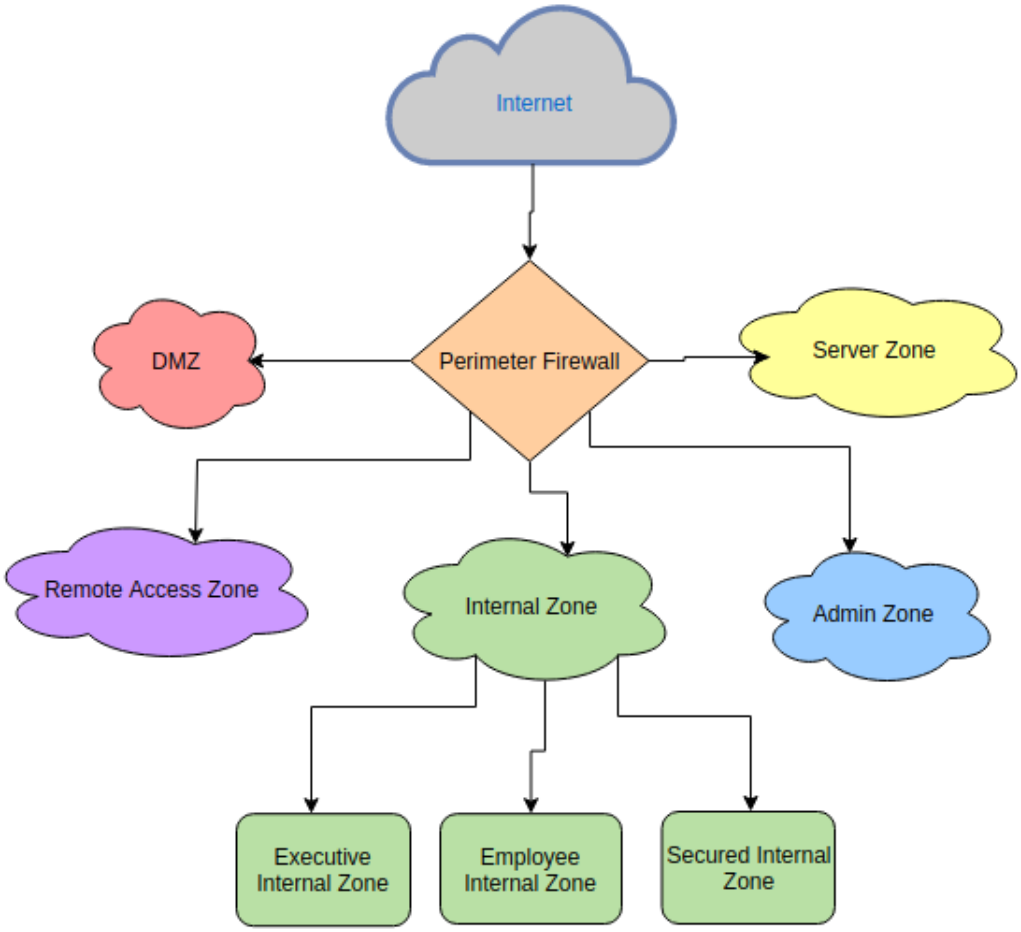

Figure 1.1: Zone of Trust in MEME Network

In the Netkit model the gateway router and Perimeter Firewall are implemented through machine `PerimF`. In practice this machine filters traffic and then routes it to another gateway router at the edge of each zone of trust. This gateway router then filters the traffic again, with more specific firewall rules, and then passes the filtered traffic onto the subnet that it is connected to. These subnets are the practical realisation of the zones of trust model and allow MEME to segment it's network into smaller LANs which each have their own, separate, IP address ranges. The practical network architecture implemented by this model can be found in figure 1.2 below.

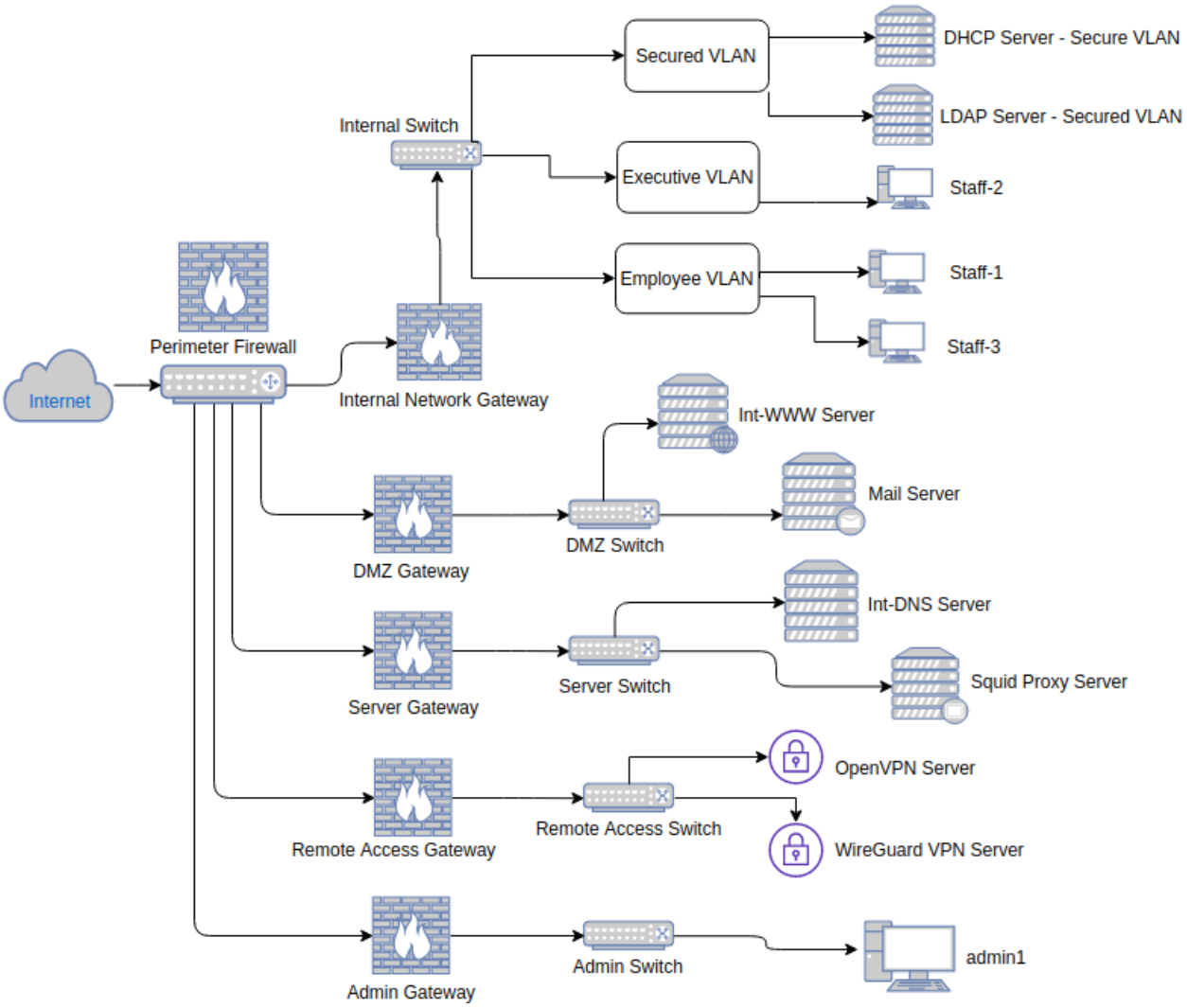

Figure 1.2: MEME Network Diagram

As previously discussed, traffic enters the organisation through the Perimeter Firewall machine `PerimF` which has as a public (Internet) facing network card with the IP `137.205.1.1` and connects to the `Internet` machine via it's `137.205.157.254` interface. This public-facing gateway machine has been added so that a perimeter firewall can be implemented and traffic can be routed to a zone of trust (a segmented LAN). Each of these zones of trust has a gateway router which has a network interface which is part of the perimeter LAN `192.168.1.0/24` and this connects to the Perimeter Firewall machine's internal network interface `192.168.1.1`. Hence, outside traffic can be routed via the Perimeter Firewall, to a zone of trust gateway, and then onto each zone of trust. However, the traffic which traverses this route must pass through the Perimeter Firewall and then a gateway firewall implemented on each zone of trust's gateway machine, and vice versa for traffic travelling out of each zone of trust. In addition to this, the Perimeter Firewall machine `PerimF` can route traffic internally between zones of trust, and this traffic is also filtered. The filtering at the gateway machines is based on the assets within their corresponding zone of trust (subnet) and is discussed in detail in section 1.2. After filtering, the traffic can then reach the assets within the subnet, be it a server, VPN gateway, employee machine, etc., for MEME these assets have been grouped into different subnets (zones of trust) as

show in figure 1.2. This network architecture is predominately a reorganising of assets which MEME already had into segmented subnets, however this reorganisation required several infrastructure assets to be added to MEME's existing network. These include:

- ◇ The Perimeter Firewall machine `PerimF` which performs initial packet filtering and then routes traffic to a zone of trust via an internal gateway machines.
- ◇ The internal gateway machines which further filter traffic and pass it onto their zone of trust (subnet). These include; `IntGW`, `DmzGW`, `SrvGW`, `RemoteAccessGW`, & `AdminGW`.
- ◇ A switch for each zone of trust (subnet) to intelligently connect each asset on the subnet, including the subnet's gateway machine. In the model these are:
  - ☐ `INT` which connects assets in the Internal zone LAN – `DHCP`, `LDAP`, `Staff-1`, `Staff-2`, & `Staff-3`. These assets are further organised into Virtual LANs (VLANs), discussed below.
  - ☐ `DMZ` which connects assets in the DMZ LAN – `Int-WWW` & `Mail`
  - ☐ `SRV` which connects assets in the Server zone LAN – `Int-DNS`, `Squid`
  - ☐ `RemoteAccess` which connects assets in the Remote Access zone LAN – `OpenVPN` & `WireGuardVPN`
  - ☐ `ADMIN` which connects assets in the Admin zone LAN – `admin1`
- ◇ An admin machine (`admin1`) which is required to manage all the gateway firewalls.

Note, switches were used in this network because they have a higher port density than most other network devices (i.e. hubs), are intelligent (learn MAC addresses), can be configured to support multiple VLANs (which logically segments a network), can be configured to monitor SNMP, provide VLAN trunking functionality, have span ports for network management, offer port security options through static or dynamic MAC address-based controls, and support the 802.1X standard for port-level authentication (Brotherston and Berlin, 2017:150). These features make switches more secure than hubs and other network devices, thus making them the obvious choice of network hardware when implementing a secure network architecture. A summary of these assets, and at what IP address these asset are located at, can be found in table 1.1 below. Each asset is grouped by their LAN (or zone of trust) and corresponding subnet.

Public Facing LAN [137.205.0.0/16]	
PerimF	137.205.1.1
Internet	137.205.167.254
Perimeter LAN [192.168.1.0/24]	
PerimF	192.168.1.1
IntGW	192.168.1.101
DmzGW	192.168.1.102
SrvGW	192.168.1.103
RemoteAccessGW	192.168.1.104
AdminGW	192.168.1.105
Internal LAN [10.10.1.0/24] – Internal Zone	
IntGW	10.10.1.1, 10.10.11.1, 10.100.0.1, 10.10.33.1
Executive VLAN	10.10.11.0/24
Employee VLAN	10.100.0.0/16
Secured VLAN	10.10.33.0/24
DMZ LAN [10.10.2.0/24] – DMZ	
DmzGW	10.10.2.1
Int-WWW	10.10.2.2
Mail	10.10.2.3
Server LAN [10.10.3.0/24] – Server Zone	
SrvGW	10.10.3.1
Int-DNS	10.10.3.2
Squid	10.10.3.3
Remote Access LAN [10.10.4.0/24] – Remote Access Zone	
RemoteAccessGW	10.10.4.1
OpenVPN	10.10.4.2
WireGuardVPN	10.10.4.3
Admin LAN [10.10.5.0/24] – Admin Zone	
AdminGW	10.10.5.1
admin1	10.10.5.201

Table 1.1: Summary of MEME Network Assets

In addition to these assets, there are infrastructure assets within the Internal zone which are used to further segment this subnet in VLANs to provide greater security. VLANs are virtual LANs that allow physically remote assets to communicate over, what is logically, the same subnet. This offers benefits in flexibility (networks can span multiple physical locations), performance (VLAN traffic can be prioritised), cost (don't need additional routers), and management (VLAN configuration can be done in one location as oppose to every device) (Brotherston and Berlin, 2017:158). Adding to this, VLANs also provide inherent security as switches which implement them only deliver the frames within the destined VLAN when sending broadcasts and different VLANs can be configured to implement different access control policies, or firewalls, (Brotherston and Berlin, 2017: 158). The firewalls implement for the VLANs in this model are discussed in section 1.2.

The VLANs this report recommends MEME implement for the Internal zone are an Executive VLAN, Employee VLAN, and a Secured VLAN, as previously stated. These

VLANs were chosen because they effectively segment internal network traffic based on the sensitivity of data (i.e. sensitive data will travel on the Executive VLAN, whereas general employee data will travel on the Employee VLAN), as well as risk level (i.e. the Secured VLAN contains the most important internal assets which are subject to the strictest monitoring). To achieve this, this Netkit model uses two distribution switches `Int-D0` & `Int-D1` and two access switches `Int-A0` & `Int-A1`. The distribution switches are used so that multiple access switches can be connected to each one and be physically separate from each other (i.e. to accommodate the layout of MEME's new Coventry offices). These distribution switches each connect to the main internal network switch `INT` which connects to the internal network's gateway router `IntGW`. The access switches are the switches which MEME's assets, geographically separated around the offices, are physically connected to for connectivity of all devices on the internal network. In this model each of the distribution switches are connected to each of the access switches to provide the internal network with redundant paths so that business continuity can be maintained even if one network infrastructure asset fails. Thus, to prevent the creation of network loops in the internal network the Spanning Tree Protocol (STP) is used on each switch. STP prevents network loops (a.k.a. frame broadcast storms) from disrupting the flow of LAN traffic by virtually disconnecting redundant links and then re-instantiating them if needed (Beaming Support, 2018). This whole internal network topology can be visualised in the network diagram in figure 1.3.

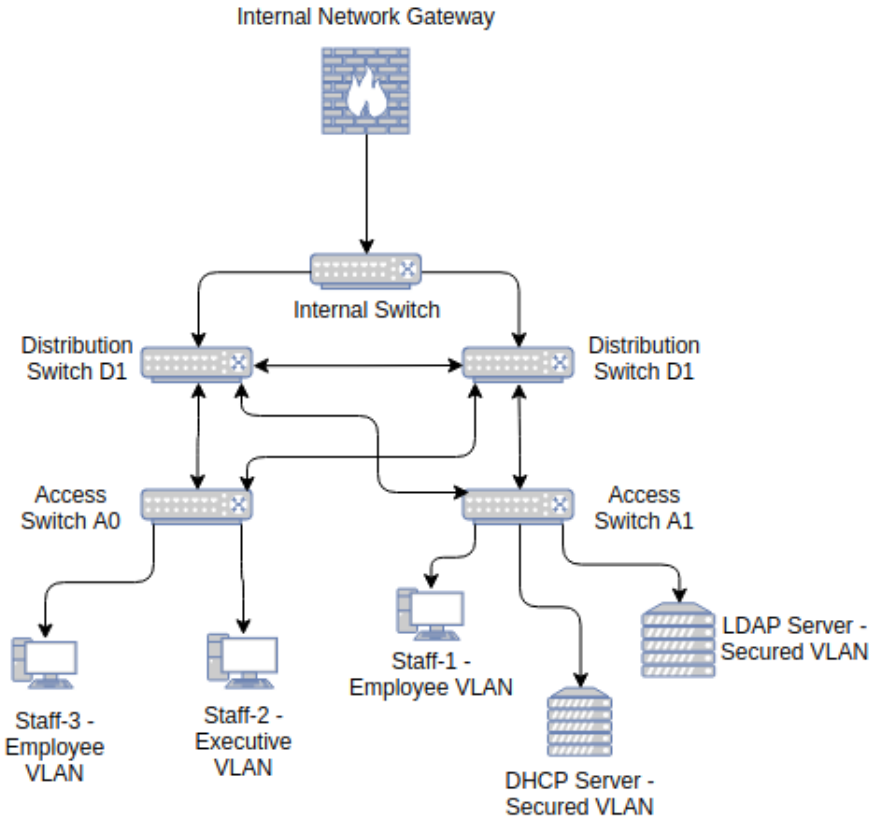

Figure 1.3: MEME Internal Zone Network Diagram

These switches ultimately allow assets on the three VLANs to efficiently and securely pass traffic around MEME's Internal zone. These include:

- ◇ `Staff-1` & `Staff-3` which are used to represent general MEME employee machines who do their work on the Employee VLAN.
- ◇ `Staff-2` which represents an executive employee machine (i.e. a manager) who is working on the Executive VLAN.
- ◇ `DHCP` which represents a server on the Secured VLAN and is used to dynamically assign IP addressed to all employees on the Internal network zone.
- ◇ `LDAP` which also represents a server on the Secured VLAN and is used for authenticating and storing information about; users, groups, and applications (Wilson, 2018).

A summary of the Internal zone assets, and at what IP address these assets are located, can be found in table 1.2 below. Each asset is grouped by their VLAN and corresponding subnet.

Executive VLAN [10.10.11.0/24] – VLAN ID 111	
`IntGW` `eth0.111`	10.10.11.1 (default gateway)
`Staff-2`	10.10.11.2
Employee VLAN [10.100.0.0/16] – VLAN ID 222	
`IntGW` `eth0.222`	10.100.0.1 (default gateway)
`Staff-1`	10.100.1.2
`Staff-3`	10.100.1.3
Secured VLAN [10.10.33.0/24] – VLAN ID 333	
`IntGW` `eth0.333`	10.10.33.1 (default gateway)
`LDAP`	10.10.33.2
`DHCP`	10.10.33.3

Table 1.2: Summary of MEME Internal Network Assets

This VLAN internal network uses the virtual interface cards `eth0.111`, `eth0.222`, & `eth0.333` to route traffic between VLANs through the `IntGW` gateway router. The specific firewall rules implemented in this model are discussed further in section 1.2.

1.1.2 Verification of Default Connectivity

To demonstrate that this initial network architecture, devoid of any traffic filters or other security modifications, works as appropriate, several terminal grabs from inside the Netkit model have been included.

Connection can be made between internal and external assets Firstly, this network architecture, which establishes zones of trust through subnets, provides a route for internal and external assets to communicate, by default. This connectivity means that all assets can send packets to each other, regardless of which subnet they are on. This connectivity is pre-filtering and requires the Perimeter Firewall machine `PerimF` and internal gateway firewall machines `IntGW`, `DmzGW`, `SrvGW`, `RemoteAccessGW`, & `AdminGW` to have routing capabilities. Firstly, to demonstrate that internal and external assets can communicate a ping from `Int-WWW` (at internal IP `10.10.2.2`) to `Ext-Office` (at public IP `22.39.224.18`) has been performed. A terminal grab of this can seen below.

```
root@Int-WWW:~# ping -c2 22.39.224.18
PING 22.39.224.18 (22.39.224.18) 56(84) bytes of data.
64 bytes from 22.39.224.18: icmp_seq=1 ttl=61 time=4.14 ms
64 bytes from 22.39.224.18: icmp_seq=2 ttl=61 time=4.07 ms

--- 22.39.224.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 4.068/4.101/4.135/0.033 ms
root@Int-WWW:~#
```

The ICMP ping packet is routed from the `10.10.2.0/24` subnet (DMZ LAN) of `Int-WWW` via `DmzGW` to `192.168.1.0/24` subnet (perimeter LAN), then to the `137.205.0.0/16` subnet (MEME's public IP range) via `PerimF`, and finally through the `Internet` machine to reach the `22.39.224.16/30` subnet of the `Ext-Office` machine. This journey is then repeated when `Ext-WWW` responds with a ICMP reply packet. For MEME's internal network two routers were required. Next, to demonstrate that internal assets within the MEME network can communicate with one another a DNS lookup for the internal web server `www.meme.cyber.test` was performed. This involved firstly configuring DNS to work with the new network configuration and then providing internal assets with connectivity to the DNS server, which is located within the Server LAN (subnet `10.10.3.0/24`). To demonstrate this connectivity a DNS request was made from the `Staff-1` machine, at IP address `10.100.1.2`, to `Int-DNS`, at IP address `10.10.3.2`. A terminal grab of this can seen below.

```
root@Staff-1:~# host -t a www.meme.cyber.test 10.10.3.2
Using domain server:
Name: 10.10.3.2
Address: 10.10.3.2#53
Aliases:

www.meme.cyber.test has address 10.10.2.2
root@Int-WWW:~#
```

This time a DNS request packet (sent over UDP as oppose to TCP) is generated by `Staff-1` (located on the Employee VLAN `10.100.0.0/16`) is routed to it's gateway router `IntGW` via it's `eth0.222` interface (this is created to receive/send VLAN traffic with the VLAN ID 222). Once it reaches this interface `IntGW` routes this traffic to the `192.168.1.0/24` subnet and onto `PerimF`, which routes it to the `SrvGW` (the Server LAN gateway) machine. Then this gateway machine routes it onto the Server subnet `10.10.3.0/24` and to the `Int-DNS` server (IP address `10.10.3.2`). As before, this journey is repeated when `Int-DNS` sends DNS response packet back to `Staff-1`. Note, all machines on MEME's network have connectivity to all other internal and external machines, by default. These these machines (and packet types) were just used for demonstration purposes.

