Secure Network Solution for ACME Scandinavia

1 Analysis of ACME Scandinavia Requirements

1.1 Access to Internal Network of Stockholm

Employees at the office branch in London should have access to the internal network, and so should company laptops located outside the Stockholm and London offices. Additionally, the connection has to be secure and not exploitable by adversaries, where employees also should be able to hide their activity from external observers.

1.2 Critical and Non-Critical Secure Web Server

There should be two different types of web servers. One which is considered critical and, thus, should only be accessible when a computer is on-site at the Stockholm or London branch networks (not when accessing remotely). Secondly, there should be a less critical but still secure web server that is accessible both on-premise and remotely. All connections to and from all servers have to be secured.

1.3 Verifiable Digital Identities for Company Devices

All company devices should be identifiable based on unique digital identities that can be verified using a third party, a CA. These digital identities should be secure and easily revoked in case they are compromised.

1.4 Authentication of Company Laptops

Company laptops connected to the internal network using the on-premise wireless network or remotely using a tunnel should be authenticated based on their digital identity. Employees should additionally be able to hide their activity from third parties when traveling.

1.5 Authentication of Personal Devices

For personal devices, only a handful of web servers should be accessible, that is, the non-critical ones. Thus, access from devices that do not have a digital identity will be authenticated through a 2FA service on the employees' phones.

1.6 Secure File Exchange and Communication

Employees shall be able to share files and communicate with each other securely.

1.7 Logging of Network Traffic and Attack Prevention

Logging of network traffic, as well as requests to the web servers, shall be performed, allowing analysis and review of traffic. Additionally, intrusions and anomalies should be detectable (such as an Intrusion Detection System (IDS)) so that proper action can be taken. Furthermore, clogging DoS attacks has to be prevented.

1.8 Secure Domain and Domain Lookup

The domain needs to be secure in that it cannot be spoofed and thereby misdirecting users. Furthermore, employees should not be tricked into entering incorrect websites while connected to the internal network.

2 Proposed Implementation

2.1 Network Topology

Seen in Figure 1 is the network topology of the proposed network solution. The physical internal network is assumed secure. For demonstrative purposes, the domain "acme.xilef.win" will be used in place of ACME's own.

