HTTPS Inspection

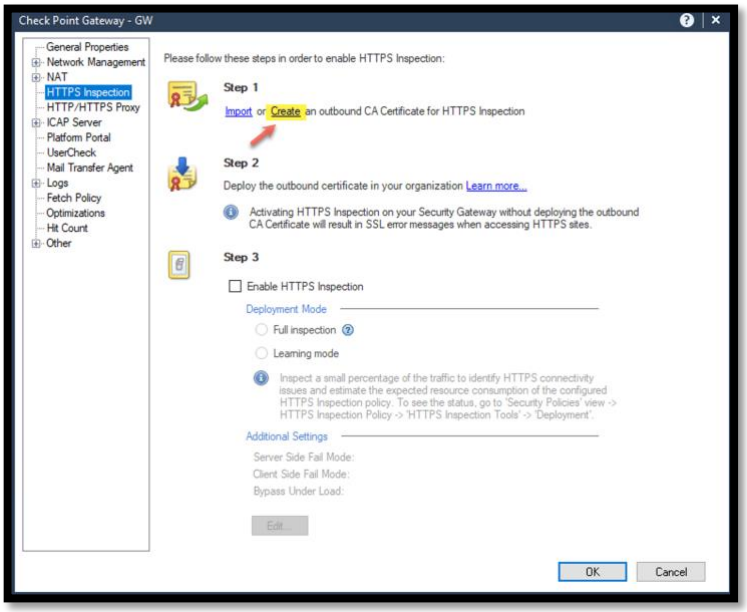
Introduction

In this lab, we will enable the HTTPS Inspection blades. HTTPS Inspection adds the capabilities to decrypt and inspect encrypted HTTPS sites.

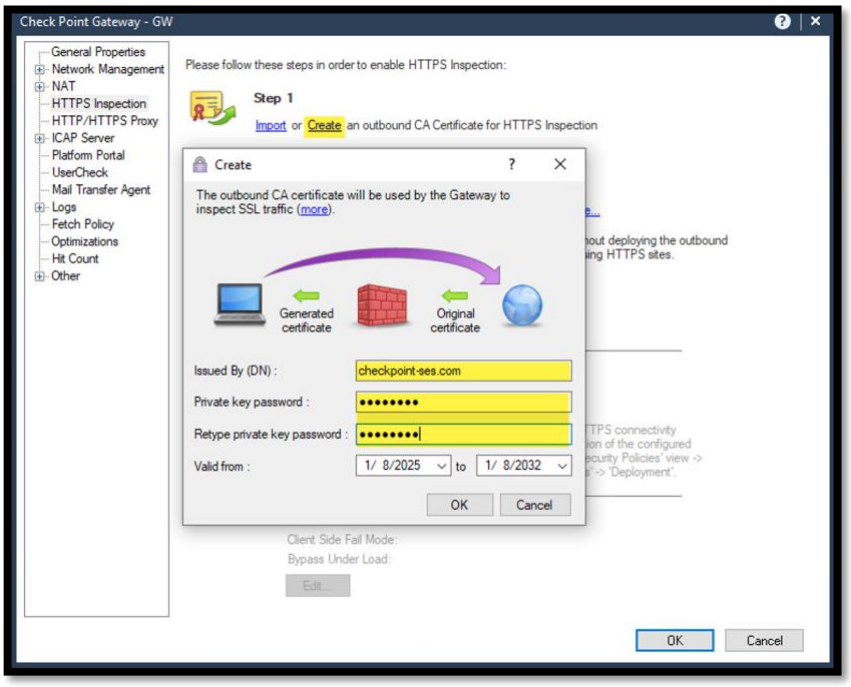
Exercise 1: Onboarding

In this exercise, we will enable the HTTPS Inspection blades on the central gateway object **GW**.

1. While connecting to the jump server, use **SmartConsole** to login to the Management server **SMS**. Use the address **10.1.1.100** and the credentials ***admin/Cpwins!1*** and edit the gateway object **GW**. From the Global Properties menu, select **HTTPS Inspection**.