There are separate subnets were filters can be implemented In the previous version of MEME's network all assets sat on the `137.205.0.0/16` subnet (MEME's public IP space), which meant any asset on this subnet could be reached by any user on the Internet (WAN), as the Internet routed traffic to this public network. This was a security concern because it meant any MEME asset could be targeted by a malicious actor who had an Internet connection, thus all of MEME's assets are now hiden behind

a perimeter gateway `PerimF`, and grouped into subnets (zones of trust). By default, the perimeter gateway and internal gateways allow traffic to be routed between subnets and to/from MEME's network, however, by implementing subnets firewalls can be created to filter the traffic travelling between these subnets. For instance, the perimeter gateway machine will become the Perimeter Firewall machine and the internal gateway machines will become the Internal Firewall machines. This is demonstrable by the existence of the machines; `PerimF`, `IntGW`, `DmzGW`, `SrvGW`, `RemoteAccessGW`, & `AdminGW`, because within their `.startup` files are routing tables and network interfaces which define each of the 5 main subnets created. Each of these machines acts as a router which connects to subnets, each subnet having different IP address ranges, thus travelling packets pass through these routers (gateways) and can be filtered.

There are separate VLANs were filters can be implemented Aside from the subnets created for each zone of trust, this network now has separate subnets in the form of VLANs which provide added flexibility and security whilst saving on the cost of additional routers. VLANs have been created for three distinct types of internal network traffic – Executive traffic, Employee traffic, and Secured traffic – as previously discussed. The evidence of these VLANs can be found in `PerimF.startup` which routes to these VLANs, `IntGW.startup` which has virtual network interfaces to route traffic between these VLANs, the distribution switches `D0.startup` & `D1.startup` which pass this VLAN traffic, and the access switches `A0.startup` & `A1.startup` which implement VLAN traffic by defining which of their switch ports carry what VLAN traffic. This VLAN traffic is denoted by the VLAN ID which is tagged to all traffic that comes through a certain ports on these access switches. Also, note that each of these VLANs define a different subnet which can carry a different number of hosts (networked devices). This report has given the Employee subnet a /16 subnet, which can carry a theoretical total of 65536 (minus 2 special addresses), because of MEME's plans to expand the organisation and, as such, a large subnet for employees has been created which allow the organisation to grow. However, other VLANs can be also be created in the future as the MEME expands, perhaps to differentiate between different organisational departments if this happens in the future.

1.2 Traffic filters

Now that the network architecture has been established, and zones of trust have been created within the MEME network through the use of subnets, traffic filters can be configured on gateway machines (network firewalls) and on endpoints (host firewalls). These traffic filters allow MEME's system administration and security operators to efficiently filter out packets which may be potentially harmful to assets within the MEME network and defend against breaches. Traffic filters (firewalls) are the most important security device for a network and enforce a predetermined set of rules as to what packets are allowed into, and out of, a subnet (Pleeger et al., 2015:451–452). In this model there are five locations where firewalls have been instantiated:

- Perimeter firewall at `PerimF` – this firewall sits at the edge of MEME's network and routes traffic to each zone of trust. It implements SNAT and DNAT (see section 2.2) and it is MEME's first line of defence.
- Internal zone firewalls – these firewalls filter traffic which is travelling in/out of MEME's internal network (local, onsite network), while also filtering traffic that is travelling between the VLANs within this internal network.

- ◇ DMZ firewalls – these firewalls filter traffic which is flowing to public facing services within the DMZ subnet.
- ◇ Server zone firewalls – these firewalls filter traffic that makes use of MEME's internal servers.
- ◇ Remote Access zone firewalls – these firewalls filter traffic from remote employees who need to access internal resources through a VPN endpoint.
- ◇ Admin zone firewalls – these firewalls filter traffic from administrators who need to access MEME assets to perform system administration via the Management Network discussed in section 3.1.

The firewall rules that are used on each machine within these fire-walled locations are discussed in the preceding sections.

1.2.1 Perimeter Firewall

The perimeter firewall is MEME's network's first line of defence against cyber attacks. It sits on the `PerimF` machine which acts as a gateway for the assets on MEME's network to the outside. The firewall rules that are implemented here are are outlined below in table 1.3. Note, that SNAT and DNAT rules, that are also implemented at this firewall, have been left till section 2.2.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Web traffic (ports 80, 443) – inbound	Forward	External IP addresses	`Int-WWW` web server
Web traffic (ports 80, 443) – outbound	Forward	`Int-WWW` proxy server	External IP addresses
Web traffic (ports 80, 443) – outbound	Forward	`Squid` proxy server	External IP addresses
Squid proxy traffic (3128)	Forward	Executive & Employee VLANs	`Squid` proxy server
DNS traffic (TCP & UDP 53) – external	Forward	`Squid` proxy server	8.8.8.8 IP address (DNS server)
DNS traffic (TCP & UDP 53) – external	Forward	`Mail` server	8.8.8.8 IP address (DNS server)
DNS traffic (TCP & UDP 53) – external	Forward	Remote Access zone subnet	8.8.8.8 IP address (DNS server)
DNS traffic (UDP 53) – internal	Forward	Remote Access subnet, Admin subnet, all three VLANs	`Int-DNS` server
SMTP traffic (ports 25, 587) – inbound	Forward	External IP addresses (mail servers)	`Mail` server
SMTP traffic (ports 25, 587) – outbound	Forward	`Mail` server	External IP addresses (mail servers)
IMAP traffic (port 993)	Forward	Remote Access zone, Executive VLAN, & Employee VLAN	`Mail` server
Encrypted Wireguard traffic (UDP 51820) – inbound	Forward	External IP addresses (Wireguard endpoint)	`WireGuardVPN` endpoint
Encrypted Wireguard traffic (UDP 51820) – outbound	Forward	`WireGuardVPN` endpoint	External IP addresses (Wireguard endpoint)
Unencrypted Wireguard traffic	Forward	`WireGuardVPN` endpoint	Employee VLAN
Encrypted OpenVPN traffic (TCP/UDP 1194) – inbound	Forward	External IP addresses (OpenVPN endpoint)	`OpenVPN` endpoint
Encrypted OpenVPN traffic (TCP/UDP 1194) – outbound	Forward	`OpenVPN` endpoint	External IP addresses (OpenVPN endpoint)
Unencrypted OpenVPN traffic	Forward	`OpenVPN` endpoint	Executive VLAN
SSH traffic (port 22)	Forward	Admin zone subnet	All zone subnets
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.3: Firewall Rules on `PerimF` Machine

The rules outlined for the `PerimF` machine can be found instantiated within it's `PerimF.startup` file and seen in action by issuing the `iptables -L -nv` command on the `PerimF` machine.

1.2.2 Internal Zone Firewalls

The internal zone is home to the onsite employees and executives who are working at the Coventry offices. This zone is compartmentalised into three additional zones; the *Secured VLAN*, the *Executive VLAN*, and the *Employee VLAN*. As such, firewalls within this zone need to filter traffic coming into and out of this larger internal zone and the traffic which is travelling in-between the smaller VLANs. To do this effectively there is a network-based firewall on the `IntGW` machine and host-based firewalls on each asset within the internal zone. The host-based firewalls are relative to which VLAN an asset is connected to. For instance, assets on the *Secured VLAN* (`DHCP` & `LDAP` servers) have their own, specific, host-based firewall rules, whereas machines on the *Employee VLAN* (`Staff-1` & `Staff-3`) have a generic Employee host-based firewall, and machines on the *Executive VLAN* (`Staff-2`) have a generic Executive host-based firewall. To begin with the firewall rules for machines on the *Secured VLAN* are outlined below in tables 1.4, 1.5, and 1.6.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
LDAP traffic (UDP/TCP ports 389, 636)	Forward	Executive & Employee VLAN subnets	`LDAP` server
DHCP traffic (UDP/TCP ports 67, 68)	Forward	All three VLANs	`DHCP` server
DNS traffic (UDP/TCP port 53)	Forward	`DHCP` server & `LDAP` server	`Int-DNS` server
SSH traffic (port 22)	Forward	Admin zone subnet	All VLANs
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.4: Firewall Rules on `IntGW` Machine – related to *Secured VLAN*

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
LDAP traffic (UDP/TCP ports 389, 636)	Input & Output	Employee or Executive VLANs	Employee or Executive VLANs
DNS traffic (UDP/TCP port 53)	Input & Output	`Int-DNS` server	`Int-DNS` server
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.5: Firewall Rules on `LDAP` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
DHCP traffic (UDP/TCP ports 67, 68)	Input & Output	Employee or Executive VLANs	Employee or Executive VLANs
DNS traffic (UDP/TCP port 53)	Input & Output	`Int-DNS` server	`Int-DNS` server
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.6: Firewall Rules on `DHCP` Machine

The *Secured VLAN* specific rules outlined for the `IntGW`, `LDAP`, `DHCP` machines can be found instantiated within their `IntGW.startup`, `LDAP.startup`, and `DHCP.startup` files, respectively. These rules can seen in action by issuing the `iptables -L -nv` command on each machine. Aside from these rules, the firewall rules that are specific for the *Employee VLAN & Executive VLAN* are shown in tables 1.7, 1.8, and 1.9. Note, rules previously defined for `IntGW` have been left out for brevity.

Traffic	Action	Source	Destination
Unencrypted Wireguard traffic	Forward	`WireGuardVPN` endpoint	Employee VLAN
Unencrypted OpenVPN traffic	Forward	`OpenVPN` endpoint	Executive VLAN
IMAP traffic (port 993)	Forward	Employee & Executive VLANs	`Mail` server
DNS traffic (UDP/TCP port 53)	Forward	`Int-DNS` server	Employee & Executive VLANs
Squid proxy traffic (ports 3128/3129)	Forward	Employee & Executive VLANs	`Squid` proxy server

Table 1.7: Firewall Rules on `IntGW` Machine – related to *Employee* & *Executive* VLANs

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Unencrypted Wireguard traffic	Input & Output	`WireGuardVPN` endpoint	`WireGuardVPN`
IMAP traffic (port 993)	Input & Output	`Mail` server	`Mail` server
DNS traffic (UDP/TCP port 53)	Output & Input	`Int-DNS` server	`Int-DNS` server
Squid proxy traffic (ports 3128/3129)	Output & Input	`Squid` proxy server	`Squid` proxy server
DHCP traffic (UDP/TCP ports 67, 68)	Output & Input	`DHCP` server	`DHCP` server
LDAP traffic (UDP/TCP ports 389, 636)	Output & Input	`LDAP` server	`LDAP` server

Table 1.8: Firewall Rules for Machines on *Employee VLAN*

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Unencrypted OpenVPN traffic	Input & Output	`OpenVPN` endpoint	`OpenVPN` endpoint
IMAP traffic (port 993)	Output & Input	`Mail` server	`Mail` server
DNS traffic (UDP/TCP port 53)	Output & Input	`Int-DNS` server	`Int-DNS` server
Squid proxy traffic (ports 3128/3129)	Output & Input	`Squid` proxy server	`Squid` proxy server
DHCP traffic (UDP/TCP ports 67, 68)	Output & Input	`DHCP` server	`DHCP` server
LDAP traffic (UDP/TCP ports 389, 636)	Output & Input	`LDAP` server	`LDAP` server

Table 1.9: Firewall Rules for Machines on *Executive VLAN*

The *Employee VLAN* & *Executive VLAN* specific rules outlined for the `IntGW`, `Staff-1`, `Staff-2`, and `Staff-3` machines can be found instantiated within their `IntGW.startup`, `Staff-1`, `Staff-2`, and `Staff-3` files, respectively. These rules can seen in action by issuing the `iptables -L -nv` command on each machine.

1.2.3 DMZ Firewalls

The DMZ has assets which are public facing (`Int-WWW` & `Mail`) and, as such, the firewalls instantiated within this zone are very important. The zone requires web traffic to/from the web server `Int-WWW` from anywhere and email traffic to/from the `Mail`. Specifically, SMTP traffic to/from the outside and IMAP traffic to/from the internal network Executive & Employee VLANs. The filter rules that this report recommends implementing on each of these machines are in tables 1.10, 1.11, and 1.12.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All – stateful packet filtering	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Web traffic (ports 80, 443)	Forward	External IP addresses	`Int-WWW` web server
Web traffic (ports 80, 443)	Forward	`Int-WWW` web server	External IP services
SMTP traffic (ports 25, 587)	Forward	External IP addresses	`Mail` email server
SMTP traffic (ports 25, 587)	Forward	`Mail` email server	External email servers
IMAP traffic (port 993)	Forward	Employee & Executive VLANs	`Mail` email server
DNS traffic (UDP/TCP port 53)	Forward	`Mail` email server & `Int-WWW` web server	External IP address 8.8.8.8
SSH traffic (port 22)	Forward	Admin zone subnet	DMZ subnet
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.10: Firewall Rules for `DmzGW` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Web traffic (ports 80, 443)	Input	External IP addresses	`Int-WWW` web server
Web traffic (ports 80, 443)	Output	`Int-WWW` web server	External web services
DNS traffic (UDP/TCP port 53)	Input	`Int-WWW` web server	External IP address 8.8.8.8
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.11: Firewall Rules for `Int-WWW` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
SMTP traffic (ports 25, 587)	Input	External IP addresses	`Mail` email server
SMTP traffic (ports 25, 587)	Output	`Mail` email server	External IP addresses
IMAP traffic (port 993)	Input & Output	Employee & Executive VLANs	`Mail` email server
DNS traffic (UDP/TCP port 53)	Input	`Mail` email server	External IP address 8.8.8.8
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.12: Firewall Rules for `Mail` Machine

The rules outlined for the `DmzGW`, `Int-WWW`, `Mail` machines can be found instantiated within their `DmzGW.startup`, `Int-WWW.startup`, and `Mail.startup` files, respectively. These rules can seen in action by issuing the `iptables -L -nv` command on each machine.

1.2.4 Server Zone Firewalls

The server zone has the machines; `DmzGW`, `Int-DNS` & `Squid`. This zone requires DNS traffic to/from the DNS server `Int-DNS` from within MEME's internal network. In addition to this, the `Squid` machine is a Squid proxy server which is currently being used by MEME. It is used to provide the organisation with an environment where system administrators can implement access controls, log employee web traffic, and optimise employee Internet access to speed up connections (Squid, 2013). The implementation of Squid in the Netkit model is discussed in depth in section 2.3, but for now this machine requires web traffic to/from the outside and traffic to/from the internal network (Executive & Employee VLANs) via ports 3128 & 3129 on the Squid proxy. The filter rules that this report recommends implementing on each of these machines to achieve this are in tables 1.13, 1.14, and 1.15.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
DNS traffic (UDP/TCP 53) – internal	Forward	`Int-DNS` server	Remote Access subnet, Admin subnet and all three internal VLANs
DNS traffic (UDP/TCP 53) – external	Forward	`Squid` proxy server	External IP address 8.8.8.8
Web traffic (ports 80, 443)	Forward	`Squid` proxy server	External IP addresses
Squid proxy traffic (ports 3128/3129)	Forward	Employee & Executive VLANs	`Squid` proxy server
SSH traffic (port 22)	Forward	Admin zone subnet	Server zone subnet
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.13: Firewall Rules for `SrvGW` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
DNS traffic (UDP/TCP 53) – internal	Input	Remote Access subnet, Admin subnet and all three internal VLANs	`Int-DNS` server
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.14: Firewall Rules for `Int-DNS` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Web traffic (ports 80, 443)	Output	`Squid` proxy server	External IP addresses
Squid proxy traffic (ports 3128/3129)	Input	Employee & Executive VLANs	`Squid` proxy server
DNS traffic (UDP/TCP 53) – external	Output	`Squid` proxy server	External IP address 8.8.8.8
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.15: Firewall Rules for `Squid` Machine

The rules outlined for the `SrvGW`, `Int-DNS`, `Squid` machines can be found instantiated within their `SrvGW.startup`, `Int-DNS.startup`, and `Squid.startup` files, respectively. These rules can seen in action by issuing the `iptables -L -nv` command on each machine.

1.2.5 Remote Access Zone Firewalls

The remote access zone contains the machines; `RemoteAccessGW` and the VPN endpoints `OpenVPN` – which uses port 1194 (TCP/UDP) for it's encrypted transport protocol – & `WireGuardVPN` – which uses port 51820 (UDP) for it's encrypted transport protocol. The VPN endpoints allow remote employees to securely communicate with internal MEME assets, therefore this zone must allow encrypted traffic to/from these machines which correspond to the port number they use for their encrypted transport protocol which is going to, or coming from, external IP addresses. Also, the `WireGuardVPN` endpoint should be able to access internal assets which are located within the server zone (i.e. `Int-DNS`), the DMZ (i.e. `Mail`), and the Employee VLAN, because this endpoint is used by remote employees. Whereas, the `OpenVPN` endpoint should be able to access the server zone (i.e. `Int-DNS`), the DMZ (i.e. `Mail`), and the Executive VLAN, because this endpoint is used by remote executives. The filter rules that this report recommends implementing on each of these machines to achieve this are in tables 1.16, 1.17, and 1.18.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Encrypted OpenVPN traffic (UDP/TCP ports 1194)	Forward	External IP addresses (VPN endpoints)	`OpenVPN` endpoint
Encrypted Wireguard traffic (UDP port 51280)	Forward	External IP addresses (VPN endpoints)	`WireGuardVPN` endpoint
IMAP traffic (port 993)	Forward	Remote Access zone subnet	`Mail` email server
DNS traffic (UDP/TCP ports 53)	Forward	Remote Access zone subnet	`Int-DNS` server
Unencrypted OpenVPN traffic	Forward	`OpenVPN` endpoint	Executive VLAN
Unencrypted Wireguard traffic	Forward	`WireGuardVPN` endpoint	Employee VLAN
SSH traffic (port 22)	Forward	Admin zone subnet	Server zone subnet
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.16: Firewall Rules for `RemoteAccessGW` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Encrypted OpenVPN traffic (UDP/TCP ports 1194)	Input	External IP addresses (VPN endpoints)	`OpenVPN` endpoint
Unencrypted OpenVPN traffic	Output	`OpenVPN` endpoint	Executive VLAN
IMAP traffic (port 993)	Forward	`OpenVPN` endpoint	`Mail` email server
DNS traffic (UDP/TCP 53) – internal	Output	`OpenVPN` endpoint	`Int-DNS` server
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.17: Firewall Rules for `OpenVPN` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
Encrypted Wireguard traffic (UDP port 51280)	Input	External IP addresses (VPN endpoints)	`WireGuardVPN` endpoint
Unencrypted Wireguard traffic	Output	`WireGuardVPN` endpoint	Employee VLAN
IMAP traffic (port 993)	Forward	`WireGuardVPN` endpoint	`Mail` email server
DNS traffic (UDP/TCP 53) – internal	Output	`WireGuardVPN` endpoint	`Int-DNS` server
SSH traffic (port 22)	Input & Output	Admin zone subnet	Admin zone subnet

Table 1.18: Firewall Rules for `WireGuardVPN` Machine

The rules outlined for the `RemoteAccessGW`, `OpenVPN`, `WireGuardVPN` machines can be found instantiated within their `RemoteAccessGW.startup`, `OpenVPN.startup`, and `WireGuardVPN.startup` files, respectively. These rules can seen in action by issuing the `iptables -L -nv` command on each machine.

1.2.6 Admin Zone Firewalls

The admin (administrator) zone makes up the Management Network, a security augmentation discussed in section 3.1. Assets within this zone are used to manage and administer the network infrastructure and security mechanisms (i.e. firewalls) that the MEME network is built on through SSH. To enable this remote administration (from within the admin network), SSH is running on each of the MEME's infrastructure and service machines, this is discussed further in section 3.1. The filter rules that this report recommends implementing on the machines within this zone are in tables 1.19 and 1.20.

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Forward **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
DNS traffic (UDP/TCP 53) – internal	Forward	Admin zone	`Int-DNS` server
SSH traffic (port 22)	Forward	Admin zone subnet	All zone subnets

Table 1.19: Firewall Rules for `AdminGW` Machine

Traffic	Action	Source	Destination
All – default policy	Block	All	All
All – stateful packet filtering	Input/Output **related** & **established** packets	All	All
All – stateful packet filtering	Drop **invalid** packets	All	All
DNS traffic (UDP/TCP 53) – internal	Output	`admin1` machine	`Int-DNS` server
SSH traffic (port 22)	Output	`admin1` administrator machine	All zone subnets

Table 1.20: Firewall Rules for `admin1` Machine

The rules outlined for the `AdminGW` and `admin1` machines can be found instantiated within their `AdminGW.startup` and `admin1.startup` files. These rules can seen in action by issuing the `iptables -L -nv` command on each machine.

1.2.7 Verification of Filtered Connectivity

It was previously demonstrated that the network architecture created provided connectivity when unfiltered. However, this is very insecure and, hence, packet filters (firewalls) were instantiated to limit this. In order to verify these filters will enable assets to connect with one another, when appropriate, this report uses several example scenarios which demonstrate filtered connectivity and uses terminal grabs from inside the Netkit model.