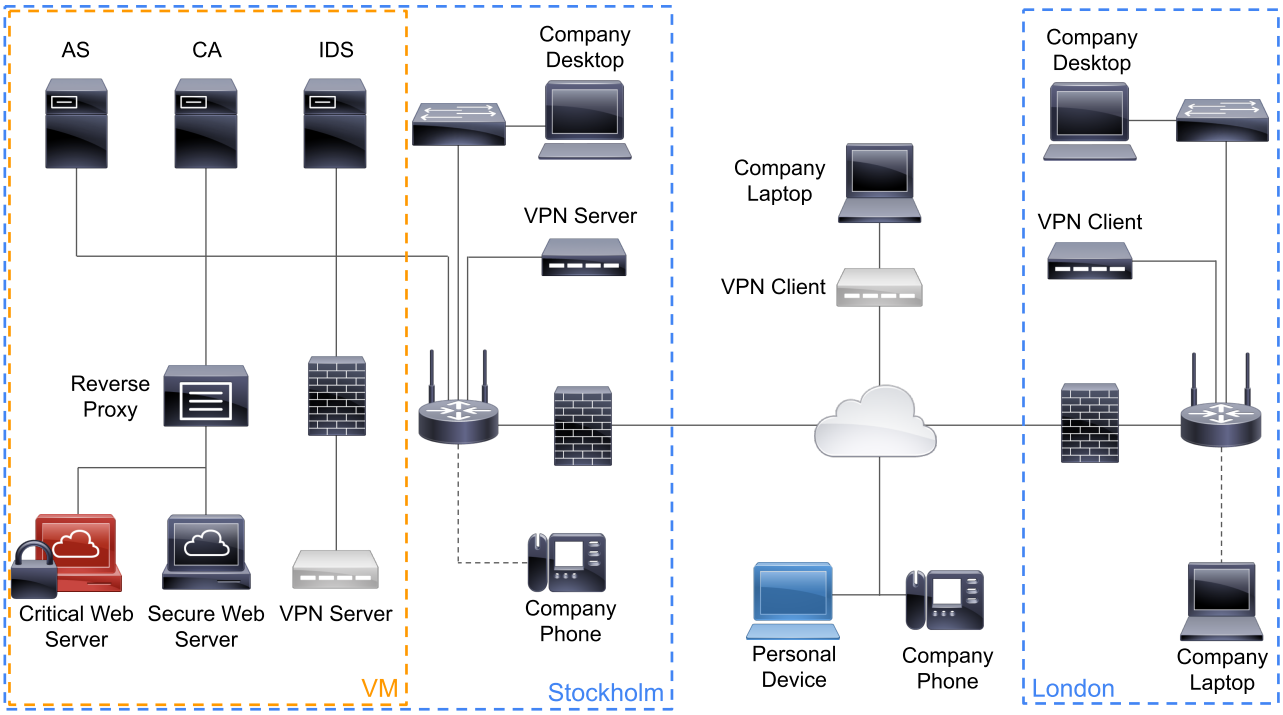

Figure 1: *Network topology of the proposed solution where "Company Phone" is the device for 2FA, light gray VPN server/client are for remote workers, and dark gray VPN server/ client are for site-to-site.*

2.2 Server Software

A single main VM is used to run separate containers for different services. The VM is running the latest version of the Ubuntu server, securing against known vulnerabilities. Services are hosted in separate Docker/LXD containers on the VM to provide isolation between them and to make sure that even if one container is compromised, the data and integrity of the remaining containers and the VM itself are not compromised. For managing the Docker containers in a GUI, Portainer is used. A single VM with multiple containers minimizes the overhead while still maintaining isolation between the services. The VM will run on a machine that is directly connected to an interface on the Stockholm router.

2.3 Access Control

To differentiate between different levels of access, the following classes have been defined:

- Privileged (root) access to the VM
- Unprivileged (non sudo account) access to the VM
- Access to a container (without access to the VM)
- Access to the internal network (connected to the network, including desktops and authenticated wireless devices)
- Access to the network over VPN (working remotely without access to the critical web server)
- Users outside the network (can still access the secure web server with 2FA over HTTPS)

2.4 Routers

The routers are flashed with OpenWRT, which includes tools for access control such as OpenVPN, OpenSSL, and a firewall. Thus, providing an all-in-one solution covering all the security tools needed for the routers. Devices connected to the internal network of the London or Stockholm office, either wireless or wired, are connected to the Internet using DHCP configuration in the OpenWRT routers. Since a single Authentication Server (AS) is used for authentication (RADIUS server), the wireless connection is secured using WPA2 Enterprise. While WPA2 does not provide quite the same security as WPA3, greater compatibility among devices is deemed necessary for the demonstration.

The Stockholm subnet is configured to use the IP address block 10.1.0.0/16, where the router has the address 10.1.0.1 and the main server (the VM hosting all the containers) 10.1.0.2, with the first 100 addresses being reserved for static leases. Similarly, for London, the IP address block is 10.2.0.0/16 with the router using 10.2.0.1 where, also here, the first 100 addresses are reserved for static leases. Remote workers on VPNs use addresses from the 10.8.0.0/24 block, where 10.8.0.1 is the VPN interface on the OpenVPN server inside the main VM.

In the hosts files of the Stockholm and London router, an entry for "critical.acme.xilef.win" has been added so that the critical web server can be reached locally through said domain.

2.5 VPN

Two different VPN implementations are used, both being OpenVPN. Firstly, a VPN between the Stockholm and London branch, server and client respectively, is securely tunneling traffic to and from the London branch to achieve secure connectivity between the two subnets. The implementation is using OpenVPN directly in the OpenWRT routers, where a shared secret between the two OpenVPN instances is used to secure the channel. Secondly, another OpenVPN server, hosted in an LXD container inside the VM in Stockholm, allows remote workers to connect to the internal network and, thus, also allows them to hide their activity from external observers. Users connect to the OpenVPN server inside the container and the VM by authenticating themselves using their certificate. A Certificate Revocation List (CRL) is pushed to the VPN server instance, allowing access to devices to be revoked.

2.6 Firewall

Three firewalls are in total used, two of them are built into OpenWRT and protect the two branches from external threats. These two firewalls are configured to, by default, only allow forwarding between LAN and the site-to-site VPN, while traffic from WAN is dropped and to WAN is rejected. From LAN, only HTTPS is allowed to establish outgoing connections from either of the two sites to maximize security. Thereby, firewall configurations based on specific needs may be necessary. The Stockholm router does, however, port forward relevant connections to the remote workers' VPN server and the reverse proxy, as well as allowing the site-to-site VPN to be established from London. To prevent direct insecure access to the secure web server and critical web server, that is not going through the reverse proxy mentioned in Section 2.10, the Stockholm firewall blocks these requests and thereby forces the reverse proxy to be used (the VM machine is directly connected to the Stockholm router).

