Part 1: Theoretical Assessment

Secure Access to HTTPS-Based Service

In the following, two scenarios describe how one could secure the access to a HTTPS-based service which runs as a VM on an IaaS platform. For context, the VM runs in a private network which must not be exposed directly to the Internet. Furthermore, the client who initiates the connections has to be authenticated by an IdP.

The first approach makes use of a reverse proxy. The reverse proxy acts as a middle-man between the VM and client and intercepts any incoming traffic from clients. Following, the VM is not exposed directly to the internet and instead only the reverse proxy is.
Moreover, one can configure a reverse proxy to ensure client authentication via SSO using a publicly available identity provider.
In order to provide more security and some kind of two factor authentication, the reverse proxy could be configured to use mutual TLS to solely allow connections to clients with valid client certificates. Additionally, mTLS could be also implemented in the HTTPS-bases service itself.

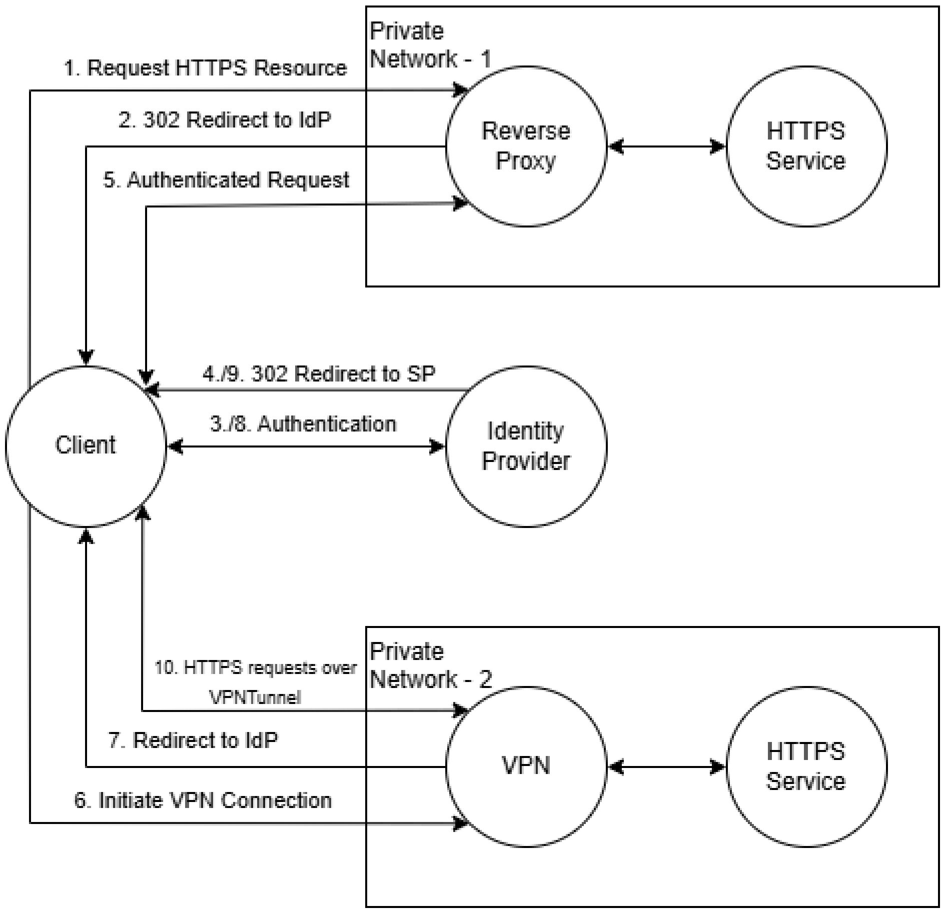

Figure 1: VPN and Reverse Proxy Approach (simplyfied)

In contrast to the first approach, one could implement a VPN server as an entrypoint to the HTTPS service. Similar to the first approach, only the VPN server would be exposed to the public internet and you could configure the VPN to use SSO with a publicly available identity provider. Furthermore, one can also make use of client certificates in the VPN directly or the HTTPS service itself.

Using a VPN requires the user to install a suitable VPN client. Moreover, in order to establish a connection to the HTTPS service , the user uses this VPN client to initiate a VPN connection (Step 6. Figure 1). Following, she is redirected to the IdP for authentication (Step 7.) and is then prompted by the IdP for authentication (Step 8.). After successful authentication, the user is provided with an authentication token which is used to authenticate itself at the VPN server. When the server accepts the token, it establishes a tunneled connection with the client using IPSec. Within this connection, the client is able to send HTTPS requests to the HTTPS service (Step 10.).

When a reverse proxy is implemented, the client is not required to install any additional software on its device. Following, she could use its browser to send a HTTPS request to the publicly available IP-Address or domain name (Step 1). When the client is not authenticated, the reverse proxy redirects the client to the IdP for authentication (Step 2). At the IdP, she authenticates itself (Step 3.) and is redirected back to the reverse proxy (Step 4.) to access the HTTPS service (Step 5.).

It is mentioned that one could also deploy a firewall in front of the service for further security.