#1.1-3: Access to internal website on `Int-WWW` server from outside MEME hosts an internal web server (`Int-WWW`) which needs to be accessible to the outside world. Therefore, web traffic needs to be forwarded by the Perimeter Firewall `PerimF` to the `Int-WWW` server which is located within the DMZ. Firewall rules to provide this functionality have been instantiated on the `PerimF`, `DmzGW`, & `Int-WWW` machines, and have been previously outlined. To demonstrate that these rules are successful in providing this functionality terminal grabs from the external `Ext-Office` machine has been taken as a client on this machine made a request for the default web page on the `Int-WWW` server:

```
root@Ext-Office:~# wget http://10.10.2.2/
--2021-02-04 08:16:21--  http://10.10.2.2/
Connecting to 10.10.2.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html  100%[===================>]  10.45K  --.-KB/s    in 0s

2021-02-04 08:16:21 (70.6 MB/s) - 'index.html' saved [10701/10701]

root@Ext-Office:~# wget --no-check-certificate https://10.10.2.2/
--2021-02-04 08:19:07--  https://10.10.2.2/
Connecting to 10.10.2.2:443... connected.
WARNING: The certificate of '10.10.2.2' is not trusted.
WARNING: The certificate of '10.10.2.2' doesn't have a known issuer.
The certificate's owner does not match hostname '10.10.2.2'
```

```
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: ‘index.html.1’

index.html.1    100%[===================>]  10.45K  --.-KB/s    in 0.001s

2021-02-04 08:19:07 (13.5 MB/s) - ’index.html.1’ saved [10701/10701]

root@Int-WWW:~#
```

These two terminal grabs show a client on the `Ext-Office` machine is able to make both a HTTP and a HTTPS request to the `Int-WWW` web server (located at `10.10.2.2`) and successfully download MEME's default web page ("index.html"). In addition to this, MEME's web server `Int-WWW` needs to be able to send web requests to other web services on the Internet to provide dynamic functionality to MEME's website. To demonstrate that this network allows for this functionality another terminal grab has been taken, this time from `Int-WWW` as it accesses an external web service `Ext-WWW` machine.

```
root@Int-WWW:~# wget http://201.224.19.7/
--2021-02-04 08:20:20--  http://201.224.19.7/
Connecting to 201.224.19.7:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: ‘index.html’

index.html   100%[==================>]  10.45K  --.-KB/s    in 0s

2021-02-04 08:20:20 (44.0 MB/s) - ‘index.html’ saved [10701/10701]

root@Int-WWW:~# wget --no-check-certificate https://201.224.19.7/
--2021-02-04 08:20:35--  https://201.224.19.7/
Connecting to 201.224.19.7:443... connected.
WARNING: The certificate of ’201.224.19.7’ is not trusted.
WARNING: The certificate of ’201.224.19.7’ doesn’t have a known issuer.
The certificate’s owner does not match hostname ’201.224.19.7’
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: ‘index.html.1’

index.html.1  100%[==================>]  10.45K  --.-KB/s    in 0.004s

2021-02-04 08:20:35 (2.78 MB/s) - ‘index.html.1’ saved [10701/10701]

root@Int-WWW:~#
```

These two terminal grabs show that MEME's web server `Int-WWW` is able to make both HTTP and HTTPS requests to the `Ext-WWW` web server (located at `201.224.19.7`) and successfully download it's default web page (i.e. access an external web service). Aside from web traffic, MEME's web server `Int-WWW` needs to be able to send external DNS requests to DNS servers to locate other web servers on the Internet and produce dynamic content. This functionality is demonstrated in the terminal grab below as `Int-WWW` sends a DNS request to the `Ext-DNS`.

```
root@Int-WWW:~# host -t a this.test.com 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
```

```
this.test.com has address 201.224.19.7
root@Int-WWW:~#
```

This terminal grab shows that `Int-WWW` is able to make a DNS request to the external DNS server `Ext-DNS` (located at `8.8.8.8`) and successfully get back a DNS response back.

#1.4: Access to external website on `Ext-WWW` server from Employee & Executive VLANs through `Squid proxy` On site employees at MEME's Coventry offices need to be able to access the outside Internet in order to efficiently complete their work. However, this business functionality comes with a lot of risks to MEME's internal network assets, thus a proxy server is used. This proxy server is on the `Squid` machine and employees need to go through this proxy before they can reach the Internet. The specifics of this Squid proxy server are discussed in depth in section 2.3 as it is a security augmentation this report recommends MEME implement to improve it's security posture. However, firstly it is important to demonstrate that the network configuration in this Netkit model works appropriately by providing Internet connectivity to machines within the Employee and Executive VLANs. To demonstrate this an employee (on `Staff-1`) and an executive (on `Staff-2`) are shown being able to download the default web page of the `Ext-WWW` web server.

```
# accessing external Internet from Employee VLAN
root@Staff-1:~# export http_proxy="10.10.3.3:3128"
root@Staff-1:~# wget http://201.224.19.7/
--2021-02-08 14:48:49--  http://201.224.19.7/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html      100%[================>]  10.45K  --.-KB/s    in 0.005s

2021-02-08 14:48:49 (2.03 MB/s) - 'index.html' saved [10701/10701]

root@Staff-1:~#

# accessing external Internet from Executive VLAN
root@Staff-2:~# export http_proxy="10.10.3.3:3128"
root@Staff-2:~# wget http://this.test.com
--2021-02-08 14:48:19--  http://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html      100%[================>]  10.45K  --.-KB/s    in 0.01s

2021-02-08 14:48:19 (894 KB/s) - 'index.html' saved [10701/10701]

root@Staff-2:~#
```

The first terminal grab shows `Staff-1` setting it's `http_proxy` environment variable to `10.10.3.3` (the `Squid` proxy server) with the port number `3128` (the port Squid proxy is listening for HTTP traffic on. Then `Staff-1` uses the `wget` utility to download the default web page of `Ext-WWW` (located at IP address `201.224.19.7`. The second terminal

grab is from `Staff-2` and it shows an Executive member of staff using the same Squid proxy server to download the same default web page of `Ext-WWW`. However, this time the domain name is give (`this.test.com`) to demonstrate that the Squid proxy has DNS capabilities.

#1.5: Ability to send/receive email via SMTP through `Mail` server MEME's email server `Mail` (located at `10.10.2.3` in the DMZ) needs to be able to both send and receive SMTP traffic so that it can provide a inbound and outbound email service for the organisation. To provide this functionality, firewalls on machines `PerimF`, `DmzGW`, and `Mail` need to be configured to allow traffic to/from external IP addresses (external email servers) on ports 25 and 587. The necessary rules to allow this filtered traffic are defined above and are demonstrated in the terminal grab below.

```
root@Ext-Office:~# nc -v 10.10.2.3 25
Connection to 10.10.2.3 25 port [tcp/smtp] succeeded!
^C
root@Ext-Office:~# nc -v 10.10.2.3 587
Connection to 10.10.2.3 587 port [tcp/submission] succeeded!
^C
root@Ext-Office:~#
```

As the terminal grab shows `Ext-Office` is able to connect to ports 25 and 587 on MEME's email server `Mail` (located at `10.10.2.3`). However, as previously stated, `Mail` also needs to be able to send traffic to other email servers on the external Internet. To demonstrate this `Ext-Office` was made to listen on ports 25 and 587, and `Mail` connected to those ports.

```
# setting up netcat listeners on Ext-Office
root@Ext-Office:~# service exim4 stop
root@Ext-Office:~# nc -lvp 25 &
[1] 2019
root@Ext-Office:~# Listening on 0.0.0.0 25

root@Ext-Office:~# nc -lvp 587 &
[2] 2020
root@Ext-Office:~# Listening on 0.0.0.0 587

# connecting to Ext-Office on ports 25 and 587 (SMTP)
root@Mail:~# nc -v 22.39.224.18 25
Connection to 22.39.224.18 25 port [tcp/smtp] succeeded!
^C
root@Mail:~# nc -v 22.39.224.18 587
Connection to 22.39.224.18 587 port [tcp/submission] succeeded!
^C
root@Mail:~#
```

This terminal grab shows two netcat listeners being setup on `Ext-Office` (located at `22.39.224.18`), which is masquerading as an email server by listening for connections on ports 25 and 587, and `Mail` being able to send traffic to these two ports. Note, by default machines in this model have a mail service (`exim4`) running so this needed to be stopped before `Ext-Office` could listen on port 25.

#1.6: Ability to send/receive email via IMAP through `Mail` server from Remote Access Zone and the Employee & Executive VLANs Aside from sending/receiving SMTP traffic, MEME's `Mail` server needs to be able to receive IMAP traffic (port 993) from the Remote Access Zone and the Employee & Executive VLANs so that employees

can access their email inboxes. As such, the network-based firewall rules on machines `PerimF` (which routes internal traffic), `DmzGW`, `IntGW`, `RemoteAccessGW`, and the host-based firewall rules on all machines within the Remote Access Zone, Employee/Executive VLANs, and `Mail` need to allow this connectivity. These rules have been previously outlined above and are demonstrated working in the screen grabs below.

```
# sending IMAP from Employee VLAN
root@Staff-1:~# nc -v 10.10.2.3 993
Connection to 10.10.2.3 993 port [tcp/imaps] succeeded!
^C
root@Staff-1:~#

# sending IMAP from Executive VLAN
root@Staff-2:~# nc -v 10.10.2.3 993
Connection to 10.10.2.3 993 port [tcp/imaps] succeeded!
^C
root@Staff-2:~#

# sending IMAP from Remote Access zone
root@OpenVPN:~# nc -v 10.10.2.3 993
Connection to 10.10.2.3 993 port [tcp/imaps] succeeded!
^C
root@OpenVPN:~#
```

These three terminal grabs show that machines within the Remote Access Zone (i.e. `OpenVPN`) and the Employee (i.e. `Staff-1`) & Executive (i.e. `Staff-2`) VLANs are able to send IMAP traffic (port 993) to the `Mail` server. The email server replies to these machines due to stateful packet filtering rules. Therefore, employees (generic, executive, and remote) can access their email inboxes and send/download email from MEME's `Mail` server.

#1.7: Access to internal DNS via `Int-DNS` server for employees and executives
MEME's employees (generic, executive, and remote) all need to be able to send internal DNS requests to MEME's `Int-DNS` server so that they can utilise the organisation's internal DNS service. To do this network-based firewalls on machines `PerimF` (which routes internal traffic), `SrvGW`, `IntGW`, `RemoteAccessGW`, and the host-based firewall rules on all machines within the Remote Access Zone, Employee/Executive VLANs, and `Int-DNS` need to allow this connectivity. These rules have been previously outlined above and are demonstrated working in the screen grabs below.

```
# DNS request from Employee VLAN
root@Staff-1:~# host -t a www.meme.cyber.test 10.10.3.2
Using domain server:
Name: 10.10.3.2
Address: 10.10.3.2#53
Aliases:

www.meme.cyber.test has address 10.10.2.2
root@Staff-1:~#

# DNS request from Executive VLAN
root@Staff-2:~# host -t a www.meme.cyber.test 10.10.3.2
Using domain server:
Name: 10.10.3.2
Address: 10.10.3.2#53
Aliases:
```

```
www.meme.cyber.test has address 10.10.2.2
root@Staff-2:~#

# DNS request from Remote Access zone
root@OpenVPN:~# host -t a www.meme.cyber.test 10.10.3.2
Using domain server:
Name: 10.10.3.2
Address: 10.10.3.2#53
Aliases:

www.meme.cyber.test has address 10.10.2.2
root@OpenVPN:~#
```

These three terminal grabs show that machines within the Remote Access Zone (i.e. `OpenVPN`) and the Employee (i.e. `Staff-1`) & Executive (i.e. `Staff-2`) VLANs are able to send DNS traffic (TCP/UDP port 53) to the `Int-DNS` server and request the IP address of MEME assets. The DNS server replies to these machines due to stateful packet filtering rules. Therefore, employees (generic, executive, and remote) can use MEME's internal DNS service.

#1.8: Access to Wireguard VPN endpoint `WireGuardVPN` from outside & then assets within the Employee VLAN Given the recent pandemic there has been an increased need for secure remote access to internal MEME assets for employees who are working from home. As such, this Netkit model includes a Wireguard VPN endpoint located at `10.10.4.3` within the Remote Access zone subnet. This endpoint is used by remote employees to securely access internal assets within MEME's internal Employee VLAN. Adding to this, the Wireguard VPN endpoint can be used to provide remote collaborators secure access to MEME assets. For instance, MEME works closely with an Australian collaborator on a range of projects and, as such, they may require secure access to assets within MEME's Employee VLAN. To demonstrate this functionality within the Netkit model, this Australian collaborator has been added to the external network (representing the larger Internet) as shown in figure 1.4.

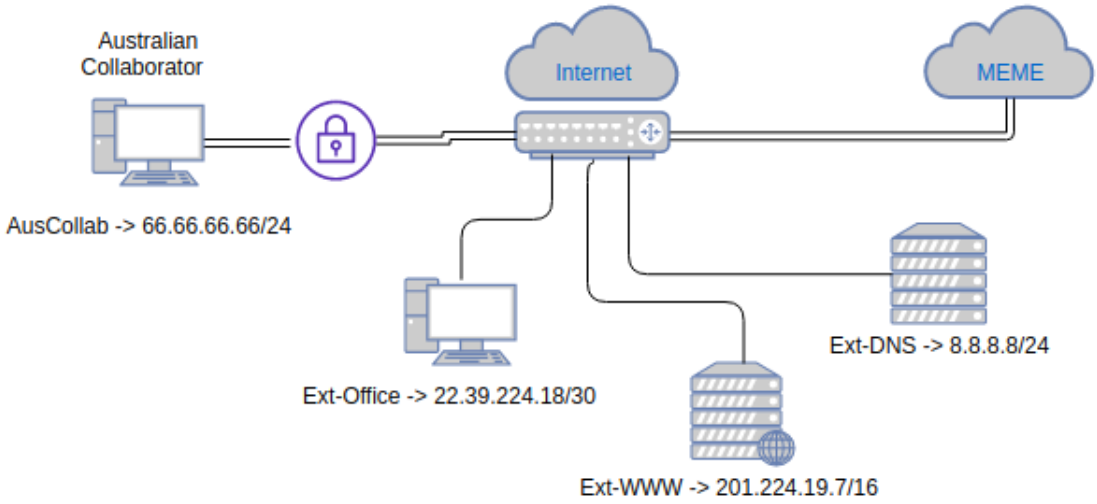

Figure 1.4: External Network in Netkit Model

As shown the remote Australian collaborator is connected to the external `Internet` machine and sits behind a Wireguard VPN rogue warrior endpoint which can connect to MEME's `WireGuardVPN` endpoint for secure access to MEME's internal assets. This is just like how a MEME employee would connect remotely to securely access assets within MEME's Employee VLAN and, thus, firewall configurations are the same. Therefore, to provide remote access to both employees and collaborators to the Employee VLAN, network-based firewalls need to be instantiated on machines `PerimF`, `RemoteAccessGW`, `IntGW`, and host-based firewall rules on the `WireGuardVPN` machine and machines within the Employee VLAN that MEME wants to provide remote access to. These configuration details were shown above and a demonstration of this functionality is shown in the terminal grabs below.

```
# remotely accessing WireGuardVPN endpoint from remote employee/collaborator
root@AusCollab:~# nc -vu vpn2.meme.cyber.test 51820
Connection to vpn2.meme.cyber.test (137.205.89.220) 51820 port [udp/*]
    succeeded!
^C
root@AusCollab:~#

# setting up netcat listener on machine in Employee VLAN and receiving connection
root@Staff-1:~# nc -lvp 1234
Listening on 0.0.0.0 1234
Connection received on vpn2.meme.cyber.test 42000
root@Staff-1:~#

# access machine listening from within Employee VLAN through WireGuardVPN endpoint
root@WireGuardVPN:~# nc -v staff-1.meme.cyber.test 1234
Connection to staff-1.meme.cyber.test (10.100.1.2) 1234 port [tcp/*]
    succeeded!
^C
root@WireGuardVPN:~#
```

These terminal grabs show a remote user (i.e. `AusCollab`) is able to connect to the `WireGuardVPN` endpoint within MEME's Remote Access zone via UDP port 51820 (the port Wireguard uses to send encypted data between endpoints). From here, a netcat listener was setup on an Employee VLAN machine (i.e. `Staff-1`) to simulate an employee asset on the Employee VLAN. Then `WireGuardVPN` made a connection to this simulated

employee asset to show how a remote employee/collaborator could make a secure connection to `WireGuardVPN` and then access assets within the Employee VLAN. Note, this setup was created after DNS and DNAT (section 2.2) was implemented for MEME's network and, hence, IP addresses a different from other examples in this section.

#1.9: Access to OpenVPN endpoint `OpenVPN` from outside & then assets within Executive VLAN In addition to providing VPN access to the internal network through Wireguard, MEME also has an OpenVPN endpoint which came with the starter pack. With the Wireguard VPN endpoint being used for regular employee access, as demonstrated, this Netkit model uses the OpenVPN endpoint as a VPN endpoint for Executive staff to securely access assets within the internal Executive VLAN. Given the recent pandemic, and need for more remote access, this report recommends instantiating multiple VPN endpoints to cope with the workload and, hence, has created a Remote Access zone of trust which hosts two VPN endpoints. For this OpenVPN endpoint to allow remote executives to securely access assets within the Executive VLAN, there needs to be network-based firewalls on machines `PerimF`, `RemoteAccessGW`, `IntGW`, and host-based firewall rules on `OpenVPN` and machines within the Executive VLAN. These configuration details were shown above and a demonstration of this functionality is shown in the terminal grabs below.

```
# remotely accessing OpenVPN endpoint from outside MEME
root@Ext-Office:~# nc -v 10.10.4.2 1194
Connection to 10.10.4.2 1194 port [tcp/openvpn] succeeded!
^C
root@Ext-Office:~#

# setup netcat listener on machine in Executive VLAN and receiving connection
root@Staff-2:~# nc -lvp 1234
Listening on 0.0.0.0 1234
Connection received on vpn1.meme.cyber.test 56632
root@Staff-2:~#

# access machine listening from within Executive VLAN through OpenVPN endpoint
root@OpenVPN:~# nc -v 10.10.11.2 1234
Connection to 10.10.11.2 1234 port [tcp/*] succeeded!
^C
root@OpenVPN:~#
```

The first terminal grab shows that a remote user (i.e. `Ext-Office`) can access the `OpenVPN` endpoint (located at `10.10.4.2`) on port 1194 (the port OpenVPN uses to send encrypted data between endpoints). Following from this, a netcat listener was setup on a machine within the Executive VLAN (i.e. `Staff-2`) to act as an Executive asset located on the Executive VLAN, and it shows a connection from `OpenVPN` was established (i.e. `vpn1.meme.cyber.test`. Finally, the asset within the Executive VLAN is accessed through the `OpenVPN` endpoint to show that a remote executive can access assets within the Executive VLAN from here – this was the connection that was received in the second terminal grab.

#1.10: Access to `LDAP` server for authentication from Employee & Executive VLANs One of MEME's pre-existing assets was an LDAP server which was used to authenticate Employees and Executives. It is a delicate asset that holds valuable company information which needs to be protected, hence is located within the Secured VLAN, but also needs to be accessible from both the Employee & Executive VLANs so

that employees & executives can authenticate themselves. As such, the network-based firewall rules on machines `PerimF` (which routes internal traffic) and `IntGW`, as well as the host-based firewall rules on the `LDAP` machine iteself and machines situated within the Employee/Executive VLANs need to allow this connectivity. These rules have been previously outlined above and are demonstrated working in the screen grabs below.

```
# sending LDAP traffic to LDAP machine from Employee VLAN machine
root@Staff-1:~# nc -v 10.10.33.2 389
Connection to 10.10.33.2 389 port [tcp/ldap] succeeded!
^C
root@Staff-1:~# nc -v 10.10.33.2 636
Connection to 10.10.33.2 636 port [tcp/ldaps] succeeded!
^C
root@Staff-1:~#

# sending LDAP traffic to LDAP machine from Executive VLAN machine
root@Staff-2:~# nc -v 10.10.33.2 389
Connection to 10.10.33.2 389 port [tcp/ldap] succeeded!
^C
root@Staff-2:~# nc -v 10.10.33.2 636
Connection to 10.10.33.2 636 port [tcp/ldaps] succeeded!
^C
root@Staff-2:~#
```

These two terminal grabs show that machines within the Employee VLAN (i.e. `Staff-1`) & Executive VLAN (i.e. `Staff-2`) are able to send LDAP traffic (ports 389/636) to the `LDAP` server. The server replies to these machines due to stateful packet filtering rules. Therefore, all internal employees can authenticate themselves via the `LDAP` server.

#1.11: Access to `DHCP` server for dynamic host configuration from Employee & Executive VLANs MEME is a medium to large size organisation with plans to expand it's workforce over the next three years. As such, with a growth in employees and employees coming back into offices post-COVID this report has decided to include a `DHCP` server within the Netkit model. This DHCP server can be used to dynamically configure hosts withn both the Employee & Executive VLANs. To do this the network-based firewall rules on machines `IntGW` (which routes internal VLAN traffic) and the host-based firewall rules on `DHCP`, `Staff-1`, `Staff-2`, & `Staff-3` need to be configured to provide this connectivity. These rules have been previously outlined above and are demonstrated working in the screen grabs below.

```
# sending DHCP traffic to DHCP server machine from Employee VLAN machine
root@Staff-1:~# nc -vu 10.10.33.3 67
Connection to 10.10.33.3 67 port [udp/bootps] succeeded!
^C
root@Staff-1:~#

# sending DHCP traffic to DHCP server machine from Executive VLAN machine
root@Staff-2:~# nc -vu 10.10.33.3 67
Connection to 10.10.33.3 67 port [udp/bootps] succeeded!
^C
root@Staff-2:~#
```

The first terminal grab shows a machine located within the Employee VLAN `10.100.0.0/16` (i.e. `Staff-1`) being able to successful send DHCP traffic (UDP port 67) to the `DHCP` server (`10.10.33.3`), and this server responding back. The second terminal grab shows the same but using a machine located within the Executive VLAN `10.10.11.0/24` (i.e.