The third firewall is placed on the OpenVPN server for remote workers mentioned in Section 2.5, also blocking direct access to both the secure web server and the critical web server. The Stockholm firewall is not able to block such direct access since the remote VPN server is already on the same physical machine as the web servers. Remote workers are, thereby, forced to go through the reverse proxy where rules regarding access to the critical web server are placed, see Section 2.12.

2.7 Intrusion Detection System (IDS)

To be able to detect potential intrusions and to log network traffic for later evaluation, the application SNORT is used. Two SNORT instances are deployed as two standalone Docker containers on the main VM server, logging and creating alerts based on the monitored packets. The first instance monitors the physical interface to the VM machine while the second one monitors the VPN connections from remote workers (packets from such VPN clients may never leave the VM machine and, thus, never traverse the physical interface).

Alerts generated by the two SNORT instances are saved to two respective files that can be accessed at any time by an administrator. These alerts are generated by a set of rules created by the SNORT community, as well as some specific rules created specifically for the proposed network solution. Additional rule sets or changes to existing ones can be easily made as different needs emerge. The specific rules for the proposed network solution include general rules that, for example, detect suspicious connection attempts to hosted services. Furthermore, rules that only log traffic, instead of creating alerts, have also been added. With these logging rules, both SNORT instances log all packets that traverse their

respective interface and save them to individual PCAP files that can be opened and analyzed in a sniffer program such as Wireshark (the file sizes remain small even after longer periods).

2.8 Authentication Server (AS)

A RADIUS server is hosted in an LXD container in Stockholm, running FreeRADIUS, providing a central and scalable point of authentication, such as for wireless access in Stockholm and London. The FreeRADIUS server will be able to take a CRL into account during the authentication of devices.

2.9 Certificate Authority (CA)

The CA is implemented directly in the VM, as it has the highest security class and consist of a set of scripts that use OpenSSL to sign and issue certificates for the devices of employees. The scripts push the CA certificate and the current CRL to the VPN server for remote connections and the FreeRADIUS server. The self-hosted CA will allow certificates to be configurable based on needs, but also easily revoked if a device would be lost. These certificates will act as the central form of authentication for all services to avoid relying on passwords, which can be an inherent weakness in overall security.

The certificates are assumed to be created and installed by an administrator, both for practical reasons and for security. Security would be greatly improved as only the administrator would know the password that is used to encrypt the private key accompanying the certificate, thus preventing it to be shared between different devices.

2.10 Reverse Proxy

A reverse proxy is hosted on a separate Docker container on the VM, essentially placed in front of the two web servers. All communication with the web servers is performed through the reverse proxy as all other methods are blocked by either the Stockholm firewall router or the remote VPN server. All communication with the reverse proxy is mandated to use HTTPS and, thus, all communication with it is secure as the traffic between the reverse proxy and the servers happens locally on the VM machine. Access control to the critical web server is performed by restricting IP addresses that have access to it, only allowing on-site addresses and thereby not devices on a remote VPN.

2.11 Secure Web Server

The secure web server is running Nextcloud and is hosted in a Docker container, providing an user-friendly all-in-one secure file exchange and communication solution between employees through an integrated website. The secure web server can be accessed, through the reverse proxy, both inside and outside the network by entering the domain “acme.xilef.win” in a web browser. The domain “acme.xilef.win” is secured using a Let’s Encrypt certificate instead of ACME as it is trusted by browsers in general and, thus, each personal device does not need to have the ACME CA certificate installed. Authentication of users in Nextcloud will be performed using personal login credentials as well as 2FA on employee phones, see Section 2.13 regarding the latter.

2.12 Critical Web Server

The critical web server, also hosted in a Docker container, is an HTTPS web server for demonstrative purposes. Authentication is indirectly performed as it is only locally accessible through “critical.acme.xilef.win” (exists only locally), but also by the reverse proxy access rules mentioned in Section 2.10. Similar to the secure web server, a Let’s Encrypt certificate is used when connecting to “critical.acme.xilef.win”.