Access the HTTPS service from within a terminal session

When a client wants to send requests to the HTTPS service through a terminal session (e.g. with curl) and the IdP only supports Browser-based authentication, you could try to extract the authentication token which is delivered by the IdP after authentication from the browser. To simplify this example, we expect the service to be protected by a reverse proxy.

The “naive” approach would be to first, request a resource of the HTTPS service via a browser to be redirected to the IdP for authentication. Furthermore, the client is required to record the requests and responses with, for example, the integrated browser tools. After authentication, the client could review the response for the authentication token, extract it and add it to the authentication headers of its terminal requests.

Another approach would be to create an automated tool that receives the redirect response from the reverse proxy and uses, for example, selenium to create a browser window with that redirect response and automatically perform the authentication and extraction of the token. Finally, the extracted token could be printed out on the terminal for further use. Depending on the identity prover, the usage of an automatically controlled browser window could be prevented by, for example, captchas or other security measures against bots.

User restrictions for the use of DNS and TLS

In both approaches described in the first section, the user is required to accept the Certificate Authority (CA) which signed the TLS-Certificate of the HTTPS service and reverse proxy or VPN server. Furthermore, when mTLS is used, the client is required to have a signed client certificate from a CA that is accepted the VPN, reverse proxy, and HTTPS service.

In regards to the reverse proxy approach, the client can make use of public DNS servers to resolve the host name of the reverse proxy if it has a domain name. Otherwise, the client would be required to use the public IP-Address of the proxy directly or configure its *etc/hosts* in order to resolve a domain name to the proxies IP-Address.

When using a VPN, the client has the same limitations to connect to the VPN server. However, the client cannot use public DNS to resolve the domain name of the HTTPS service. In order to connect to the service, the client would have to know and use the private network IP-Address of the HTTPS service. In order to make the service accessible through a domain name, an additional DNS has to be deployed within the private network. Furthermore, the client must configure this DNS on its device, for example, in its *etc/resolve.conf*.

Part II Practical Assessment

VM Image hardening

Task 1

The following table displays all rules that could not be configured or verified by openscap. The complete before and after reports can be found in the Git-Repository.

Nr	RuleID	Status	Comment
1	xccdf_org.ssgproject.content_rule_grub2_password	Fail	Rule has to be executed manually.
2	xccdf_org.ssgproject.content_rule_partition_for_tmp	Fail	No automated partitioning.
3	xccdf_org.ssgproject.content_rule_nftables_ensure_default_deny_policy	Not Checked	Cannot be performed during packer build.
4	xccdf_org.ssgproject.content_rule_set_nftables_base_chain	Not Checked	Cannot be performed during packer build.

5	xccdf_org.ssgproject.content_rule_set_nftables_loopback_traffic	Not Checked	Cannot be performed during packer build.
6	xccdf_org.ssgproject.content_rule_set_nftables_table	Not Checked	Cannot be performed during packer build.
7	xccdf_org.ssgproject.content_rule_package_rsync_removed	Error	Automated evaluation failed. Manual evaluation states a successful execution
8	xccdf_org.ssgproject.content_rule_package_telnet_removed	Error	Automated evaluation failed. Manual evaluation states a successful execution

Rule 1 has to be executed manually in order to set the grub2 bootloader. Passwords are stored in plaintext in readable files which allows other users to achieve unforeseen privileges such as sudo.

Rule 2 Mounting /tmp on its own partition allows the user/administrator to set the flags *nosuid* and *noexec* which prevents privilege-escalation attacks. This must be done on systems serving multiple users.

Rule 3 to 6 have not been executed to not disturb the required remote connection during the packer build. **Rule 3** provides a default deny policy so that all network traffic is denied. This rule secures the system from unwanted and unrequired network traffic which highly reduces the attack surface from the internet and mitigates the risk of accidentally exposing service to a network.
Rule 4 checks and creates nfttables base chains. Without these chains, nfttables would not be able to check network traffic.
Rule 5 mitigates that services that do not run on localhost interface ignore traffic from that inferface in order to prevent spoofing attacks. In our system we have to apply exceptions because we run different services on localhost and public services that have to interact with each other.
Rule 6 creates a default rule set table. Similar to rule 4, nfttables would not function without a basic rule set table.
Rule 7 and 8 seem to be false negatives. Upon manual inspection, both packages have been removed from the operating system.

Task 2

In order to run the VM image in Azure without Azure Agent, the **--enable-agent false** has to be set in the azure create command.

Setup a remote access service with Boundary

Task 1

The following table provides an overview of settings to secure the Apache 2 HTTPS service besides enabling SSL/TLS and restrictive netfilter rules.

Setting	Reason
ServerTokens Prod/ ServerSignature Off	Prevent information leaks about the server OS and Apache version
AllowOverride None	Locks down the current configuration to prevent reading a file that might be placed by an attacker.
Require ip 127.0.0.1	Restricts access from localhost to prevent any unwanted access
RequestReadTimeout	Set request timeout to prevent DoS attacks by sending slow requests.
Add headers to requests. X-XSS-Protection X-Content-Type-Options X-Frame-Options Content-Security-Policy Referrer-Policy Strict-Transport-Security	Add security Headers to responses to prevent XSS attacks, enforcing the same-origin-policy, preventing iframe embedding and enforcing HTTPS connections.