`Staff-2`). This functionality shows that thes internal employee/executive machines have the ability to communicate with the `DHCP` server so that dynamic host configuration can be setup.

#1.12: Remote administration through SSH via the Admin Zone As part of it's secure network architecture this Netkit model includes a Admin zone which this report deemed necessary to implement in order to efficiently manage MEME's network infrastructure. The specialist, on-site system administration and security operators will utilise this network to perform their roles. This zone is further discussed in section 3.1 when this report outlines how a Management Network has been implemented within this Netkit model as a augmentation to enhance MEME's security posture. However, in order to implement this management network there must be connectivity between each of MEME's infrastructure assets and administrator assets within the Admin zone, in particular SSH connectivity. The network-based firewall rules on machines `PerimF` (which routes internal traffic), `RemoteAccessGW`, `SrvGW`, `DmzGW`, `IntGW`, `AdminGW`, as well as the host-based firewall rules on all of MEME's infrastructure assets which includes `OpenVPN`, `WireGuardVPN`, `Mail`, `Int-WWW`, `Int-DNS`, `Squid`, `LDAP`, `DHCP`, in addition to the gateway machines already mentioned provide this connectivity. These rules have been previously outlined above and are demonstrated working in the screen grabs below.

```
# sending SSH traffic to PerimF machine from admin1 machine
root@PerimF:~# root@Staff-2:~# nc -lvp 22

root@admin1:~# root@Staff-2:~# nc -v 192.168.1.1 22
Connection to 192.168.1.1 22 port [tcp/ssh] succeeded!
^C
root@admin1:~#


# sending SSH traffic to Int-DNS machine from admin1 machine
root@Int-DNS:~# root@Staff-2:~# nc -lvp 22

root@admin1:~# root@Staff-2:~# nc -v 10.10.1.1 22
Connection to 10.10.1.1 22 port [tcp/ssh] succeeded!
^C
root@admin1:~#


# sending SSH traffic to IntGW machine from admin1 machine
root@IntGW:~# root@Staff-2:~# nc -lvp 22

root@admin1:~# root@Staff-2:~# nc -v 10.10.3.2 22
Connection to 10.10.3.2 22 port [tcp/ssh] succeeded!
^C
root@admin1:~#
```

Each of these terminal grabs shows an infrastructure machine (i.e. `PerimF`, `Int-DNS`, `IntGW`) starting a netcat listener on port 22 (mimicking an SSH server) and then `admin1` connecting to this port (mimicking an SSH connection). The actual SSH connection and remote administration are discussed further in section 3.1 when the Management Network this Netkit model implements are highlighted. Note, a representative sample of the machines which SSH traffic is allowed to/from are shown here, but all machines mentioned have this functionality.

1.2.8 Future Work

The network architecture provided by this Netkit model has all of the structural features which a medium to large size organisation like MEME would require in order to implement firewalls and security augmentations to increase the organisation's security posture. However, despite the necessary network architecture being implemented there are a few features that would need to be actualised when this Netkit model is implemented in the real world.

A fully-functioning DHCP server This Netkit model uses a mock DHCP server (`DHCP`) to simulate the necessary filtered access are fully-functioning DHCP server would require. By doing this, when this model is implemented in the real-world all that would need to be done is configuring of the DHCP server and the network architecture (and firewalls) can remain untouched. As mentioned previously, with MEME planning to expand, the configuration of hosts dynamically will be beneficial to the organisation in the future and, therefore, this architecture required for this has been implemented in this Netkit model.

A LDAP server Provided in the "starter" files for this Netkit model was a mock LDAP server (`LDAP`) which this model used to simulate the security architecture a fully-functioning LDAP server would require to authenticate employees and executives in their respective VLANs. Therefore, when this model is implemented in the real-world all that would need to be done is configuring of the LDAP server and the network architecture (and firewalls) can remain untouched.

A mail server Provided in the "starter" files for this Netkit model was a mock mail server (`Mail`) which this model used to simulate the security architecture a fully-functioning email server would require to send email using SMTP and allow employees, executives, and remote employees to access their email using IMAP. Therefore, when this model is implemented in the real-world all that would need to be done is configuring of the Mail server and the network architecture (and firewalls) can remain untouched.

A WireGuardVPN server This Netkit model created a operational Wireguard VPN server to demonstrate how remote employees/collaborators could securely access assets within the Employee VLAN, as well as accessing internal MEME services `Mail` and `Int-DNS` . However, the `WireGuardVPN` is not a fully-functioning VPN solution and would need to be configured appropriately whne this model is is implemented in the real-world. As such, this Netkit model just provides the secure network architecture (and firewalls) required to implement a Wireguard VPN endpoint in MEME's network and leaves the implementation as future work.

A OpenVPN server Provided in the "starter" files for this Netkit model was a mock OpenVPN server (`OpenVPN`) which this model used to simulate the security architecture a fully-functioning OpenVPN server would require to allow remote executives to securely interact with assets on MEME's Executive VLAN, as well as accessing internal MEME services `Mail` and `Int-DNS` . Therefore, when this model is implemented in the real-world all that would need to be done is configuring of the OpenVPN server and the network architecture (and firewalls) can remain untouched.

1.3 Phase 1 Summary

Reference & Claim – section 1.2.7	Evidence
Ref: #1.1 – External clients connected to the `Internet` machine are able to access MEME's `Int-WWW` web server	Firewall rules on `PerimF`, `DmzGW`, & `Int-WWW` and terminal grabs in section 1.2.7
Ref: #1.2 – `Int-WWW` can access external web services (i.e. `Ext-WWW`) to provide dynamic web content	Firewall rules on `PerimF`, `DmzGW`, & `Int-WWW` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.3 – `Int-WWW` can access external DNS servers (i.e. `Ext-DNS`) for host name look-ups	Firewall rules on `PerimF`, `DmzGW`, & `Int-WWW` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.4 – Machines located within the Employee & Executive VLANs can use the `Squid` proxy machine to access external web content (i.e. `Ext-WWW`)	Firewall rules on `PerimF`, `SrvGW`, `Squid`, `IntGW`, `Staff-1`, `Staff-2`, & `Staff-3` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.5 – MEME's `Mail` server can both send/receive SMTP traffic (ports 25/587) to/from external email servers	Firewall rules on `PerimF`, `DmzGW`, & `Mail` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.6 – MEME's `Mail` server can receive IMAP traffic (port 993) from the Remote Access Zone subnet and the Employee & Executive VLANs, and then use stateful packet filtering rules to reply	Firewall rules on `PerimF`, `DmzGW`, `Mail`, `RemoteAccessGW`, `OpenVPN`, `WireGuardVPN`, `Staff-1`, `Staff-2`, & `Staff-3` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.7 – MEME's `Int-DNS` server can receive DNS traffic (TCP/UDP 53) from the Remote Access Zone subnet and the Employee & Executive VLANs, and then use stateful packet filtering rules to reply	Firewall rules on `PerimF`, `SrvGW`, `Int-DNS`, `RemoteAccessGW`, `OpenVPN`, `WireGuardVPN`, `Staff-1`, `Staff-2`, & `Staff-3` (section 1.2) and terminal grabs in section 1.2.7
Ref: #1.8 – Remote employees and collaborator (i.e. `AusCollab`) can remotely access assets within the Employee VLAN in a secure manner via `WireGuardVPN`	Firewall rules on `PerimF`, `RemoteAccessGW`, `WireGuardVPN`, `Staff-1`, & `Staff-3`
Ref: #1.9 – Remote executives can remotely access assets within the Executive VLAN in a secure manner via `OpenVPN`	Firewall rules on `PerimF`, `RemoteAccessGW`, `OpenVPN`, & `Staff-2` and terminal grabs in section 1.2.7
Ref: #1.6 & **#1.7** – Remote employees, collaborators, and executives can access the internal MEME services `Int-DNS` & `Mail` via `WireGuardVPN` & `OpenVPN`	Firewall rules on `PerimF`, `RemoteAccessGW`, `WireGuardVPN`, `OpenVPN`, `DmzGW`, `Mail`, `SrvGW`, `Int-DNS`, & `Staff-3` and terminal grabs in section 1.2.7
Ref: #1.10 – Employees and executives working within their respective VLANs can access the `LDAP` server in the Secured VLAN to authenticate themselves	Firewall rules on `PerimF`, `IntGW`, `LDAP`, `Staff-1`, `Staff-2`, & `Staff-3` and terminal grabs in section 1.2.7
Ref: #1.11 – Employees and executives can have their machines dynamically configured via access to the `DHCP` server on the Secured VLAN	Firewall rules on `PerimF`, `IntGW`, `DHCP`, `Staff-1`, `Staff-2`, & `Staff-3` and terminal grabs in section 1.2.7
Ref: #1.12 – The onsite system administration and security team can remotely administer MEME's infrastructure assets through SSH from the Admin zone subnet	Firewall rules on all gateway machines and `PerimF`, `Int-WWW`, `WireGuardVPN`, `OpenVPN`, `Mail`, `Squid`, `Int-DNS`, & `admin1` and terminal grabs in section 1.2.7

Table 1.21: Phase 1 Work Summary Table

2 Phase 2

2.1 Verification of Connectivity Being Prevented

Now that the necessary connectivity for MEME's network to function appropriately has been verified, this report looks to verify that connectivity is prevented between inappropriate clients and the services that they should not be able to access. This is important because the firewalls that have been instantiated need to prevent unauthorised access as much as they need to allow authorised access. To demonstrate that the firewalls which have been instantiated within this Netkit model, and discussed within this report, also prevent unauthorised access, several scenarios have been used. These scenarios are listed below.

#2.1: Prevent access to internal MEME assets from the outside It is very important that external machines cannot access internal assets within the MEME network which are not meant to be accessed. For instance, assets within the Internal zone, Server zone, and Admin zone should not be accessible to the outside. This is demonstrated in the terminal grabs below.

```
# outside machine trying to ping Employee VLAN machine
root@Ext-Office:~# ping 10.100.1.2
PING 10.100.1.2 (10.100.1.2) 56(84) bytes of data.
^C
--- 10.100.1.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1016ms

# outside machine trying to ping Executive VLAN machine
root@Ext-Office:~# ping -c2 10.100.1.2
PING 10.100.1.2 (10.100.1.2) 56(84) bytes of data.

--- 10.100.1.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1078ms

# outside machine trying to ping Admin machine
root@Ext-Office:~# ping -c2 10.10.5.201
PING 10.10.5.201 (10.10.5.201) 56(84) bytes of data.

--- 10.10.5.201 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1103ms

root@Ext-Office:~#
```

This first set of terminal grabs shows a ping request to assets (machines) within the internal Employee and Executive VLANs, as well as the `admin1` machine. As is shown, each of these machines are unreachable from the outside. This is because the the network firewalls `PerimF`, `IntGW`, and `AdminGW`, as well as the host-based firewalls on these machines, have been made to discard ICMP ping packets. In addition to this, these firewalls also prevent any packets reaching these firewalls through their default deny policies, whereby if a rule doesn't exist to specifically allow that packet access, then it is dropped. This functionality is shown in the second set of terminal grabs where a random port `1234` has been opened on `Staff-1` and an outside machine (i.e. `Ext-Office` tries to connect to this port, but is unable to.

```
# outside machine trying to ping Employee VLAN machine with open port
root@Staff-1:~# nc -lvp 1234
Listening on 0.0.0.0 1234

root@Ext-Office:~# nc -v 10.100.1.2 1234
^C
root@Ext-Office:~#
```

This is a brief sample of how default connectivity is prevented. By having a default deny policy on all machines for all firewall chains (i.e. FORWARD, INPUT, & OUTPUT) means that unless there is a specific rule which allows a packet to enter one of these zones/machines, then the packet is dropped. These rules can be found in section 1.2. In general, the machines above won't have ports open, however `Int-DNS` and `Squid` (assets within the Server zone will). Therefore, to show that these machine only offer their services to internal assets several more terminal grabs have been taken.

```
# outside machine trying to access internal DNS
root@Ext-Office:~# nc -v 10.10.3.2 53
^C
root@Ext-Office:~#

# outside machine trying to access internal Squid proxy
root@Ext-Office:~# nc -v 10.10.3.3 3128
^C
root@Ext-Office:~# nc -v 10.10.3.3 3129
^C
root@Ext-Office:~#
```

These terminal grabs show that an external machine cannot access port 53 on `Int-DNS` to make DNS requests (even though it is open), nor can it access port 3128 or 3129 on `Squid` to use it as a proxy (even though it is also open). Thus, the default, is for no connectivity from external machines to internal machines as provided by MEME's firewalls.

#2.2: Prevent unauthorised access to protected internal MEME assets from the inside Aside from preventing external machines from connecting to internal assets, it is also important to only allow connections between internal machines when it is necessary. This is done so that if a system is breached then the damage this will potentially cause is limited (i.e. preventing the lateral movement of an attacker). For instance, assets within the DMZ should not be able to access assets within any other zone. This is demonstrated in the terminal grabs below.

```
#### web server trying to access simulated LDAP server
# LDAP set to listen (simulate real LDAP ports used)
root@LDAP:~# while true; do nc -lvp 389;done &
  while true; do nc -ulvp 389;done &
  while true; do nc -lvp 636;done &
  while true; do nc -ulvp 636;done &
[1] 1777
[2] 1778
[3] 1779
[4] 1780
root@LDAP:~#

# outside connection = denied)
root@Ext-Office:~# nc -v 10.10.33.2 389
^C
root@Ext-Office:~# nc -v 10.10.33.2 636
```

```
^C
root@Ext-Office:~#

# unauthorised internal connection = denied)
# mail server trying to access executive VLAN
root@Staff-2:~# nc -lvp 1234
Listening on 0.0.0.0 1234

root@Mail:~# nc -v 10.10.11.2 1234
^C
root@Mail:~#
```

As is shown, firstly an LDAP server is simulated on `LDAP` (open ports TCP/UDP 389 & 636) and the external machine `Ext-Office` tries to connect. However, this is unsuccessful as this connectivity is denied. Next, a port is opened on `Staff-2` (a machine on the Executive VLAN) and then the `Mail` server tries to connect to this port. However, yet again this connectivity is denied by default. Thus, unauthorised access to protected internal MEME assets from machines inside MEME's network is acheived. That said, the `Mail` server is still accessible by employees because firewalls have been configured to allow connections coming from the Executive/Employee VLANs & the Remote Access zone which are destined for port 993 on the `Mail` server. Then, through stateful packet filtering, the `Mail` server can reply to these requests. These configurations were outlined in the firewall rules in section 1.2. In addition to this, MEME employees should not be able to reach assets within the Admin zone as these assets are for administrators only. To demonstrate this ...

```
# Employee VLAN machine trying to access admin machine
root@admin1:~# nc -lvp 22
Listening on 0.0.0.0 22

root@Staff-1:~# nc -v 10.10.5.2 22
^C
root@Staff-1:~#
```

#2.3: Prevent unauthorised access to external assets from within MEME Another network security consideration that MEME needs to address is unauthorised access to external assets from within MEME's network. It is important that any connection made to the external network is accounted for and unauthorised connections are blocked by default. For instance, an attacker who breaches MEME's network will look to exfiltrate data and, as such, internal to external network connections must also be monitored. This report has created a network architecture whereby all employees must go through a proxy server before they can reach the Internet. This has been done to allow MEME to use the additional security controls that a proxy server provides (i.e. whitelisting, blacklisting, packet inspection, etc.) and prevent un-monitored Internet connections by employees. The specifics of the Squid proxy are discussed in section 2.3, but by default employees within the Internal network (i.e. Executive/Employee VLANs) should not be able to access the Internet directly. This is demonstrated in the terminal grabs below.

```
# Employee VLAN machine trying to access through Squid proxy = success
root@Staff-1:~# export http_proxy="10.10.3.3:3128"
root@Staff-1:~# wget http://201.224.19.7
--2021-02-08 16:44:28--  http://201.224.19.7/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
```

```
Saving to: 'index.html'

index.html      100%[==============>]  10.45K  --.-KB/s    in 0.004s

2021-02-08 16:44:28 (2.30 MB/s) - 'index.html' saved [10701/10701]

# Employee VLAN machine trying to access directly = denied
root@Staff-1:~# export http_proxy=""
root@Staff-1:~# wget http://201.224.19.7
--2021-02-08 16:44:39--  http://201.224.19.7/
Connecting to 201.224.19.7:80... ^C
root@Staff-1:~#

# Executive VLAN machine trying to access web through Squid proxy = success
root@Staff-2:~# export http_proxy="10.10.3.3:3128"
root@Staff-2:~# wget http://201.224.19.7
--2021-02-08 16:50:13--  http://201.224.19.7/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html      100%[==============>]  10.45K  --.-KB/s    in 0s

2021-02-08 16:50:13 (66.8 MB/s) - 'index.html' saved [10701/10701]

# Employee VLAN machine trying to access web directly = denied
root@Staff-2:~# export http_proxy=""
root@Staff-2:~# wget http://201.224.19.7
--2021-02-08 16:50:20--  http://201.224.19.7/
Connecting to 201.224.19.7:80... ^C
root@Staff-2:~#
```

These two terminal grabs from `Staff-1` and `Staff-2` show that when either of these machines (representing users on the Employee and Executive VLANs) uses a proxy (set with `http_proxy`) then they can access the web server at `201.224.19.7`, but when no proxy is used they cannot connect to this web server. Aside from employees, internal servers which are providing services to MEME assets should not be able to access the Internet. For instance, `Int-DNS` and the Perimeter Firewall machine `PerimF` should not be able to make external Internet connections. This is shown in below.

```
# internal DNS machine cannot connect to web server
root@Int-DNS:~# wget http://201.224.19.7/
--2021-02-08 16:57:09--  http://201.224.19.7/
Connecting to 201.224.19.7:80... ^C
root@Int-DNS:~#

# perimeter firewall machine cannot connect to web server
root@PerimF:~# wget http://201.224.19.7/
--2021-02-08 16:54:42--  http://201.224.19.7/
Connecting to 201.224.19.7:80... ^C
root@PerimF:~#
```

#2.4: Prevent remote administration by unauthorised parties from the within MEME In this Netkit model remote administration is done through SSH (port 22). This remote administration (i.e. SSH access) should only be possible for machines from within the Admin zone subnet. This subnet is where the onsite system administration

and security operators conduct their work from and is discussed further in section 3.1. It was previously demonstrated that machines within this subnet (i.e. `admin1`) could SSH into other machines to perform their administration operations, however all other machines (both internal and external) should not be able to do this. To demonstrate that this limitation has been provided by the instantiated firewall rules in section 1.2, several terminal grabs have been taken.

```
# SSH (port 22) on internal machines set to listen
root@PerimF:~# service ssh start
root@Int-WWW:~# service ssh start
root@SrvGW:~# service ssh start

# can connect from admin machine
root@admin1:~# nc -v 192.168.1.1 22
Connection to 192.168.1.1 22 port [tcp/ssh] succeeded!
SSH-2.0-OpenSSH_8.4p1 Debian-3
^C
root@admin1:~#

root@admin1:~# nc -v 10.10.2.2 22
Connection to 10.10.2.2 22 port [tcp/ssh] succeeded!
SSH-2.0-OpenSSH_8.4p1 Debian-3
^C
root@admin1:~#

root@admin1:~# nc -v 10.10.3.1 22
Connection to 10.10.3.1 22 port [tcp/ssh] succeeded!
SSH-2.0-OpenSSH_8.4p1 Debian-3
^C
root@admin1:~#

# cannot connect from external machine
root@Ext-Office:~# nc -v 192.168.1.1 22
^C
root@Ext-Office:~# nc -v 10.10.2.2 22
^C
root@Ext-Office:~# nc -v 10.10.3.1 22
^C
root@Ext-Office:~#

# cannot connect from unauthorised internal machine
root@WireGuardVPN:~# nc -v 192.168.1.1 22
^C
root@WireGuardVPN:~# nc -v 10.10.2.2 22
^C
root@WireGuardVPN:~# nc -v 10.10.3.1 22
^C
root@WireGuardVPN:~#
```

Firstly, the SSH service (running on default port 22) is started on `PerimF` (`192.186.1.1`), `Int-WWW` (`10.10.2.2`), and `SrvGW` (`10.10.3.1`) to show a variety of machines which can be remotely administered. The next set of terminal grabs show the `admin1` machine being able to successfully connect to each of these machines. Then both an external machine `Ext-Office` and an unauthorised internal machine `WireGuardVPN` are shown to be unable to connect to the SSH service running on each of the machines. Thus, these terminal grabs shows that only authorised internal machines (i.e. machines coming from the Admin zone) are able to send/receive SSH traffic to MEME assets which require

remote administration.

2.2 IP Addressing

With the required connectivity, and dis-connectivity, for MEME's new network architecture to be secure whilst providing the services the organisation needs to fulfil business functions, this report now looks to address MEME's IP addressing. It was requested by MEME that the organisation retains IP addresses for it's Internet facing assets from it's `137.205.0.0/16` public address block. For instance, the `Int-WWW`, `Mail`, `WireGuardVPN`, and `OpenVPN` should be reachable via an IP address from the `137.205.0.0/16` address block by external hosts. Adding to this, MEME's employees should appear that they are accessing external assets from the `137.205.0.0/16` address block, from the external asset's perspective. As such, Source Network Address Translation (SNAT) and Destination Network Address Translation (DNAT) firewall configurations need to be implemented on the `PerimF` machine.