2.13 Two-Factor Authentication (2FA)

2FA is implemented on the Nextcloud secure web server using the “Two factor TOTP” Nextcloud app which works with Google Authenticator used on the phones, allowing for a simple 2FA configuration onto an already existing solution. All users are required to authenticate using 2FA after entering their login credentials to the Nextcloud server as bypassing the 2FA based on IP address would open up the server to a larger attack surface.

2.14 Preventing Clogging DoS Attacks

To prevent TCP SYN flooding (a type of clogging attack), TCP cookies have been enabled on both the secure web server and the reverse proxy. Furthermore, OpenWRT in the routers has built-in SYN flooding protection and will also,

considering the firewall, filter out other types of clogging DoS attacks targeting the network, with potential exceptions having to be dealt with by contacting the ISP to try and redirect the traffic.

2.15 Secure DNS

The configured DNS resolver is Cloudflare's 1.1.1.1, a secure and publicly available one. Since it is hosted by Cloudflare, it has security practices that exceed the requirements of ACME. Furthermore, the DNS port number 53 will be, indirectly, blocked at the firewall of both routers, forcing the use of DoH from the Stockholm or London router to the Cloudflare DNS resolver to secure Internet activity. The domain used for demonstrative purposes, a subdomain of an already owned domain, has DNSSEC explicitly enabled, allowing requests to ACME's domain to be validated. Included in the domain price is also free DNS hosting where the DNS server is provided by the domain register, in this case Cloudflare, and is used to secure the domain.

Appendix

A Technical Details

A.1 Routers

A.1.1 Software

The routers of model *NETGEAR AC1750 Smart Wi-Fi Router (R6350)* has been flashed with OpenWRT using `nmrpflash`. DoH have been enabled through the use of the `https-dns-proxy` and `luci-app-https-dns-proxy` OpenWRT packages where Cloudflare's 1.1.1.1 has been specified as the DNS resolver. The preinstalled WPAD implementation required modifications to include RADIUS, resulting in that `wpad-openssl` had to be uninstalled and replaced by installing `wpad` instead. Furthermore, to implement OpenVPN in both routers, the packages `openvpn-openssl` and `luci-app-openvpn` were installed, where UDP was configured as the used protocol over port 42069 with the server in Stockholm using IP address 10.200.200.3 and the client in London 10.200.200.1. To create the shared secret between Stockholm and London, the command `openvpn --genkey --secret static.key` was used separately, with the resulting `static.key` file being imported to both OpenWRT instances. For the site-to-site VPN to work, static routes had to be added to each router to forward any traffic destined for 10.1.0.0/16 to 10.200.200.3 and 10.2.0.0/16 to 10.200.200.1, respectively.

To be able to demonstrate the network solution on KTH LAN, more precisely to get the routers to acquire a public IP address on the LAN, their MAC addresses have to be overridden. Since the supplied USB Ethernet adapters have MAC addresses that are configured to receive an IP address on the LAN, their MAC addresses were copied and entered into the OpenWRT GUI to manually assign them to the WAN interface card on the routers.

A.1.2 Firewall Configuration

Both firewalls have had their DDoS protection enabled (TCP cookies). To achieve the specified rules for the Stockholm and London router in the OpenWRT GUI, the general settings of the nftables are to drop any incoming packets from the WAN zone as well as those that are forwarded within the WAN zone via the router. These rules allow the firewall configurations to be better hidden from the WAN (drop instead of reject). The site-to-site VPN is configured as its own firewall zone. Since Stockholm and London are to be considered the same internal network, the LAN zone has been set to, by default, allow the forwarding of all traffic to the VPN zone and vice versa.

External access to the remote OpenVPN server and the secure web server in Stockholm is enabled through two port forwarding rules. These are UDP on port 1194 to be forwarded to 10.1.0.2 on port 1194, as well as UDP and TCP connections over port 443 are forwarded to 10.1.0.2 on port 443 (the reverse proxy), respectively. Other types of access are performed using traffic rules, more specifically the site-to-site VPN connection establishment from London to Stockholm is enabled by allowing UDP connections over port 1194 with the source IP of the London WAN interface and the Stockholm router as a destination (no forwarding at the router). After the VPN between the sites has been established, said rule can be disabled if desired. Finally, outgoing HTTPS connection establishments are allowed over port 443 from both Stockholm and London LAN.