2.2.1 SNAT

Firstly, MEME wants network traffic leaving their internal network to have an address within their `137.205.0.0/16` public IP address block. However, currently, due to the new secure network architecture implemented by this report the source IP addresses of traffic leaving the network has an internal MEME address. This is an issue because it reveals internal IP addresses that MEME uses and, ultimately, can cause routing issues in the real world. This default behaviour is shown in the figure 2.1 below, which is a packet capture from external web server `Ext-WWW` when a `wget http://this.test.com/` request is sent from `Staff-1`.

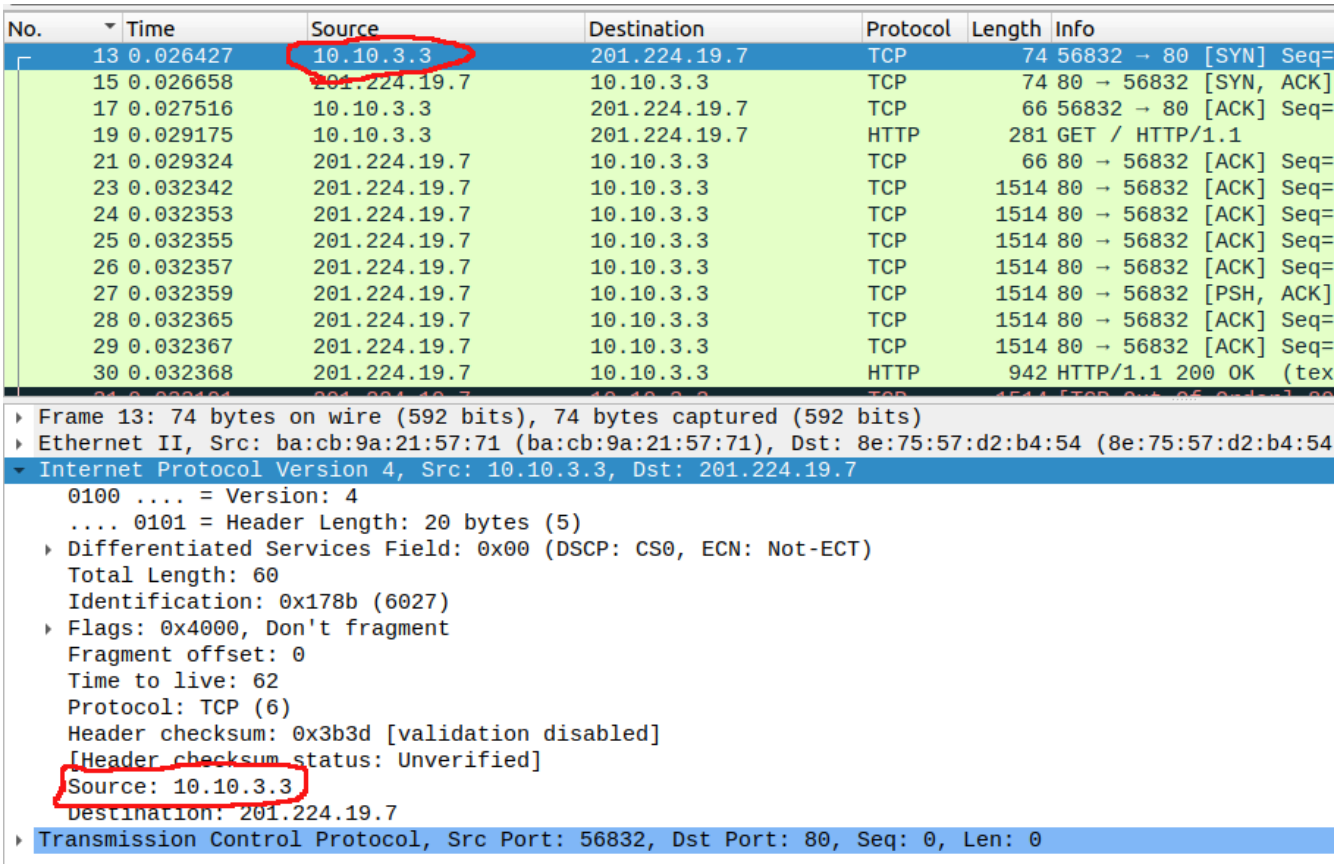

Figure 2.1: Screenshot of Packet Capture Showing Successful DNAT and SNAT for `Int-WWW`

In this screenshot the packet capture shows that the Squid proxy machine (`Squid`) made a HTTP request for the default web page from it's internal IP address `10.10.3.3` – highlighted in red. As such, internal MEME network configuration information is leaked and this needs to be changed so that employees can access the Internet securely. The packet capture file which this screenshot was taken from can be found in `evidence/SNAT_DNAT/before_snat.pcap`. Note, the Squid proxy address is shown because employees use this as a gateway to the Internet for increased security which is discussed in section 2.3.

In order to change this default behaviour, and meet the IP addressing requirements of MEME, Source Network Address Translation (SNAT) needs to be implemented at the Perimeter Firewall `PerimF`. The specific SNAT rules that need to be instantiated are included in table 2.1.

Traffic	Action	Source	Destination
Web traffic (ports 80, 443)	Masquerade as default gateway 137.205.1.1	`Squid` proxy server	External IP addresses
DNS traffic (TCP/UDP port 53)	Masquerade as default gateway 137.205.1.1	`Mail` web server, `Squid` proxy server & Remote Access zone	External IP addresses

Table 2.1: SNAT Firewall Rules on `PerimF` Machine

Evidence that these SNAT rules have been instantiated on the `PerimF` machine can be found in appendix A.1.

Now that SNAT rules have been enabled on the Perimeter Firewall the organisation's internal IP addressing configurations are kept out of the public domain. To demonstrate this the same packet capture as before was taken on the `Ext-WWW` machine as an employee (i.e. `Staff-1`) tried to download it's default root web page through the `Squid` proxy server, once again. This is shown in figure 2.2.

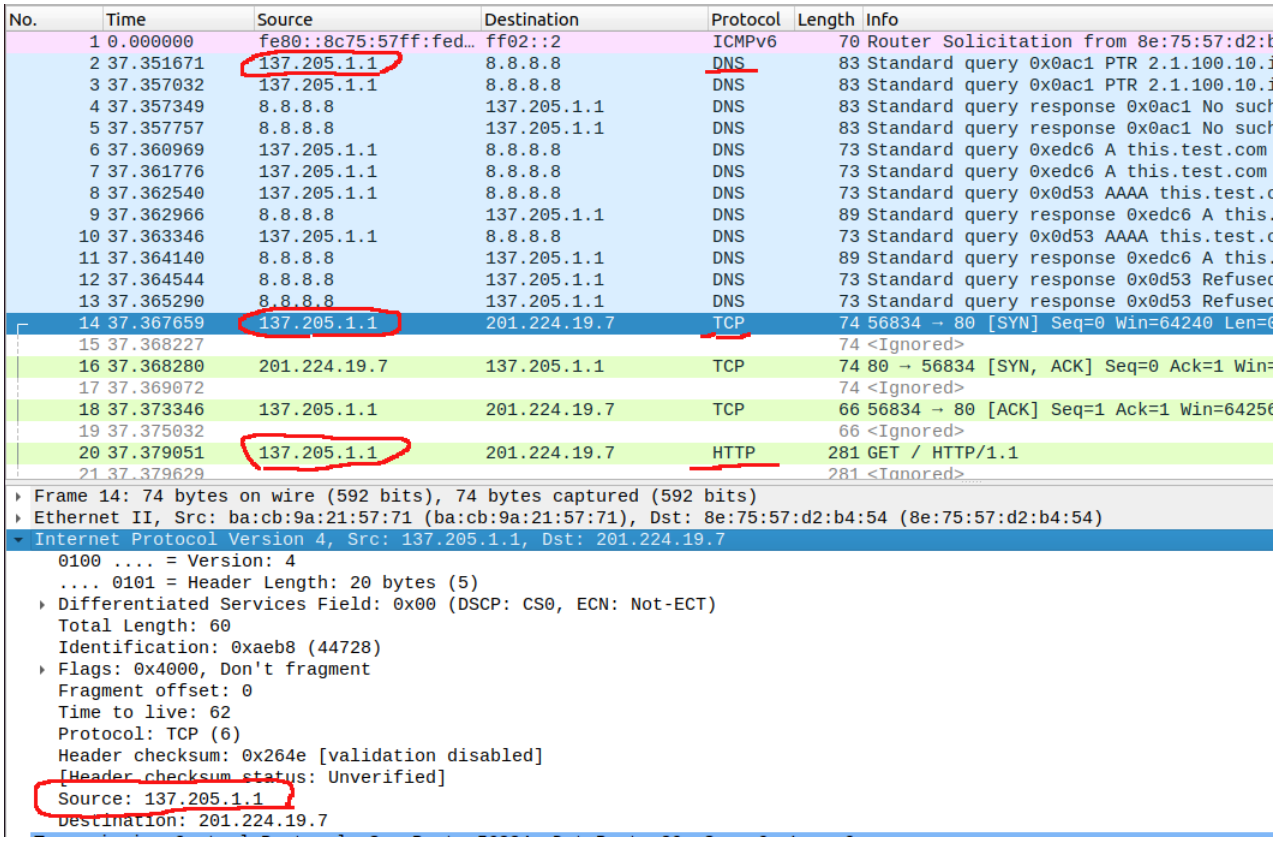

Figure 2.2: Screenshot of Packet Capture Showing Successful DNAT and SNAT for `Int-WWW`

#2.4 This screenshot of the evidence file `evidence/SNAT_DNAT/after_snat.pcap` shows that for both DNS requests to `8.8.8.8` (for `this.test.com`) and TCP/HTTP requests to `201.224.19.7` (`Ext-WWWW`) come from a source IP address of `137.205.1.1` – the default gateway for MEME. As such, anyone intercepting or capturing packets would not know from where within the internal MEME network these requests came from. For brevity packet captures of DNS SNAT for machines within the Remote Access zone and the `Mail` server have been left out, however the same rules have been applied and evidence for this can be found in `PerimF.startup` file's "iptables SNAT/DNAT firewall rules" section.

2.2.2 DNAT

MEME also hosts services which need to be reachable through the public Internet – `Int-WWW`, `Mail`, `OpenVPN`, `WireGuardVPN` – and, as such, these services should be reachable through one of MEME's public IP addresses from in their `137.205.0.0/16` public address block. At the moment external machines use MEME's internal IP addresses to reach

these machines. For instance, in the terminal grab below `Ext-Office` reaches MEME's public web server at `10.10.2.2`.

```
# public web server accessed through internal IP address
root@Ext-Office:~# wget http://10.10.2.2/
--2021-02-09 10:11:41--  http://10.10.2.2/
Connecting to 10.10.2.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html       100%[===============>]  10.45K  --.-KB/s    in 0.001s

2021-02-09 10:11:41 (8.23 MB/s) - 'index.html' saved [10701/10701]

root@Ext-Office:~#
```

This needs to be changed for a few reasons:

- ◇ It is a security concern as it reveals MEME's internal IP addresses.
- ◇ It invalidates MEME's public DNS records for the assets it hosts on the Internet.
- ◇ It invalidates RFCs 1918 and 6598 which define IP address ranges should be private and not reachable from the public Internet (Internet Engineering Task Force, 1996; Internet Engineering Task Force, 2012).

Therefore, Destination Network Address Translation (DNAT) needs to be instantiated on the Perimeter Firewall machine `PerimF` so that these internal assets can be reached via a public IP address. These public IP addresses are listed in table 2.2.

MEME Asset	Public IP Addresse	Internal IP Address	Hostname
`Int-WWW`	137.205.98.155	10.10.2.2	www.meme.cyber.test
`Mail`	137.205.139.114	10.10.2.3	mail.meme.cyber.test
`OpenVPN`	137.205.89.215	10.10.4.2	vpn1.meme.cyber.test
`WireGuardVPN`	137.205.89.220	10.10.4.3	vpn2.meme.cyber.test

Table 2.2: MEME Public Assets with IP Addresses

To translate these public IP addresses into into internal IP addresses where MEME assets can be reached, the DNAT firewall rules in table 2.3 need to be instantiated.

Traffic	Action	Source	Destination
Web traffic (ports 80, 443)	Translate IP to 10.10.2.2	External IP addresses	`Int-WWW` web server
SMTP traffic (ports 25, 587)	Translate IP to 10.10.2.3	External IP addresses	`Mail` server
OpenVPN encrypted traffic (ports TCP/UDP 1194)	Translate IP to 10.10.4.2	External IP addresses	`OpenVPN` server
Wireguard encrypted traffic (UDP port 51820)	Translate IP to 10.10.4.3	External IP addresses	`WireGuardVPN` server

Table 2.3: DNAT Firewall Rules on `PerimF` Machine

Evidence that these DNAT rules have been instantiated on the `PerimF` machine can be found in appendix A.1.

#2.5 In order to demonstrate that these DNAT firewall rules several terminal grabs have been taken. These terminal grabs show an external machine (i.e. `Ext-Office`) connecting to the public facing MEME assets (described above) through their public IP addresses.

```
# web server accessed through public IP address
root@Ext-Office:~# nc -v 137.205.98.155 80
Connection to 137.205.98.155 80 port [tcp/http] succeeded!
^C
root@Ext-Office:~#

# mail server accessed through public IP address
root@Ext-Office:~# nc -v 137.205.139.114 25
Connection to 137.205.139.114 25 port [tcp/smtp] succeeded!
^C
root@Ext-Office:~#

# OpenVPN server accessed through public IP address
root@Ext-Office:~# nc -v 137.205.89.215 1194
Connection to 137.205.89.215 1194 port [tcp/openvpn] succeeded!
^C
root@Ext-Office:~#

# WireGuardVPN server accessed through public IP address
root@Ext-Office:~# nc -vu 137.205.89.220 51820
Connection to 137.205.89.220 51820 port [udp/*] succeeded!
^C
root@Ext-Office:~#
```

As shown `Ext-Office` can make a successful connection to each public facing service that MEME is running on their network via the service's public IP address. This is because of the DNAT firewall rules on the `PerimF` machine. Also, these DNAT rules need a corresponding SNAT rule for the outgoing traffic (replying to request) to transform the internal IP address used within MEME to the external IP address which the public asset uses. These corresponding SNAT firewall rules can be found alongside the aforementioned DNAT rules in the `PerimF.startup` file's "iptables SNAT/DNAT firewall rules" section. To demonstrate that these DNAT and SNAT firewall rules work together as intended, a packet capture of the above terminal grabs has been performed and is partly shown below figure 2.3.

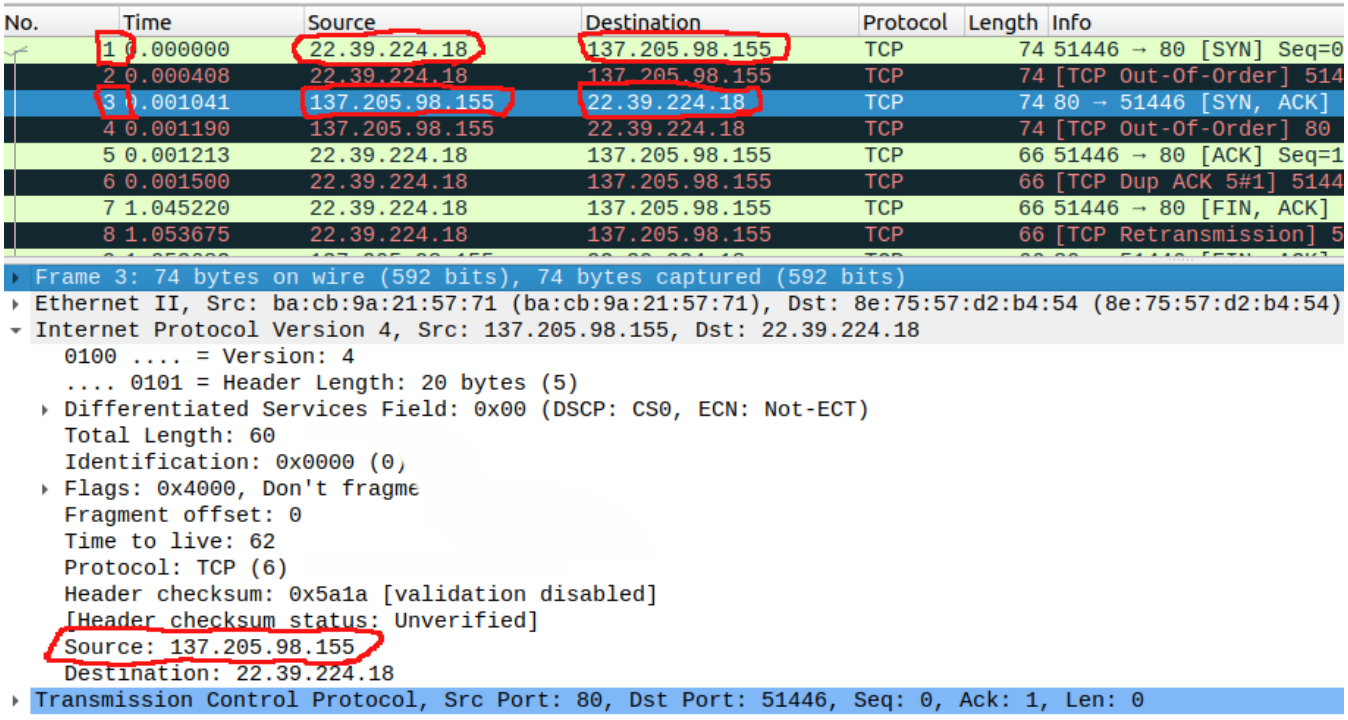

Figure 2.3: Screenshot of Packet Capture Showing Successful DNAT and SNAT for `Int-WWW`

This screen shot the packets captured when `Ext-Office` executed `nc -v 137.205.98.155 80` and sent packets to `Int-WWW`. Packet 1 shows a TCP SYN packet that `Ext-Office` addressed to the public IP address `137.205.98.155`, this was then replied to with a SYN ACK in packet number 3. Packet number 3 is highlighted because it shows that the packet returned in response to the request made by `Ext-Office` has a source IP address of `137.205.98.155` – highlighted in red. As such, this packet capture shows that DNAT and SNAT have been successfully implemented on the `PerimF` machine to translate packets destined to the public IP address `137.205.98.155` to, and then back from, the internal IP address `10.10.2.2`. This translation is needed to achieve a successful route to/from `Int-WWW`. The DNAT and SNAT packet captures for the other terminal grabs are located in packet capture file `evidence/SNAT_DNAT/public_facing_dnat_snat.pcap` which is in the root directory of the Netkit model. For brevity only a screenshot of the first network connection was included.

2.3 Augmentation 1 – Squid Proxy

Now that MEME's network has been re-structured and IP addressing has been re-organised to use MEME's public IP address block `137.205.0.0/16`, this report looks to justify the addition of augmentations which will enhance the organisation's security posture. The first of these is the configuration of the Squid proxy server on machine `Squid`. The Squid proxy service is a "caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator." (Squid, 2013). This web proxy aligns with MEME's foundation charter – whereby the organisation uses open source solutions where possible – and provides the system administration and security operators with an added level of security as they can monitor, filter, and protect employee web traffic. The `Squid` machine was

provided as part of the starter pack, however this machine was not implemented and, as such, this report has implemented a robust configuration of the Squid proxy service to provided MEME with another layer of security. It has been previously shown in section 1.2.7 that the `Squid` machine provided a proxy for employees and executives to access external web services through. In this section the report discusses the security configurations made to make this connectivity secure.

2.3.1 Configuration

The Squid configuration file for this Netkit model can be found on the `Squid` machine at `/etc/squid/squid.conf` and it contains all the configuration settings for the Squid proxy service. This service is automatically started on-boot from within the `Squid.startup` file. The basic configurations in this file can be broken down into five main sections which are highlighted below.

Ports Squid listens on This configuration of Squid uses ports `3128` and `3129` to listen for client connections on. These are the default Squid proxy ports and have been accounted for within the firewalls that allow connections between the Employee & Executive VLANs to this machine.

ACL subnets The Employee VLAN subnet (`10.100.0.0/24` and the Executive VLAN subnet (`10.10.11.0/24`) are defined as the only ACL (Access Control List) subnets which are allowed HTTP access. As such, only machines which connect to the proxy server from these subnets will be allowed to use this proxy for HTTP traffic.

ACL ports This configuration defines several "safe ports" which it allows HTTP traffic to be passed on. For MEME, employees only need to use ports 80 and 443 for web traffic, other types of traffic (i.e. FTP, gopher, wais, etc.) are not needed and so are commented out in this file. They have been left in for if the organisation wants to allow other traffic in the future. This configuration blocks all "unsafe ports" (ports that are not explicitly added to the `Safe_ports` ACL), by default.

DNS server In this configuration file the DNS server which this proxy should use for IP host name look-ups is defined as `8.8.8.8`. This is an external machine and, as such, the firewall configurations made in `SrvGW` and `PerimF` which were previously defined in section 1.2 come into fruition here. An external DNS server is needed for host look-ups because employees will be sending DNS requests for external web services, hence there host names will be on external DNS servers.

Log files This configuration file also defines where access logs and cache logs should be stored – `/var/log/squid/access.log` and `/var/log/squid/cache.log`, respectively. The cache log is used to log when files are cached and the access log is used by administrators to view which machines are requesting what files. These log can be viewed by administrators from within the Admin zone (via SSH) to monitor the websites that employees are visiting and determine if they are appropriate or not. It can also be used in network forensic investigations to identify malicious activity.