To prevent direct access to the web servers locally, either from LAN or the VPN interfaces, two additional traffic rules have been implemented. These are to reject packets with destination address 10.1.0.2 and destination port 7777 or 8080 on the LAN interface. The two rules force the reverse proxy to be used for accessing the web servers and, thus, enforce the use of HTTPS when accessing them.

A.2 OpenVPN Container Firewall

To prevent direct access to both the secure web server and the critical web server for remote workers (not going through the reverse proxy), rules in the OpenVPN container's iptables are added on the Linux container, blocking packets destined for port 7777 or 8080 originating from remote VPN connections. Furthermore, an Access List is added to the reverse proxy to restrict access to the critical web server for remote VPN clients, see Appendix A.7 for details.

A.3 RADIUS

The configuration files for RADIUS are located at `/home/john/freeradius/raddb`. Here it has been configured to use the supplied certificates and CRL from the CA, while any other authentication methods have been disabled. RADIUS is deployed in an LXD container, where details can be found in Appendix A.9.

A.4 Two-Factor Authentication (2FA)

2FA enforcement on Nextcloud has been configured under "Settings" and then the "Security" tab under "Administration" when logged in to an admin account. When 2FA enforcement is enabled, new users will be forced to set up 2FA on their first login. Here, they can set up a Time-Based One-Time Password (TOTP) using a QR code and an authenticating mobile application, such as Google Authenticator or Authy.

A.5 Certificates and CA

Certificates and private keys are created using scripts to make it much easier to maintain and use them, as well as to make sure that all files that may be needed are created. These files include, for example, different file formats for compatibility but also OVPN files for remote OpenVPN clients. The private keys created by the scripts have a length of 4096 bits and use RSA encryption. The created certificates are valid for 365 days to minimize the amount of administrative work, but this is not a security issue due to them being easily revoked. Furthermore, if wanted to, the valid period can be easily configured. Both FreeRADIUS and OpenVPN require Diffie-Hellman parameters. These parameters, which are set to be 2048 bits long, are recreated each time the CRL is updated to minimize the possibility of brute-force attacks. They are pushed to the corresponding services together with the CRL during the corresponding script.

A.6 Nextcloud

Nextcloud Talk, a video call application in Nextcloud, can be used between employees that are connected to the same network, such as the company wide internal network.

A.7 Reverse Proxy

The reverse proxy is an Nginx proxy manager (NPM) instance running in a Docker container. Two different hosts have been added, one for "acme.xilef.win" and one for "critical.acme.xilef.win" where the latter is not a public domain and, thus, will only work when accessing it from the LAN. To prevent access to the critical web server from remote users, an Access list was made in NPM. This application layer firewall only allows users with IP ranges in 10.1.0.0/16 (Stockholm) and 10.2.0.0/16 (London) to access it. Since remote workers get IP addresses in the 10.8.0.0/24 range, they will get an HTTP 403 response from the reverse proxy when attempting to access the critical web server.

The reverse proxy communicates with the two web servers using HTTP, which is not secure. However, since no packets are actually sent as they are only passed between different virtual Ethernet interfaces, the packets will not be able to be captured by any adversaries.

A.8 Docker containers

The hosted Docker containers have been created according to the following settings, where "Image" specifies the used image (originating from the Docker Hub Container Image Library), "Port(s)" opened ports with the written format `HOST PORT:CONTAINER PORT`, "Bind mounts" the directory at the host that is linked to a directory in the container (`PATH IN HOST` → `PATH IN CONTAINER`), and finally which Docker volumes are assigned and to where with the following format `VOLUME NAME` → `PATH IN CONTAINER`.

Reverse proxy

- Image: `jc21/nginx-proxy-manager`
- Port(s): `443:443` & `80:80`
- Bind mounts: `/home/john/proxy` → `/data`
- Volumes: `proxt_certs` → `/etc/letsencrypt`

Nextcloud

- Image: `nextcloud`
- Port(s): `8080:80`
- VOLUME: `Data` → `/var/www/html`

nginx_critical

- Image: `nginx`
- Port(s): `7777:80`
- Bind mounts: `/home/john/criticalWebServer` → `/usr/share/nginx/html`

DDNS

- Image: `oznu/cloudflare-ddns`

Portainer

- Image: `portainer/portainer-ce`
- Port(s): `8000:8000` & `9100:9443`
- Bind mounts: `/var/run/docker.sock` → `/var/run/docker.sock`
- Volumes: `portainer_data` → `/data`