These are the basic configuration parameters that this Netkit implementation of the Squid proxy uses. The next section goes beyond these basic configurations to show how they can be used to control the access of clients to the Internet.

2.3.2 Client Authentication & Access Control

With the basic configurations defined, any machine residing within the `10.10.11.0/24` and `10.100.0.0/16` subnets can access any website, at any time of day, without authentication. This means that anyone within MEME's internal, onsite network can access the Internet free of restrictions. However, the Squid proxy service allows administrators to add restrictions to this access, which is the main reason this report has implemented a web proxy in this Netkit model. Firstly, clients connecting to the proxy web server can be made to go through an authentication process in order to access web content. In this model, Squid's basic authentication is used. This authentication mechanism uses an NCSA httpd-style password file to validate clients using the proxy service (Squid, 2020), however other client authentication mechanisms are available and are discussed in section 2.3.3. With Squid's basic client authentication mechanism a client machine connecting to Squid has to provide a username and password, which Squid validates using it's `/etc/squid/passwd` file. Once validated the client can then access web content. This client authentication is done against an ACL (i.e. `AuthUsers`), yet following on from this authenticated users are split into Employees and Executives using the `/etc/squid/employees.txt` and `/etc/squid/executives.txt` files. These authenticated group can then be combined with other ACLs, like source IP subnet, the time of day, or blocked websites, to restrict certain clients. In this Netkit model both employees and executives are required to authenticate themselves, yet these two groups have different access restrictions. To authenticated all users and then split them into employee and executive groups, the following code in `/etc/squid/squid.conf` is required:

```
# enabling squid basic authentication ncsa
auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/passwds
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive on
...

# require authentication and then split in groups
acl AuthUsers proxy_auth REQUIRED
acl EmplGroup proxy_auth "/etc/squid/employees.txt"
acl ExecGroup proxy_auth "/etc/squid/executives.txt"
```

This code checks the `/etc/squid/passwd` file to see if the user and corresponding password are valid, if not then they are denied access. If a valid user then the `/etc/squid/executives.txt` file is checked to see if the username is listed here (i.e. are they an executive) and if it is they are tied to the `ExecGroup` ACL. There are only two executives ("Exec_JohnSmith" & "Exec_JaneDoe") in this model. They both must connect through an IP address in the Executive VLAN range and then they have unrestricted access to any site at any time of the day, which is achieved with the follow configuration in the `/etc/squid/squid.conf` file.

```
# define Executive VLAN subnet for ACL
acl ExecutiveNet src 10.10.11.0/24
...
# deny unauthenticated users
```

```
http_access deny !AuthUsers
...
# allow authenticated Executives from Executive VLAN
http_access allow ExecGroup ExecutiveNet
...
# deny access by default - if user not on ACL
http_access deny all
```

The `/etc/squid/passwd` file in this Netkit model (located on the `Squid` machine) contains the Executives; "Exec_JohnSmith" & "Exec_JaneDoe", and the Employees; "DaveJones", "LewisDunn", & "AnneHall". To demonstrate that this client authentication, and access control from the Executive VLAN works as intended, terminal grabs from a successful and a unsuccessful Executive squid authentication have been taken.

#2.6: Successful Executive squid authentication This terminal grab shows the Executive "Exec_JohnSmith" setting their HTTPS proxy to the `Squid` proxy server with their web proxy access credentials prepended. After this is set, a `wget` request is made to `https//this.test.com`. Note, HTTPS is used here (as port 443 was previously defined as acceptable in the basic configurations above) which requires the `wget` option `--no-check-certificate` to be used as x509 certificates have not been implemented on this demo web server `Ext-WWW`. Ultimately, this web request is successful and the default web page is downloaded to the `Staff-2` machine.

```
# successful executive HTTPS web request
root@Staff-2:~# export https_proxy="https://Exec_JohnSmith:pass@10.10.3.3:3128/"
root@Staff-2:~# wget --no-check-certificate https://this.test.com
--2021-02-10 14:27:05--  https://this.test.com/
Connecting to 10.10.3.3:3128... connected.
WARNING: The certificate of 'this.test.com' is not trusted.
WARNING: The certificate of 'this.test.com' doesn't have a known issuer.
The certificate's owner does not match hostname 'this.test.com'
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html       100%[===============>]  10.45K  --.-KB/s    in 0.003s

2021-02-10 14:27:05 (3.71 MB/s) - 'index.html' saved [10701/10701]

root@Staff-2:~#
```

#2.7: Unsuccessful Executive squid authentication This first terminal grab shows an Employee trying authenticate from machine within the Executive VLAN. However, they are not on the list of users who can authenticate from this subnet and so are revoked.

```
# failed executive web request - bad authentication
root@Staff-2:~# export https_proxy="http://DaveJones:pass@10.10.3.3:3128/"
root@Staff-2:~# wget --no-check-certificate https://this.test.com
--2021-02-15 10:28:06--  https://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy tunneling failed: ForbiddenUnable to establish SSL connection.
root@Staff-2:~#
```

This second terminal grab shows an Executive trying to authenticate from the a machine on the Employee VLAN (i.e. `Staff-1`). However, this VLAN is, currently, not allowed HTTP access (only the ExecutiveNet) and so is forbidden.

```
# failed executive web request - bad subnet source
root@Staff-1:~# export https_proxy="http://Exec_JohnSmith:pass@10.10.3.3:3128/"
root@Staff-1:~# wget --no-check-certificate https://this.test.com
--2021-02-15 10:29:59--  https://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy tunneling failed: ForbiddenUnable to establish SSL connection.
root@Staff-1:~#
```

This final terminal grab shows an Executive using an invalid password to authenticate themselves, despite being on the correct VLAN. Therefore, they are also revoked.

```
# failed executive web request - bad password source
root@Staff-2:~# export https_proxy="http://Exec_JohnSmith:wrong_pass@10.10.3.3:3128/"
root@Staff-2:~# wget --no-check-certificate https://this.test.com
--2021-02-15 10:39:52--  https://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy tunneling failed: Proxy Authentication RequiredUnable to
    establish SSL connection.
root@Staff-2:~#
```

Although both executive and employees can authenticate themselves to the Squid proxy and then access any web content,employees have access control restraints implemented against them in this Netkit model. These restraints are implemented by appending the following ACLs to their `http_access` and include:

- Employees are limited to access web content to a certain time of day (i.e. 8:30 to 18:00). This is achieved through the `daytime` ACL:

  ```
  acl daytime time 08:00-18:00
  ```

- Employees are not permitted to access certain websites (i.e. forbidden from blacklisted websites). This is achieved through the ACL `bad_urls` which uses an ACL file `/etc/squid/sites.blacklist.txt` to deny HTTP access to certain websites:

  ```
  acl blacklist dstdomain "/etc/squid/sites.blacklist.txt"
  ```

- Employees are not permitted to search for certain web content (i.e. certain keywords are blacklisted). This is achieved through the ACL `block_words` which uses the file `/etc/squid/words.blacklist.txt` to deny HTTP access when certain words are included in the URL:

  ```
  acl blackwords url_regex "/etc/squid/words.blacklist.txt"
  ```

These three ACLs are then used to limit employee access with the following configuration:

```
http_access allow EmplGroup EmployeeNet !blacksites !blackwords daytime
```

This configuration means that employees must authenticate (and be on the `/etc/squid/employees.txt` list), have a source IP address from the `EmployeeNet` (10.100.0.0/16), their request cannot contain any blacklisted websites or keywords, and the request must be between the working hours of 08:00 and 18:00. If any of these ACLs are not met, then access is denied by default the `http_access deny all` fallback rule. Evidence that these access controls have been configured can be found in the `/etc/squid/squid.conf` as before. In order to demonstrate that these access controls have been successfully implemented in the Netkit model, several terminal grabs have been captured.

#2.8: Successful connection from Employee machine The terminal grab below shows a successful connection from `Staff-1` using the Squid proxy as employee "AnneHall". This request was made to a website that is not on the blacklisted sites list (in the `/etc/squid/sites.blacklist.txt` file) and the URL was void of blocked keywords (keywords in the `/etc/squid/words.blacklist.txt` file). In addition to this, the time when the request was made was is "daytime" (between 08:00 and 18:00 hours).

```
# successful employee request - follows access control rules
root@Staff-1:~# export http_proxy="http://AnneHall:pass@10.10.3.3:3128/"
root@Staff-1:~# wget http://this.test.com
--2021-02-15 10:47:14--  http://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html'

index.html       100%[===============>]  10.45K  --.-KB/s    in 0.01s

2021-02-15 10:47:15 (1003 KB/s) - 'index.html' saved [10701/10701]

root@Staff-1:~#
```

#2.9: Unsuccessful connections from Employee machine The terminal grabs below show several unsuccessful connections made from the `Staff-1` using the Squid proxy as employee "AnneHall". The first request was made to a blacklisted site (i.e. `http://bad-website.com/`) which is specified in the `/etc/squid/sites.blacklist.txt` file and, as such, the request was forbidden. The request second contained a URL that had a blocked keyword (i.e. `poop`) as specified in the `/etc/squid/words.blacklist.txt` file, thus was also forbidden. Finally, the third request was made out of normal working hours (08:00 and 18:00 hours), a request at "23:00:34" is indication of suspicious activity, hence was also forbidden.

```
# request 1:  failed - blacklisted website
root@Staff-1:~# wget http://www.bad-website.com
--2021-02-15 10:48:53--  http://www.bad-website.com/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 403 Forbidden
2021-02-15 10:48:53 ERROR 403: Forbidden.

root@Staff-1:~#

# request 2:  failed - blocked keyword
root@Staff-1:~# wget http://www.google.com?q=poop/
--2021-02-15 10:50:06--  http://www.google.com/?q=poop/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 403 Forbidden
2021-02-15 10:50:06 ERROR 403: Forbidden.

root@Staff-1:~#

# request 3:  failed - out of company working hours (23:00)
# change time on Squid and Staff-1 machines (for test)
root@Squid:~# timedatectl set-time '23:00'
root@Staff-1:~# timedatectl set-time '23:00'

# making web request at 23:00:34 (11pm)
```

```
root@Staff-1:~# wget http://this.test.com
--2021-02-15 23:00:34--  http://this.test.com/
Connecting to 10.10.3.3:3128... connected.
Proxy request sent, awaiting response... 403 Forbidden
2021-02-15 23:00:34 ERROR 403: Forbidden.

root@Staff-1:~#
```

These terminal grabs show that the access control rules instantiated in the `squid.conf` are successful and provide the system administration and security team with finer grain control over employee web activity.

2.3.3 Future Work

It is important to highlight that the basic Squid authentication mechanism is not very secure. It exchanges passwords in clear-text and it stores passwords in clear-text in its basic authentication memory cache (Squid, 2016). Therefore, if the `Squid` proxy server is compromised, or another machine is compromised within MEME's network and this traffic is sniffed, then all employee proxy passwords can be easily obtained. To combat this, this report recommends implementing Squid's LDAP authentication mechanism which can be used to authenticate employees through an LDAP server (Kahn, 2009). This LDAP server can securely exchange authentication traffic with the Squid server through a secure TLS channel which can be created through the use of TLS certificates. However, this would require the implementation of a fully-functioning LDAP server and so has been postponed for future work when MEME transitions the Netkit model into a commercial network environment.

Another possible future addition to this Squid proxy configuration would be to setup an `ssl_bump` setting to intercept HTTPS request, decrypt the request to search for illicit activity, and then re-encrypt it before sending it out to the web. However, for a medium sized organisation like MEME this report considered the breaking of TLS encryption too problematic to try to implement in this Netkit model. This is because of the security concerns that such a practice would raise and the increased security practices that would need to be implemented to address these concerns. Intercepting and breaking the encryption of web requests required all the organisation's TLS/SSL certificates to be kept on the `Squid` machine which makes this machine a prime target. If an attacker is able to compromise this machine then they will be able to break the encrypted communication of all web requests made by MEME and perform a man-in-the-middle attack against the entire organisation. This level of security risk was considered too great at this current time, however in the future when more resources are able to be allocated to securing MEME's machines (and a commercial web proxy is available) then this could be an option for the organisation.

2.4 Phase 2 Summary

Reference & Claim – sections 2.1, 2.2, 2.3	Evidence
Ref: #2.1 Internal MEME assets cannot be accessed by external machines unless a specific rule exists allowing them to do so	Firewall rules on all internal MEME assets (in particular `PerimF`) and terminal grabs in section 2.1
Ref: #2.2 Internal MEME assets cannot be accessed by other internal MEME machines unless a authorised to do so	Firewall rules on all internal MEME assets and terminal grabs in section 2.1
Ref: #2.3 MEME assets cannot access external machines unless authorised to do so	Firewall rules on all internal MEME assets and terminal grabs in section 2.1
Ref: #2.3 Unauthorised machines (those outside of MEME's Admin zone subnet) cannot remotely administer other machines (i.e have SSH access)	Firewall rules on all internal MEME assets (and for access rules on particular `PerimF`, `AdminGW`, & `admin1`) and terminal grabs in section 2.1
Ref: #2.4 SNAT has been implemented for MEME assets which require access to the external network	SNAT firewall rules on `PerimF` machine, appendix A.1, SNAT screenshot in figure 2.2, and the packet capture file `evidence/SNAT_DNAT/after_snat.pcap`
Ref: #2.5 DNAT (and corresponding SNAT) has been implemented for MEME's public facing assets so they are reachable via MEME's public IP range	DNAT/SNAT firewall rules on `PerimF` machine, the DNAT terminal grabs in section 2.2, appendix A.1, the DNAT screenshot in figure 2.3, and the packet capture file `evidence/SNAT_DNAT/public_facing_dnat_snat.pcap`
Ref: #2.6 Squid proxy allows MEME executives to access the Internet provided they use authentication and the Executive VLAN	Configuration settings in `/etc/squid/squid.conf` on `Squid` and terminal grabs in section 2.3
Ref: #2.7 Squid proxy blocks anyone who is not a MEME executive if they use the Executive VLAN with bad authentication or the Employee VLAN as an executive from accessing the Internet	Configuration settings in `/etc/squid/squid.conf` on `Squid` and terminal grabs in section 2.3
Ref: #2.8 Squid proxy allows MEME employees to access the Internet provided they use authentication, don't try to access blocked content, and the Employee VLAN	Configuration settings in `/etc/squid/squid.conf` on `Squid` and terminal grabs in section 2.3
Ref: #2.9 Squid proxy blocks MEME employees from accessing the Internet if they use bad authentication, try to access blocked content, or don't use the Employee VLAN	Configuration settings in `/etc/squid/squid.conf` on `Squid` and terminal grabs in section 2.3

Table 2.4: Phase 2 Work Summary Table

3 Phase 3

3.1 Augmentation 2 – Management Network

Another augmentation that this report recommends MEME implement, and has implemented in the Netkit model, is a Management Network. This has been discussed throughout with the inclusion of the Admin zone from which system administrators and security operators could remotely manage MEME's network and infrastructure assets. This remote administration is done through SSH and it was previously demonstrated that every machine within MEME's network is reachable through SSH only from machines within this Admin zone subnet. With this architecture in place, the administration of MEME's assets is significantly easier and more efficient and this will lead to an increase in the organisation's security posture for a number of reasons:

- ◇ Alerts from all of MEME's network can reach the administration/security team.
- ◇ Incidents can be responded to quickly with rapid configuration changes.
- ◇ Machines which are not easy to physically access can be easily accessed remotely.
- ◇ Multiple administration/security team members can configure a single remote machine as they don't need to be physically logged in and so the work can be streamlined.
- ◇ Administration/security team members can provide troubleshooting remotely to employees
- ◇ Additional security architecture like IDS, IPS, and honey pot systems can be setup to communicate directly with the Admin zone.

These benefits are why this report deemed it necessary to implement a Management Network into MEME's network when redesigning the organisation's network architecture. However, thus far only SSH connectivity of machines within the Admin zone subnet to other MEME assets has been show. As such, this report will now discuss how it has securely implemented SSH on these other machines so that secure, efficient remote administration is available.

3.1.1 SSH Server Configuration – # 3.1

For a the Management Network to function each of the main infrastructure and service assets runs an SSH server. With the previous firewall settings defined in section 1.2, these SSH servers only allow SSH connections from IP addresses within the Admin zone subnet (`10.10.5.0/24`). However, the default OpenBSD SSH server (the SSH server that this Netkit model has implemented) configuration can be improved with security hardening settings so that MEME's assets are more secure. The server-specific configurations that this report recommends, and has implemented, are:

Strong cryptographic policy It is important that strong symmetric algorithms (`Ciphers`), host key algorithms (`HostKeyAlgorithms`), key exchange algorithms (`KexAlgorithms`), and message authentication code algorithms (`MACs`) are used for the exchange of data when using the SSH protocol. This report recommends, and has implemented, the following cryptography algorithms/protocols in the `/etc/ssh/sshd_config` file:

```
ciphers aes128-ctr,aes192-ctr,aes256-ctr
HostKeyAlgorithms ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,
    ecdsa-sha2-nistp521,ssh-rsa,ssh-dss
KexAlgorithms ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,
    diffie-hellman-group-exchange-sha256,diffie-hellman-group16-sha512,
    diffie-hellman-group18-sha512,diffie-hellman-group14-sha256
MACs hmac-sha2-256,hmac-sha2-512
```

These cryptographic algorithms/protocols have been chosen based on the guidance of the SSH.COM – a company who specialise in modern SSH solutions for enterprises – and NIST (SSH.COM, 2021; National Institute of Standards and Technology, 2015). Note, SHA-1 has been excluded from this list because it's security capabilities are questionable and it is only used in industry for compatibility with legacy assets. MEME has not specified that they use any legacy equipment which would require SHA-1 to be used, thus it has been excluded from this list of acceptable cryptoghraphic algorithms. Despite SHA-1 being excluded, many of the default ciphers are kept on the SSH server so that if there are any future SSH clients that require these ciphers, then they can be used. The ciphers used are more specific on the SSH client machine `admin1`.

Disabling X11 forwarding X11 allows for a GUI to be displayed over a remote SSH connection, however it is not needed for MEME's system administration team and only increases the organisation's attack surface as it opens up a channel back to the client through which the server could send malicious commands (Boelen, 2019; Scaife, 2020). This is disabled in the SSH server's configuration file `/etc/ssh/sshd_config` with the line:

```
X11Forwarding no
```

Disabling rhosts *rhosts* is a weak authentication method that allows a system to be trusted just based on it's IP address (Boelen, 2019). It is usually disabled by default, but to make this explicit the following line has been included in `/etc/ssh/sshd_config`:

```
IgnoreRhosts yes
```

Disabling root login It is best practice not to log in as the root user and instead use a normal user account to manage remote services. As such, each asset which has remote management functionality has an `admin1` account which the `admin1` machine logs into to perform administrative operations from. This account created in each assets `*.startup` file and is added to the `sudoers` group so it can make system changes. With this account in use direct root logins should be disabled to prevent unwarranted privileges (Boelen, 2019; Scaife, 2020). This is done with the following line in the `/etc/ssh/sshd_config` file:

```
PermitRootLogin no
```

Managing passwords To make MEME's network-side SSH implementation as secure as possible this report has implemented public key authentication into the Management Network. This means that SSH keys are used for authentication rather than passwords or other authentication mechanisms. Public key authentication is considered more secure and is less prone to brute-force attacks (Boelen, 2019), thus has been implemented in this Netkit model. The public keys which are authorised for use on each SSH server are stored within the `/etc/ssh/%u/authorized_keys` file in this configuration, as defined by the settings below:

```
PubkeyAuthentication yes
AuthorizedKeysFile /etc/ssh/%u/authorized_keys
```

Each asset that has remote administration functionally has an `authorized_keys` file that contains the public key of the `admin1` used to login to that machine. Adding to this, with public key authentication setup the following configurations can also be set in the `/etc/ssh/sshd_config` file to prevent passwords leaking, to prevent logins if a user's password is set to blank or is empty, and to disable authentication methods that are not used to reduce the attack surface of the SSH server (Scaife, 2020):

```
PasswordAuthentication no
PermitEmptyPasswords no

ChallengeResponseAuthentication no
KerberosAuthentication no
GSSAPIAuthentication no
```

Protecting against brute force attacks To protect against an attacker repeatedly trying to guess an SSH username/password combination one can limit the maximum amount of authentication attempts for a particular login session (Scaife, 2020). Failed SSH login attempts can also be monitored through a SIEM solution and forwarded to the security team or through a Pluggable Authentication Module (PAM) (Boelen, 2019). This login attempt limit is set on the SSH server with the following setting in the `/etc/ssh/sshd_config` file:

```
MaxAuthTries 3
```

Protecting against denial-of-service (DoS) attacks To protect against DoS attacke one can reduce the login grace period. This is the time a user has to complete their initial authentication to connect to the SSH server (Scaife, 2020). This protects against DoS attackers where multiple authentication sessions are kept open for a prolonged period in an effort to eat up system resources on the SSH server. Limiting login grace time is done in the `/etc/ssh/sshd_config` file with the following line:

```
LoginGraceTime 20
```

Disabling custom client environment variables By default, clients can pass custom environment variables (i.e. their `$PATH` or terminal settings) to their SSH shell (Scaife, 2020). However, this is not commonly used and is not required for remote administrators in MEME's network environment, thus has been disabled here with the following configuration setting in the `/etc/ssh/sshd_config` file:

```
PermitUserEnvironment no
```

Displaying a banner It is common practice for organisations to display a login banner which displays legal warning before a user starts their SSH session on a machine (SSH.COM, 2021). This is instantiated for MEME with the following line in the `/etc/ssh/sshd_config` file:

```
Banner /etc/ssh/MEME_banner
```

The specific banner that is displayed can be found in the `/etc/ssh/MEME_banner` file.

Disabling port forwarding Most organisations want to prevent port forwarding on their servers as it poses a substantial risk that SSH tunnelling will be used to bypass firewalls (SSH.COM, 2021). As such, this Netkit model has decided to disable port forwarding on SSH servers, in lieu of any requirements for this functionality requested for by MEME (i.e. to support legacy applications). Disabling port forwarding is done with the following configurations in the `/etc/ssh/sshd_config` file:

```
AllowAgentForwarding no
AllowTcpForwarding no
AllowStreamLocalForwarding no
PermitTunnel no
GatewayPorts no
```

Restricting to Admin zone clients In this Netkit model of MEME's network, only machines which are located within the Admin zone subnet (`10.10.5.0/24`) should be able to make SSH connections to the SSH servers running on networked assets. As such, an IP allow list has been created within each networked asset's `/etc/ssh/sshd_config` file whereby only connecting IP addresses from within the `10.10.5.0/24` block are allowed to make an SSH connection. Ensuring that only clients from within this pre-approved IP address range can log in means that the risk of a breach event occurring due to the leak of a private SSH key is reduced (Scaife, 2020). The following configuration instantiates this functionality:

```
AllowUsers *@10.10.5.0/24
```

SFTP server In order to upload content via SSH to remote assets a Secure File Transfer Protocol (SFTP) server has to be used. This is defined in the `/etc/ssh/sshd_config` file with the following line:

```
Subsystem sftp      /usr/lib/openssh/sftp-server
```

By specifying a path to an SFTP server that is compatible with the SSH server running, a client can now use their SSH login to login to the SFTP server an asset is running and then securely upload/download files. This means a client (i.e. `admin1`) does not need to create files (i.e. web pages, iptables firewall rules, etc.) on-the-fly on the target machine, instead they can create these files in advance, test them, and then upload them.

These SSH configurations have been made network-wide on each MEME networked asset which requires remote administration functionality. See appendix B.1 for a list of all SSH server configurations that have been made, including the ones specified. The SSH server configuration files can be found at `<machine>/etc/ssh/sshd_config` in this Netkit model.

3.1.2 SSH Client Configuration – #3.2

Aside from hardening the SSH server configuration on MEME's infrastructure and service assets, this report also recommends hardening the security configuration of the SSH client machines which interact with these assets. For the Management Network this includes all machines within the Admin zone subnet (i.e. `admin1`). These machines are used to interact with the SSH servers running on MEME's infrastructure and service assets and, as such, need to run securely. The client-specific configurations that this report recommends, and has implemented, are:

Parameters to match SSH server configuration To start with, it is important that the configurations the SSH client uses match the configurations the SSH server uses. The following configurations in `/etc/ssh/ssh_config` on the `admin1` machine do this:

```
...
ForwardAgent no
ForwardX11 no
PasswordAuthentication no
HostbasedAuthentication no
GSSAPIAuthentication no
...
```

These settings make sure the client and server SSH configuration matches. Although, no necessary as the server will ultimately dictate what is allowed, this makes it explicit about what should be happening.

Cryptography algorithms/protocols It is important that the SSH client (i.e. `admin1`) uses strong encryption algorithms to make a connection and send data to the SSH server. Also, because this implementation of SSH uses public key authentication the `IdentityFile` (the file containing the SSH public key) needs to be set. All of these configurations are included in the `/etc/ssh/ssh_config` file on the client machine and are shown below:

```
IdentityFile /root/.ssh/id_rsa
Ciphers ae128-ctr,aes192-ctr,aes256-ctr, aes128-cbc
MACs hmac-sha2-256,hmac-sha2-512
```

These secure configurations have been set to be used on all hosts the `admin1` machine connects to through SSH with the following setting in the machine's `/etc/ssh/ssh_config` file:

```
Host *
    ...
```

This report decided to implement this configuration because each MEME asset `admin1` makes an SSH connection to will be running the same server configuration and will require the same (secure) algorithms/protocols to be used. If there were other MEME assets which required different host configurations so that `admin1` could remotely interact with it (i.e. legacy assets) then this can be configured at a later data with a configuration like:

```
Host LegacyAsset
    Hostname LegacyAsset.meme.cyber.text
    User admin1
    ...
    ...
```

These SSH client configurations above have been made on the `admin1` machine within the `/etc/ssh/ssh_config` file. See appendix B.1 for a list of all SSH client configurations that have been made, including the ones specified.

3.1.3 Evidence

To demonstrate that the Management Network works as an augmentation that will increase the security posture of MEME, this report has used a typical example of a remote

administration scenario. In this scenario, a member of the system administration/security team uses `admin1` to initiate a secure SSH session with the `Int-WWW` web server so that they can remotely upload content via SFTP. Firstly, a member of the administration/security team uses `admin1` to SSH into the MEME's web server `Int-WWW` using public key authentication, which means they only need to know the password for the public key on `admin1` to securely log into the `admin1` account on `Int-WWW`. From here they then can securely upload a web page to the MEME's website, all without needing direct, physical access to the web server. In this example the `admin1` account has been given ownership of the `/etc/www` directory in `Int-WWW.startup` file so that is can upload files remotely – this can be made more secure in the real world by creating a `www` group and adding the `admin1` account as a member.

```
# show web page doesn't already exists (404 error)
root@Ext-Office:~# wget http://www.meme.cyber.test/newpage.html
--2021-02-15 16:35:57--  http://www.meme.cyber.test/newpage.html
Resolving www.meme.cyber.test (www.meme.cyber.test)... 137.205.98.155
Connecting to www.meme.cyber.test
    (www.meme.cyber.test)|137.205.98.155|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
2021-02-15 16:35:57 ERROR 404: Not Found.

root@Ext-Office:~#

# login int to Int-WWW using SFTP
root@admin1:~# sftp admin1@10.10.2.2
...
Enter passphrase for key '/root/.ssh/id_rsa':
Connected to 10.10.2.2.
sftp>

# upload new web page to Int-WWW and logout
sftp> cd /var/www/html/
sftp> put NewPage.html
Uploading NewPage.html to /var/www/html/NewPage.html
NewPage.html                                  100%  116    12.5KB/s   00:00
sftp> exit
exit
root@admin1:~#

# external user accessing MEME's new web content
root@Ext-Office:~# wget http://www.meme.cyber.test/NewPage.html
--2021-02-15 16:49:28--  http://www.meme.cyber.test/NewPage.html
Resolving www.meme.cyber.test (www.meme.cyber.test)... 137.205.98.155
Connecting to www.meme.cyber.test
    (www.meme.cyber.test)|137.205.98.155|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117 [text/html]
Saving to: 'NewPage.html'

NewPage.html     100%[===============>]     117  --.-KB/s    in 0s

2021-02-15 16:49:28 (772 KB/s) - 'NewPage.html' saved [117/117]

root@Ext-Office:~#
```

#3.4 Next, a new DNS entry (i.e. `staff-2.meme.cyber.test`) is added to `Int-DNS` remotely from the `admin1` machine using SSH, and then the DNS service (`dnsmasq`) is restarted to reflect this. Note, `sudo` (super user) capabilities are required here to make system changes so the `admin1` account needs to be configured to do so – this is done in `Int-DNS.startup`.

```
# show DNS entry doesn't already exists
root@Staff-1:~# host -t a staff-2.meme.cyber.test dns.meme.cyber.test
Using domain server:
Name: dns.meme.cyber.test
Address: 10.10.3.2#53
Aliases:

Host staff-2.meme.cyber.test not found: 3(NXDOMAIN)

root@Staff-1:~#

# login to Int-DNS using SSH
root@admin1:~# ssh admin1@10.10.3.2
...
Enter passphrase for key '/root/.ssh/id_rsa':
Linux Int-DNS 5.10.4 #1 Mon Jan 25 03:08:46 PST 2021 x86_64
Welcome to Netkit

admin1@Int-DNS:~$

# add DNS entry, restart DNS service, and logout
admin1@Int-DNS:~$ vim /etc/dnsmasq_static_hosts.conf
...
10.100.1.3 staff-2.meme.cyber.test
...
"dnsmasq_static_hosts.conf" 22L, 456B written
admin1@Int-DNS:~$ sudo service dnsmasq restart
[sudo] password for admin1:
admin1@Int-DNS:~$ exit
logout
Connection to 10.10.3.2 closed.
root@admin1:~#

# new DNS entry can be seen by MEME asset (Staff-1)
root@Staff-1:~# host -t a staff-2.meme.cyber.test dns.meme.cyber.test
Using domain server:
Name: dns.meme.cyber.test
Address: 10.10.3.2#53
Aliases:

staff-2.meme.cyber.test has address 10.100.1.3
root@Staff-1:~#
```

#3.5 Finally, a firewall change is made on `PerimF` remotely from the `admin1` machine using SSH. This change blocks all incoming traffic to port 25 (i.e. SMTP) – for demonstration purposes only.

```
# show can access port 25 on Mail server from outside
root@Ext-Office:~# nc -v mail.meme.cyber.test 25
Connection to mail.meme.cyber.test (137.205.139.114)
    25 port [tcp/smtp] succeeded!
```

```
^C
root@Ext-Office:~#

# login to PerimF using SSH
root@admin1:~# ssh admin1@192.168.1.1
...
Enter passphrase for key '/root/.ssh/id_rsa':
Linux PerimF 5.10.4 #1 Mon Jan 25 03:08:46 PST 2021 x86_64
Welcome to Netkit

admin1@PerimF:~$

# make a change to the PerimF's firewall and logout
admin1@PerimF:~$ sudo iptables -I FORWARD 3 -i eth1 -o eth0 \
    -d 10.10.2.3 -p tcp --dport 25 \
    -m comment --comment "block all SMTP from external hosts" \
    -j DROP
[sudo] password for admin1:
admin1@PerimF:~$ exit
logout
Connection to 192.168.1.1 closed.
root@admin1:~#

# external user blocked from port 25 by new firewall rule
root@Ext-Office:~# nc -v mail.meme.cyber.test 25
^C
root@Ext-Office:~#
```

This remote administration ability is made possible through the Management Network that this Netkit model creates and only a sample has been shown above. For instance, gateway machines like `IntGW` & `RemoteAccessGW` can be managed through SSH, outward facing services like `Mail` & `Squid` & `OpenVPN` can be monitored through SSH, and internal services like `LDAP` & `DHCP` can be configured through SSH. All assets which require remote administration have an SSH server running which is configured to only accept SSH traffic from MEME's administration/security assets located within the Admin zone subnet. This happens through network-based firewall settings, host-based firewall settings, SSH server settings, and SSH client settings. In addition to this, the whole process is password-less (on the network end) with only the user on `admin1` having to know the password to use the public key on that machine. This password-less interaction is possible because each networked asset accepts `admin1`'s public key as being "authorised" to use the `admin1` account on the asset to perform administrative duties.

3.1.4 Future Work

The SSH server configuration changes are not entirely reflective of how this Netkit model should be deployed in the real world. Although, almost all security configurations are final, the `StrictModes no` is not. This configuration is required to get the SSH server (and clients) to function appropriately in the Netkit model. This is a limitation of the Netkit software which struggles to handle file permissions appropriately when new users are created and would be changed when this model is deployed in the real world. Because this Netkit model chose to not use the Root account on MEME assets which are remotely administered, but instead create and admin1 account on each machine within their `*.startup` files, the file permissions for the `authorized_keys` file become estranged

and difficult to change without effecting the whole file system. As such, without the proper file permissions implemented the `authorized_keys` file on the SSH server is not accessible and public key authentication cannot be used. To get around this limitation the configuration `StrictModes yes` is set to ignore loose file permissions on the SSH server side. Meanwhile, SSH file permissions on the client side are unaffected by this limitation and can be properly configured. In the future, when this Netkit model is deployed in the real world, the file permissions can be properly managed and the `StrictModes yes` settings can be instantiated to make the SSH server more secure.

3.2 Augmentation 3 – Intrusion Detection System

The final augmentation that this report recommends implementing to bolster the security posture of MEME is an Intrusion Detection System (IDS) and, more specifically, Suricata. Suricata is a free and open source network threat detection engine that can run in a variety of modes – real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM), and offline pcap processing (Open Information Security Foundation, 2021d). It supports an extensive rule and signature language which can detect complex threats by inspecting network traffic and then output this detection in a range of formats that can be integrated with existing SIEMs, Splunk, Logstash/Elasticsearch, Kibana, and other databases (Open Information Security Foundation, 2021d). Implementing such an IDS solution into MEME's network (creating a NIDS) boasts several benefits:

- ◇ Insight into network paths and activity by looking at data in the context of a protocol.
- ◇ Instant notifications if harmful activity is detected.
- ◇ Can track network attacks/viruses as they develop and move around the network.
- ◇ Can classify attacks in real time.
- ◇ Gives the company a better security reputation among clients.
- ◇ Can tune the IDS to look at specific content in network packets to monitor for specific threats.

In addition to this, Suricata is open source which aligns with the organisation's foundation charter and makes a good fit for MEME.

3.2.1 Design & Configuration

#3.6 When implementing an IDS solution it is important to consider where the IDS will be located within MEME's network so that it is used most efficiently. In this Netkit model the Suricata IDS has been placed on the DMZ gateway machine `DmzGW` machine. This machine is the gateway to/from MEME's public facing assets `Int-WWW` and `Mail`, therefore is the node in MEME's network which is most likely to see malicious traffic heading to these open services. This malicious traffic is detected through Suricata rules (using a similar syntax to Snort rules) which are discussed in 3.2.2. In this model Suricata is running in live IDS mode using the AF_PACKET format to capture packets at high speed on the Linux platform (Open Information Security Foundation, 2021c). From here, Suricata analyses packets and then generates alerts based on predefined rules. For the most part the default Suricata (`suricata-6.0.1`) configuration file (`suricata.yaml`) is

used to configure the IDS in this Netkit model. Nevertheless, there are notable changes made to this file to customise Suricata to run in MEME's network environment:

- ◇ Variables changed (in `vars`) section to reflect IP addresses of MEME assets (i.e. `Int-WWW` and `Mail`) and the DMZ subnet `10.10.2.0/24`.
- ◇ "EVE" logging is enabled (in `outputs` section) to display alerts in the JSON format for easy reading using the `jq` Linux utility (which has been installed) and integration into APIs.
- ◇ Packet capture changes in the `af-packet` and `pcap` sections of the configuration file, including; changing the capture interface, enabling "mmap" (memory mapping) to speed up capture, and enabling "tpacket-v3" – the latest version of the ring buffer API for Linux kernels 3.2 onward which allows variable-length frames, has improvements to polling, and provides access to hash data, but at the cost of higher latency when run as an IPS (Shaw, 2018; Camara and Baudy, 2020).
- ◇ Changing the "server-config" in the `app-layer` section to reflect the apache2 configuration installed on `Int-WWW`.
- ◇ Extending the logging information for the "smtp" protocol (in the `outputs` section) to include; bcc, message-id, subject x_mailer, and user-agent.
- ◇ Changing the host specific policy for defragmentation and TCP reassembly to Linux in the `host-os-policy` section.
- ◇ Changing the `rule-files` section to point to the paths of the rule files on the `DmzGW` machine.

#3.7 To use Suricata on the `DmzGW` machine it has to be re-installed every time the Netkit model is started because all the Netkit machines are ephemeral. A limitation of the Netkit software rather than the model itself – when this model is implemented in the real world this won't be the case and the IDS will automatically startup as a service when the `DmzGW` machine is booted. To re-install Suricata each time the packages to do so had to be pre-downloaded from the correct Ubuntu repositories and are now located in the machine's `/root/suricata_files/` directory as `*.deb` files. In order to install these files, and have Suricata operational, the `/root/install_suricata.sh` script has to be run whenever a user wants to run Suricata – this is the case because the installation requires user interaction and so cannot be put in the `DmzGW.startup` file. Once installed the `/root/run_suricata.sh` script can be executed which will get the IDS and machine correctly configured for operation. This script does several things:

- ◇ It disables packet offloading. Network cards (NICs), drivers, and the Linux kernel speed up packet handling using techniques like Large Receive Offload (LRO) and Generic Receive Offload (GRO) which lead to the merging of smaller packets into big packets that can break some Suricata rules (Open Information Security Foundation, 2021a). To prevent this functionality the `ethtool` utility can be used to disable GRO and LRO with the command:

```
ethtool -K eth0 gro off lro off
```

- ◇ Updates the `suricata.rules` file (the default Suricata rules file that detects malicious traffic)

◇ It restarts Suricata as a service

These actions, along with the correctly configured files already on the machine within the Netkit model's `DmzGW` directory, make sure Suricata is running correctly and is ready to detect malicious activity.

3.2.2 Rules Implemented – #3.8

In this Netkit model, Suricata has been implemented as an IDS solution which generates alerts based on potentially malicious activity and to do this it uses Suricata rules. These rules are defined in rule files which are listed in the Suricata configuration file `/etc/suricata/suricata.yaml` under the `rule-files` section. For this Suricata implementation the following rule files have been created:

suricata.rules This is the file is automatically generated by Suricata when the `suricata-update` command is used. It consists of all of the default Suricata rules which are located within the `/etc/suricata/rules` directory compiled into a single file for the Suricata engine to check packets against. It has been used in this model to provide a base-level of protection for assets within MEME's subnet against common attacks by generating alerts when malicious activity (indicative of an attack) is detected. The file can be found at: `/var/lib/suricata/rules/suricata.rules`

dos.rules This file contains rules which aim to detect if a Denial of Service (DoS/DDos) attack is targeting MEME assets within the DMZ zone. The rules in this file monitor the frequency of packets travelling to both the `Int-WWW` and `Mail` machines to detect if a DoS attack is occurring. The file can be found at: `/etc/suricata/dos.rules`.

test.rules This file is used for demonstration purposes in this report and should be discarded when this Netkit model is implemented in the real world. The file acts as a demonstration of Suricata's capabilities when implemented as an IDS and is used in section 3.2.3. The file can be found at: `/etc/suricata/test.rules`.

3.2.3 Evidence of IDS at Work

In order to demonstrate that Suricata has been successfully implemented as an IDS solution in this Netkit model, the `/etc/suricata/test.rules` has been created. This file contains rules which showcase Suricata's capabilities. To begin with, Suricata needs to be installed and running on the `DmzGW` machine:

```
# installing and running Suricata
root@DmzGW:~# ./install_suricata.sh
...
< Configuring libhyperscan5: >
    [Yes]
...
< Configuration file '/etc/suricata/suricata.yaml': >
    [N - keep your currently-installed version]
...
root@DmzGW:~# ./run_suricata.sh
...
root@DmzGW:~#
```

#3.9 With Suricata running, it's capabilities can be demonstrated. Firstly, Suricata can filter by IP address alone. For instance, if there is an IP address which is known to be controlled by a threat actor or a Command & Control (C2) server then an alert can be generated if it sends traffic through the IDS (Mohanta and Saldanha, 2020). A Suricata rule which generates an alert based on source IP address is shown below:

```
alert ip 22.39.224.18 any -> any any
    (msg:"notorious IP address";classtype:bad-unknown; sid:2100498;
    rev:7; metadata:created_at 2021_02_16, updated_at 2021_02_16;)
```

The notorious IP address here is `22.39.224.18` (`Ext-Office`) and this rule is shown in action in the terminal grabs below:

```
# send traffic to Int-WWW from notorious IP address
root@Ext-Office:~# nc -v www.meme.cyber.test 80
Connection to www.meme.cyber.test (137.205.98.155) 80 port [tcp/http]
    succeeded!
^C
root@Ext-Office:~#
# alert generated in Suricata fast.log file in real time
root@DmzGW:~# tail -f /var/log/suricata/fast.log
02/17/2021-11:34:49.825232  [**] [1:2100498:7] notorious IP address [**]
    [Classification: Potentially Bad Traffic] [Priority: 2]
    {TCP} 22.39.224.18:51308 -> 10.10.2.2:80
^C
root@DmzGW:~#
```

#3.10 Suricata can check for content within network traffic and generate an alert based on a match. For instance, a web hacking technique is to "directory walk" where an attacker will try and discover hidden (not indexed) directories/web pages on the web server by searching for them through URLs. Commonly, there is an administrator web page on a web server that allows the site's administrator to upload content remotely via HTTP or make database changes through web-based MySQL clients like phpMyAdmin. An attacker could search for this hidden web page by entering `http://www.meme.cyber.test/admin.html` and then try to log in as an administrator. To monitor attempts to access this directory, a Suricata rule can be written to generate alerts based on HTTP GET requests with the content "admin.html" included in the URL. A rule do to this is shown below:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
    (msg:"Web admin zone lookup"; content:"/admin.html"; http_uri;
    classtype:attempted-recon; sid:2100499; rev:7;
    metadata:created_at 2021_02_16, updated_at 2021_02_16;)
```

In this rule variables are used with which are initialised in the `suricata.yaml` configuration file. This rule is shown in action in the terminal grabs below:

```
# send traffic to Int-WWW from notorious IP address
root@Ext-Office:~# wget http://www.meme.cyber.test/admin.html
--2021-02-17 11:54:31--  http://www.meme.cyber.test/admin.html
Resolving www.meme.cyber.test (www.meme.cyber.test)... 137.205.98.155
Connecting to www.meme.cyber.test
    (www.meme.cyber.test)|137.205.98.155|:80 ... connected.
HTTP request sent, awaiting response... 404 Not Found
2021-02-17 11:54:31 ERROR 404: Not Found.

root@Ext-Office:~#
```

```
# alert generated in Suricata fast.log file in real time
root@DmzGW:~# tail -f /var/log/suricata/fast.log
02/17/2021-11:54:31.159292 [**] [1:2100499:7] Web admin zone lookup [**]
    [Classification: Attempted Information Leak] [Priority: 2]
    {TCP} 22.39.224.18:51314 -> 10.10.2.2:80
^C
root@DmzGW:~#
```

#3.11 Aside from demonstrating the potential of Suricata rules, this report also looks to showcase how Suricata can be used in conjunction with the Management Network in a real-world scenario. For instance, if a member of the onsite system administration team receives reports that MEME's web server is running slow, be it through telemetry data or an anecdotal report, they can first remotely check the IDS solution installed on the `DmzGW` machine to see if any malicious network activity has been detected. They can do this by first remotely logging into the `DmzGW` machine in a secure manner through SSH.