SNORT

- Image: `linton/docker-snort`
- NETWORK: `HOST`
- Port(s): `ALL (Automatic with host network)`
- Bind mounts:
  `/var/log/snort → /var/log/snort`
  `/home/john/snort/config -> /home/snort`

A.9 Linux Containers (LXD)

Name: `Radius`
`$(RADIUS) = /etc/freeradius/3.0/`
`$(DATABASE) = /home/john/freeradius/raddb`
Image: `ubuntu:22.04`
Commands that were used for setting it up:

1. `sudo apt install freeradius freeradius-utils -y`
2. `sudo systemctl enable freeradius`

Configured devices:

- sharedModsAvaiableEAP:
  path: `$(RADIUS)/mods-available/eap`
  source: `$(DATABASE)/mods-available/eap`
  type: `disk`
- sharedSitesAvaiableDEFAULT:
  path: `$(RADIUS)/sites-available/default`
  source: `$(DATABASE)/sites-available/default`
  type: `disk`
- sharedSitesAvaiableTLS:
  path: `$(RADIUS)/sites-available/tls`
  source: `$(DATABASE)/sites-available/tls`
  type: `disk`
- sharedCerts:
  path: `$(RADIUS)/certs`
  source: `$(DATABASE)/certs`
  type: `disk`
- sharedClientsConf:
  path: `$(RADIUS)/clients.conf`
  source: `$(DATABASE)/clients.conf`
  type: `disk`
- port1812:
  connect: `udp:10.10.250.75:1812`
  listen: `udp:10.1.0.2:1812`
  type: `proxy`

Name: `OpenVPN`
Image: `ubuntu:22.04`
Commands that were used for setting it up:

1. `wget https://git.io/vpn -O openvpn-ubuntu-install.sh`
2. `chmod -v +x openvpn-ubuntu-install.sh`
3. `sudo ./openvpn-ubuntu-install.sh`

Configured devices:

- certFolder
  path: `/etc/openvpn/cert/`
  source: `/home/john/openvpn`
  type: `disk`
- port1194
  connect: `udp:10.10.250.167:1194`
  listen: `udp:10.1.0.2:1194`
  type: `proxy`

To add the extra firewall rule in the iptables, the file `/etc/systemd/system/openvpn-iptables.service` had to be changed. This file is created by the installation script and implements the firewall rules on startup and removes them on shutdown. For each iptables rule that is to be added, an `ExecStart` and `ExecStop` command have to be added. For example, to block the TCP connections directly to the critical web server, the following two additions had to be made:

```
ExecStart=/usr/sbin/iptables -I FORWARD -s 10.8.0.0/24 -p tcp --dport 7777 -j REJECT
ExecStop=/usr/sbin/iptables -D FORWARD -s 10.8.0.0/24 -p tcp --dport 7777 -j REJECT
```

The first addition inserts the rule (added above already inserted rules) when the service is booted. The second one removes it if the service is restarted (preventing duplicate rules).

README.md

Routers

The Stockholm and London routers can be configured by logging into them, while on the internal network, through `10.1.0.1` and `10.2.0.1`, respectively.

Portainer

The Portainer GUI interface to manage and configure the Docker containers can be accessed by logging in at the local address `https://10.1.0.2:9100`. In Portainer you can create, recreate, remove, run, stop, and configure docker containers. A guide on how to use Portainer can be found in the Portainer documentation[1].

Certificates (CA)

Create CA + Server certificates

After having logged into the VM using a user with sudo permission, run the command `sudo -i` to log in as root. Scripts and configuration files for managing certificates can be found in `/home/john/CA/` (only accessible by root). To create a new CA instance, also destroying any existing CA (for example if it has been compromised), run

```
./createCA.sh
```

This creates a new private key and corresponding certificate for the root CA. Created are also private keys and certificates for the FreeRADIUS and OpenVPN servers, which are automatically copied to the correct location for each respective service.