```
# remotely logging into DmzGW to check for IDS alert
root@admin1:~# ssh admin1@10.10.2.1
...
Enter passphrase for key '/root/.ssh/id_rsa':
Linux DmzGW 5.10.4 #1 Mon Jan 25 03:08:46 PST 2021 x86_64
Welcome to Netkit

admin1@DmzGW:~$
```

Then they can check the output of Suricata's `fast.log` to quickly check if an alert has been raised by the IDS engine.

```
# checking the fast.log generated by IDS for quick lookup
admin1@DmzGW:~$ cat /var/log/suricata/fast.log
02/17/2021-12:34:55.979810  [**] [1:5:0]
    LOCAL DOS SYN packet flood inbound, Potential DOS [**]
    [Classification: Misc activity] [Priority: 3]
    {TCP} 203.219.130.74:18569 -> 10.10.2.2:80
...
admin1@DmzGW:~$
```

As shown in the terminal grab, an alert for a SYN DoS attack has been raised as the network traffic has matched a rule in the `dos.rules` Suricata rule file. In this demo this attack was simulated with the `hping3` utility:

```
# generating a SYN DoS attack
hping3 -S -p 80 --flood --rand-source www.meme.cyber.test
HPING www.meme.cyber.test (eth0 137.205.98.155):
    S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- www.meme.cyber.test hping statistic ---
97904 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@Ext-Office:~#
```

And captured with the Suricata rules:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
    (msg:"LOCAL DOS SYN packet flood inbound, Potential DOS";
    flow:to_server; flags: S,12; threshold: type both, track by_dst,
    count 5000, seconds 5; classtype:misc-activity; sid:5;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any
```

```
    (msg:"LOCAL DOS SYN packet flood outbound, Potential DOS";
    flow:to_server; flags: S,12; threshold: type both, track by_dst,
    count 5000, seconds 5; classtype:misc-activity; sid:6;)
```

These rules are located within the `/etc/suricata/dos.rules` rule file. At this point the system administrator can take action and then extract the `eve.log` generated by Suricata. The `eve.log` is a thorough log of the alert in a easy-to-use JSON format which can be used to link with APIs or compile a report from. The terminal grab below shows this extraction and the installation of the `jq` utility on the `admin1` machine to view the alert in a nicely formatted manner.

```
# extracting the eve.log file generate by IDS and inspecting it
root@admin1:~# sftp admin1@10.10.2.1
...
Enter passphrase for key '/root/.ssh/id_rsa':
Connected to 10.10.2.1.
sftp> get /var/log/suricata/eve.json
Fetching /var/log/suricata/eve.json to eve.json
/var/log/suricata/eve.json              100%   12MB   8.8MB/s   00:01
sftp> exit
root@admin1:~# ./install_jq.sh
...
root@admin1:~# cat eve.json | jq 'select(.event_type=="alert")' | less
{
  "timestamp": "2021-02-17T12:34:55.979810+0000",
  "flow_id": 996310231544674,
  "in_iface": "eth1",
  "event_type": "alert",
  "src_ip": "203.219.130.74",
  "src_port": 18569,
  "dest_ip": "10.10.2.2",
  "dest_port": 80,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 5,
    "rev": 0,
    "signature": "LOCAL DOS SYN packet flood inbound, Potential DOS",
    "category": "Misc activity",
    "severity": 3
  },
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 0,
    "bytes_toserver": 54,
    "bytes_toclient": 0,
    "start": "2021-02-17T12:34:55.979810+0000"
  }
}
...
^C
root@admin1:~#
```

3.2.4 Future Work

The Suricata IDS in this Netkit implementation enhances the security posture of MEME by providing the organisation with IDS solution that monitors traffic flowing to/from the DMZ subnet where MEME's public-facing assets are located. It analyses this traffic for potentially malicious activity and then generates alerts which can be inspected remotely in a secure manner by the onsite system administration and security operators via the Management Network. However, Suricata can be expanded in several ways that will further improve MEME's security posture and make the service easier to manage.

Automatically send alerts to SOC In it's current configuration a member of the system administration team has to manually log into the `DmzGW` machine to view the logs that Suricata generates in response to suspicious network activity. This can slow down the response of the system administration team to any suspicious activity as they have to initiate interaction with the machine. However, Suricata can be configured to, instead, initiate an interaction with the system administration team when an alert is generated. For this to be effective MEME would need to setup a SOC (Security Operations Centre) or have a centralised repository where security alerts can be collated for inspection (i.e. a Redis server). With a central location setup, on the Management Network, Suricata can be configured to send it's output to this machine and then system administrators can use this machine to inspect alerts. This setup means that multiple operators can work on combing through the log files and whenever a new alert is generated the system administration team will be notified without having to log into the `DmzGW` machine. Suricata provides support for a variety of technologies that allow this automated logging setup to be created. This includes support for; Unix sockets, Redis servers, the Linux netfilter firewall, JSON output, and Lua scripting (Open Information Security Foundation, 2021b). That said, implementing a SOC, Redis server, SIEM, or automated Unix socket solution is outside the scope of this Netkit model and is left as future work due to the further infrastructure and time requirements it would entail.

Setting up Suricata as an IPS Currently, Suricata is setup as a IDS which generates alerts when suspicious network activity occurs. In this mode a system administration is only notified when a rule is triggered, rather than the rule blocking the malicious traffic that triggered it. To block malicious traffic Suricata needs to be setup in IPS/inline for Linux mode where it operates at layer 3 and uses the `iptables` `NFQUEUE` option to force the machine to delegate filtering decisions to Suricata by marking packets. Once `iptables` has been configured, Suricata can be run in NFQ mode with the following command:

```
suricata -c /etc/suricata/suricata.yaml -q 0
```

When Suricata is acting as an IPS, rather than an IDS, rules can be created that "drop" packets rather that just generating an "alert". This functionality can be effective in blocking sophisticated malware attacks that have the potential to cause significant damage to organisational assets. However, if an IPS generates a false positive the legitimate traffic will be blocked, whereas if an IDS generates a false positive then only an alert will be created. This is the fundamental trade-off between IDS and IPS systems with IPS solutions often requiring a higher degree of sophistication and fine tuning to be effective, as the decision to block traffic is taken out of a human's discretion. As such, in this Netkit model it was decided that running Suricata as an IDS will still substantially improve the organisation's security posture, without causing potential self-inflicted denial of service, and implementing Suricata as an IPS is left for future work.

Create a web-based interface for Suricata The final piece of future work which this report believes would benefit MEME is implementing a web-based interface for the Suricata IDS. This piece of future work ties in with the first one, automatically sending alerts to a SOC, but is more of a convenience rather than something that will drastically improve MEME's security posture. By having configuration options available in a web-based GUI means that system administrators can more easily make changes without having to deal with text-based configuration files where they are more prone to making a syntax mistake. In addition to configuration changes, a web-based GUI would provide easier to digest logging information which can be neatly formatted through scripts because of the support Suricata has for outputting logs in JSON format. Being able to output logs in an easy to read way will mean it is easier for the system administration team to spot anomalies and reduce the chance that alerts will go unnoticed or be skipped over. Implementing a web-based interface is left to future work because it does not necessarily improve MEME's security posture, instead it is a feature which will make the system administration team's job easier.

3.3 Phase 3 Summary

Reference & Claim – sections 3.1 & 3.2	Evidence
Ref: #3.1 A Management Network has been created that uses secure SSH server configurations	SSH server configuration changes outlined in section 3.1, the `/etc/ssh/sshd_config` file on each MEME asset that required remote administration, and appendix B.1
Ref: #3.2 A Management Network has been created that uses secure SSH client configurations	SSH client configuration changes outlined in section 3.1, the `/etc/ssh/ssh_config` file on the `admin1` machine, and appendix B.2
Ref: #3.3 The Management Network can be used to remotely upload new files to a MEME asset in a secure manner through a SFTP and public key authentication	Terminal grabs in section 3.1.3
Ref: #3.4 The Management Network can be used to remotely make DNS configuration changes to a MEME asset in a secure manner through a SSH and public key authentication	Terminal grabs in section 3.1.3
Ref: #3.5 The Management Network can be used to remotely make firewall configuration changes to a MEME asset in a secure manner through a SSH and public key authentication	Terminal grabs in section 3.1.3
Ref: #3.6 An IDS has been implemented which is tailored to MEME's network environment	The `/etc/suricata/suricata.yaml` configuration file on the `DmzGW` machine
Ref: #3.7 Installation and configuration script have been included to automatically setup Suricata with minimal user interaction	The shell scripts `/root/install_suricata.sh` & `/root/run_suricata.sh` and the terminal grabs in section 3.2.3
Ref: #3.8 Several custom Suricata rule files have been created to enhance the IDS's ability to detect malicious activity	The Suricata rule files `/etc/suricata/dos.rules` and `/etc/suricata/test.rules` outlined in section 3.2.2
Ref: #3.9 Suricata has been configured to detect traffic coming from a notorious IP address	The terminal grabs in section 3.2.3 and the Suricata rules in `/etc/suricata/test.rules`
Ref: #3.10 Suricata has been configured to detect attempted web reconnaissance through directory walking by monitoring packet content	The terminal grabs in section 3.2.3 and the Suricata rules in `/etc/suricata/test.rules`
Ref: #3.11 Suricata can be used in real-time to detect SYN DoS attacks through the Management Network and download JSON-formatted log files	The terminal grabs in section 3.2.3, the Suricata rule file `/etc/suricata/dos.rules`, and the Management Network setup in 3.1

Table 3.1: Phase 3 Work Summary Table

References

Beaming Support (2018). *Understanding the Spanning Tree Protocol (STP)*. [online] Available from: https://www.beaming.co.uk/knowledge-base/understanding-spanning-tree-protocol-stp/ (Accessed 26 January 2021).

Boelen, M. (2019). *OpenSSH security and hardening*. [online] Available from: https://linux-audit.com/audit-and-harden-your-ssh-configuration/ (Accessed 11 February 2021).

Brotherston, L. and Berlin, A. (2017). *Defensive Security Handbook: Best Practices for Securing Infrastructure*. Sebastopol, CA: O'Reilly Media.

Camara, U. A. and Baudy, J. (2020). *Packet Mmap*. [online] Available from: https://www.kernel.org/doc/Documentation/networking/packet_mmap.txt (Accessed 17 February 2021).

Gilman, E. and Barth, D. (2017). *Zero Trust Networks: Building Secure Systems in Untrusted Networks*. Sebastopol, CA: O'Reilly Media.

Internet Engineering Task Force (1996). *Address Allocation for Private Internets*. [online] Available from: https://tools.ietf.org/html/rfc1918 (Accessed 8 February 2021).

— (2012). *IANA-Reserved IPv4 Prefix for Shared Address Space*. [online] Available from: https://tools.ietf.org/html/rfc6598 (Accessed 8 February 2021).

Kahn, A. A. (2009). *Configuring a Squid Server to authenticate off LDAP*. [online] Available from: https://wiki.squid-cache.org/ConfigExamples/Authenticate/Ldap (Accessed 9 February 2021).

Mohanta, A. and Saldanha, A. (2020). *Malware Analysis and Detection Engineering*. California: Apress.

National Institute of Standards and Technology (2015). *Hash Functions*. [online] Available from: https://csrc.nist.gov/projects/hash-functions/nist-policy-on-hash-functions (Accessed 11 February 2021).

Open Information Security Foundation (2021a). *9.2 Packet Capture*. [online] Available from: https://suricata.readthedocs.io/en/suricata-6.0.1/performance/packet-capture.html (Accessed 17 February 2021).

— (2021b). *Suricata — All features*. [online] Available from: https://suricata-ids.org/features/all-features/ (Accessed 17 February 2021).

— (2021c). *Suricata User Guide*. [online] Available from: https://suricata.readthedocs.io/en/suricata-6.0.1/index.html (Accessed 17 February 2021).

— (2021d). *Suricata: Open Source IDS / IPS / NSM engine*. [online] Available from: https://suricata-ids.org/ (Accessed 11 February 2021).

Pleeger, C. P., Pfleeger, S. L., and Margulies, J. (2015). *Security in Computing*. 5th ed. Upper Saddle River, NJ: Prestige Hall.

Scaife, J. (2020). *How To Harden OpenSSH on Ubuntu 18.04*. [online] Available from: https://www.digitalocean.com/community/tutorials/how-to-harden-openssh-on-ubuntu-18-04 (Accessed 11 February 2021).

Shaw, G. (2018). *Capture Ethernet frames using an AF_PACKET ring buffer in C*. [online] Available from: http://www.microhowto.info/howto/capture_ethernet_frames_using_an_af_packet_ring_buffer_in_c.html (Accessed 17 February 2021).

Squid (2013). *About Squid*. [online] Available from: http://www.squid-cache.org/Intro/ (Accessed 1 February 2021).

— (2016). *Proxy Authentication*. [online] Available from: https://wiki.squid-cache.org/Features/Authentication (Accessed 9 February 2021).

— (2020). *Authenticate with NCSA httpd-style password file*. [online] Available from: https://wiki.squid-cache.org/ConfigExamples/Authenticate/Ncsa (Accessed 9 February 2021).

SSH.COM (2021). *sshd_config – SSH Server Configuration*. [online] Available from: `https://www.ssh.com/ssh/sshd_config/` (Accessed 11 February 2021).

Wilson, N. (2018). *LDAP.com – Lightweight Directory Access Protocol*. [online] Available from: `https://ldap.com/` (Accessed 26 January 2021).

A SNAT and DNAT Firewall Rules

A.1 Evidence of SNAT & DNAT Firewall Rules on Perimeter Firewall

```
root@PerimF:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target  prot opt source    destination
DNAT    tcp  --  anywhere  137.205.98.155   multiport dports 80,443  to:10.10.2.2
DNAT    tcp  --  anywhere  137.205.139.114  multiport dports 25,587  to:10.10.2.3
DNAT    udp  --  anywhere  137.205.89.215   udp dpt:1194             to:10.10.4.2
DNAT    tcp  --  anywhere  137.205.89.215   tcp dpt:1194             to:10.10.4.2
DNAT    udp  --  anywhere  137.205.89.220   udp dpt:51820            to:10.10.4.3

...

Chain POSTROUTING (policy ACCEPT)
target     prot opt source      destination
MASQUERADE all  --  10.10.3.3    anywhere
MASQUERADE all  --  10.10.2.3    anywhere
MASQUERADE all  --  10.10.4.0/24 anywhere
SNAT       tcp  --  10.10.2.2    anywhere multiport sports 80,443 to:137.205.98.155
SNAT       tcp  --  10.10.2.3    anywhere multiport sports 80,443 to:137.205.139.114
SNAT       udp  --  10.10.4.2    anywhere udp spt:1194            to:137.205.89.215
SNAT       tcp  --  10.10.4.2    anywhere tcp dpt:1194            to:137.205.89.215
SNAT       tcp  --  10.10.4.3    anywhere multiport sports 80,443 to:137.205.89.220
root@PerimF:~#
```

The output of the `iptables -t nat -L` on `PerimF` to show the SNAT & DNAT firewall rules present on this machine – some output has been cut for brevity.

B SSH Configurations

B.1 Evidence of SSH Server Configuration on MEME Assets

```
# SSH server settings running
root@RemoteAccessGW:~# sshd -T
port 22
addressfamily any
listenaddress [::]:22
listenaddress 0.0.0.0:22
usepam yes
logingracetime 20
x11displayoffset 10
maxauthtries 3
maxsessions 10
clientaliveinterval 0
clientalivecountmax 3
streamlocalbindmask 0177
permitrootlogin no
ignorerhosts yes
ignoreuserknownhosts no
hostbasedauthentication no
hostbasedusesnamefrompacketonly no
pubkeyauthentication yes
kerberosauthentication no
kerberosorlocalpasswd yes
kerberosticketcleanup yes
gssapiauthentication no
```

```
gssapicleanupcredentials yes
gssapikeyexchange no
gssapistrictacceptorcheck yes
gssapistorecredentialsonrekey no
gssapikexalgorithms [...]
passwordauthentication no
kbdinteractiveauthentication no
challengeresponseauthentication no
printmotd no
printlastlog yes
x11forwarding no
x11uselocalhost yes
permittty yes
permituserrc yes
strictmodes no
tcpkeepalive yes
permitemptypasswords no
compression yes
gatewayports no
usedns no
allowtcpforwarding no
allowagentforwarding no
disableforwarding no
allowstreamlocalforwarding no
streamlocalbindunlink no
fingerprinthash SHA256
exposeauthinfo no
pidfile /run/sshd.pid
xauthlocation /usr/bin/xauth
ciphers aes128-ctr,aes192-ctr,aes256-ctr
macs hmac-sha2-256,hmac-sha2-512
banner /etc/ssh/MEME_banner
forcecommand none
chrootdirectory none
trustedusercakeys none
revokedkeys none
securitykeyprovider internal
authorizedprincipalsfile none
versionaddendum none
authorizedkeyscommand none
authorizedkeyscommanduser none
authorizedprincipalscommand none
authorizedprincipalscommanduser none
hostkeyagent none
kexalgorithms ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,
    diffie-hellman-group-exchange-sha256,diffie-hellman-group16-sha512,
    diffie-hellman-group18-sha512,diffie-hellman-group14-sha256
casignaturealgorithms [...]
hostbasedacceptedkeytypes [...]
hostkeyalgorithms ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,
    ssh-rsa,ssh-dss
pubkeyacceptedkeytypes [...]
loglevel INFO
syslogfacility AUTH
authorizedkeysfile /etc/ssh/%u/authorized_keys
hostkey /etc/ssh/ssh_host_rsa_key
hostkey /etc/ssh/ssh_host_ecdsa_key
hostkey /etc/ssh/ssh_host_ed25519_key
allowusers *@10.10.5.0/24
acceptenv LANG
acceptenv LC_*
authenticationmethods any
subsystem sftp /usr/lib/openssh/sftp-server
```

```
maxstartups 10:30:100
permittunnel no
ipqos lowdelay throughput
rekeylimit 0 0
permitopen any
permitlisten any
permituserenvironment no
pubkeyauthoptions none
root@RemoteAccessGW:~#
```

The output of the `sshd -T` command on `RemoteAccessGW` to show the SSH server configurations for this machine. The main changes which have been changed are highlighted in red and some unchanged configurations has been redacted for brevity. Note, the SSH server configurations on this machine are representative of all MEME networked assets which require remote administration functionality.

B.2 Evidence of SSH Client Configuration

```
# SSH default client settings running
root@admin1:~# ssh -G a
user root
hostname a
port 22
addressfamily any
batchmode no
canonicalizefallbacklocal yes
canonicalizehostname false
challengeresponseauthentication yes
checkhostip yes
compression no
controlmaster false
enablesshkeysign no
clearallforwardings no
exitonforwardfailure no
fingerprinthash SHA256
forwardx11 no
forwardx11trusted yes
gatewayports no
gssapiauthentication no
gssapikeyexchange no
gssapidelegatecredentials no
gssapitrustdns no
gssapirenewalforcesrekey no
gssapikexalgorithms [...]
hashknownhosts no
hostbasedauthentication no
identitiesonly no
kbdinteractiveauthentication yes
nohostauthenticationforlocalhost no
passwordauthentication no
permitlocalcommand no
proxyusefdpass no
pubkeyauthentication yes
requesttty auto
streamlocalbindunlink no
stricthostkeychecking ask
tcpkeepalive yes
tunnel false
verifyhostkeydns false
visualhostkey no
updatehostkeys false
```

```
canonicalizemaxdots 1
connectionattempts 1
forwardx11timeout 1200
numberofpasswordprompts 3
serveralivecountmax 3
serveraliveinterval 0
ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc
hostkeyalgorithms [...]
hostbasedkeytypes [...]
kexalgorithms [...]
casignaturealgorithms [...]
loglevel INFO
macs hmac-sha2-256,hmac-sha2-512
securitykeyprovider internal
pubkeyacceptedkeytypes [...]
xauthlocation /usr/bin/xauth
identityfile /root/.ssh/id_rsa
canonicaldomains
globalknownhostsfile /etc/ssh/ssh_known_hosts /etc/ssh/ssh_known_hosts2
userknownhostsfile /root/.ssh/known_hosts /root/.ssh/known_hosts2
sendenv LANG
sendenv LC_*
addkeystoagent false
forwardagent no
connecttimeout none
tunneldevice any:any
controlpersist no
escapechar ~
ipqos lowdelay throughput
rekeylimit 0 0
streamlocalbindmask 0177
syslogfacility USER
root@admin1:~#
```

The output of the `ssh -G a` command on `admin1` to show the SSH client configuration on this machine – `a` is a placeholder for a host, but since no host `a` exists it means the default settings will be parsed. The main changes which have been changed are highlighted in red and some unchanged configurations has been redacted for brevity.