Create and Revoke Users

A new user certificate can be generated using the `./createUser.sh` script where a username can be passed as a parameter, for example,

```
./createUser.sh <Username>
```

The script creates different certificate file formats for compatibility, as well as an OVPN file for a remote OpenVPN client. From the files created, the legacy (name ending with `LEG`) `.p12` file has the greatest compatibility and is, thus, recommended to be installed in the client device. The legacy `.p12` file and the OVPN file both include the certificate and the private key. Thereby, the respective file, combined with the installation of the CA certificate (`ca.pem`), allows the device to be authenticated to the corresponding service. Since the created `.p12` file contains an encrypted private key, they are password protected. The password is set in the script, and is, for demonstrative purposes, the word `password` followed by the username. For example, a user called `Steve` would have the password `passwordSteve`. For non-root users (for example, network admins with SSH access to the server) to be able to access the newly created certificate files, run the following command to change the permission of the directory created by the `./createUser.sh` script as well as moving it outside the current directory

```
./exportUsers
```

To revoke a user, run the script `revokeUser.sh` and pass a username as a parameter to the script

```
./revokeUser.sh <Username>
```

Create CRL

For the revocation of a user to take effect, a CRL has to be created and pushed to the correct locations. A list of certificates and their status, where also the accompanying usernames can be seen for easy revocation, can be found in `index.txt`. To create the CRL containing all revoked certificates and push it to the FreeRADIUS and OpenVPN server, as well as restart the services to apply the changes, run

```
./createAndCopyCRL.sh
```

Configure Certificates

The user information and parameters of the certificates can, before being created, be configured directly in the respective `./createUser.sh` and `./createCA.sh` scripts, as well as the two corresponding configuration files `client.cnf` and `ca.cnf`.

[1] `https://docs.portainer.io/user/home`

Reverse Proxy

To access/configure the reverse proxy, log in at `http://10.1.0.2:81`. For instructions, see the Nginx Proxy Manager Guide[2]. To start/restart/stop the container, see the Portainer documentation.

Web Servers

Secure Web Server (Nextcloud)

Nextcloud starts automatically with the VM. To configure the container and change boot settings, etc. see the Portainer README for information and tutorials. To configure Nextcloud, see the Nextcloud User Manual[3].

The Nextcloud web server can be reached securely from a web browser through `https://acme.xilef.win`.

Critical Web Server

The critical web server can be accessed securely on-site through `https://critical.acme.xilef.win`.

LXD Containers

There are two different Linux containers as seen in the technical description. To operate them, the following commands are used

```
lxc start <ContainerName>
lxc stop <ContainerName>
lxc restart <ContainerName>
```

Similarly, to connect to the containers for configuration, use

```
lxc console <Name>
```

Both containers are configured to start on boot using the following nixCraft tutorial[4].

SNORT

SNORT is started on boot and can be configured and managed through the Portainer GUI. The alerts generated by SNORT, as well as network traffic logs, can be found in the corresponding files in `/home/var/snort/` for the instance monitoring the physical interface connected to the router, and `/home/var/snort2/` for the one monitoring the remote VPNs. The alerts are saved in clear text and the logs are in PCAP format, which can be opened using, for example, Wireshark.

All used rules can be found in the `/home/john/snort/config/rules/` directory. Additional rules specific to the ACME network implementation can be added or changed inside `local.rules` (including the logging rules). Entire rule sets can be added to SNORT by adding the `.rules` file to the `/home/john/snort/config/rules/` directory and then editing `snort.conf` in the `/home/john/snort/config/` directory. The name of the `.rules` file should be added under section 7 in `snort.conf`. After making changes to the rules, restart the SNORT instances in the Portainer GUI. Any changes to the rules are applied to both SNORT instances.

VPN

The site-to-site VPN can be configured directly in the OpenWRT GUI by logging into each respective router.

Clients can connect to the OpenVPN server by installing `OpenVPN Connect` on their device and then create a new profile using their OVPN file provided when creating their user certificate (the `./createUser.sh` script). The OVPN file is simply dragged into the GUI when creating a new profile, or by opening the file directly if `OpenVPN Connect` is installed.

Wireless Authentication

To connect to the wireless network in either Stockholm or London, install both the ACME CA certificate and the user certificate to the device and enter the password that was used to encrypt the private key. Then connect to the Wi-Fi using TLS and select the certificate(s) according to the specific fields. If a domain is asked for verification, such as for Android, enter `radiusServer`.

[2] `https://nginxproxymanager.com/guide/`
[3] `https://docs.nextcloud.com/server/latest/user_manual/en/`
[4] `https://www.cyberciti.biz/faq/how-to-auto-start-lxd-containers-at-boot-time-in-linux/`