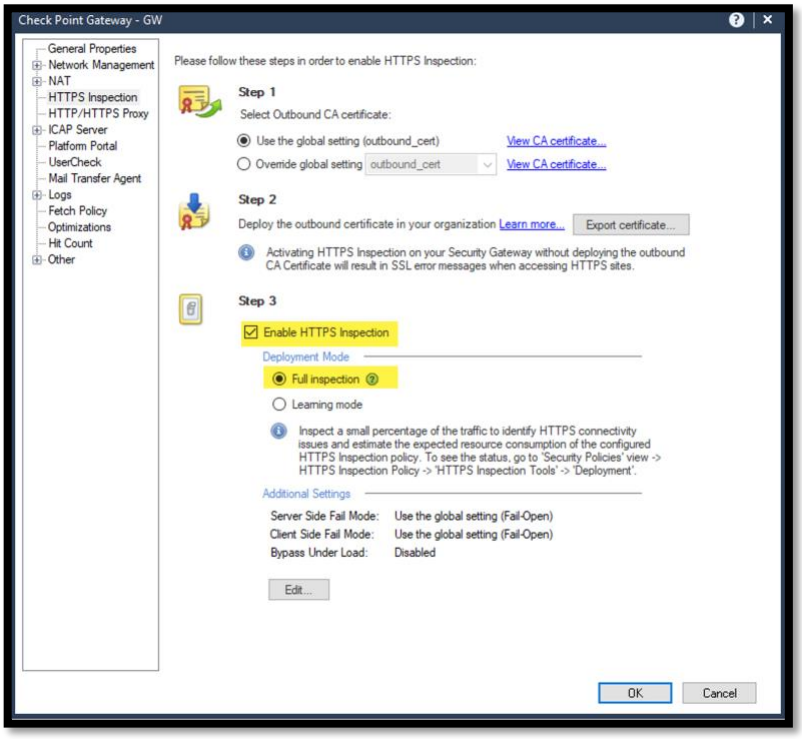


2. Under **Step 1,** click **Create** and fill in the details for the root certificate and click OK to create the certificate.
    a. Note that this certificate is a self-signed CA. This certificate is untrusted publicly by default.
    b. Generally, public vendors will not be able to provide their own trusted Certificate Authority certificate.
    c. It is common to import a trusted CA from an internal CA. for example, in an AD environment, the trusted CA from the domain controller is imported to issue certificates. The issued certificate will be trusted by all hosts that are part of this AD domain.

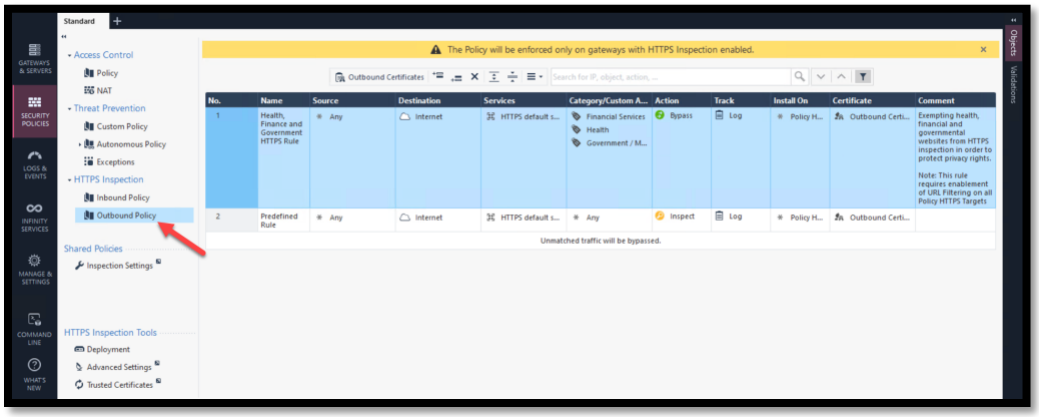


Note that this certificate will be used by the GW to issue certificate mimicking the server certificate while inspect HTTPS traffic.

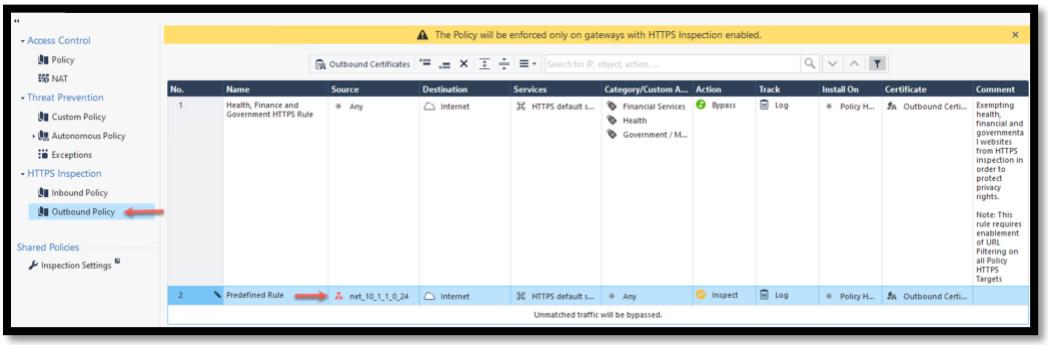
3. Skip **Step 2** for now, we will export the certificate in a later step. Under **Step 3** check the option **Enable HTTPS Inspection** and select **Full Inspection.**



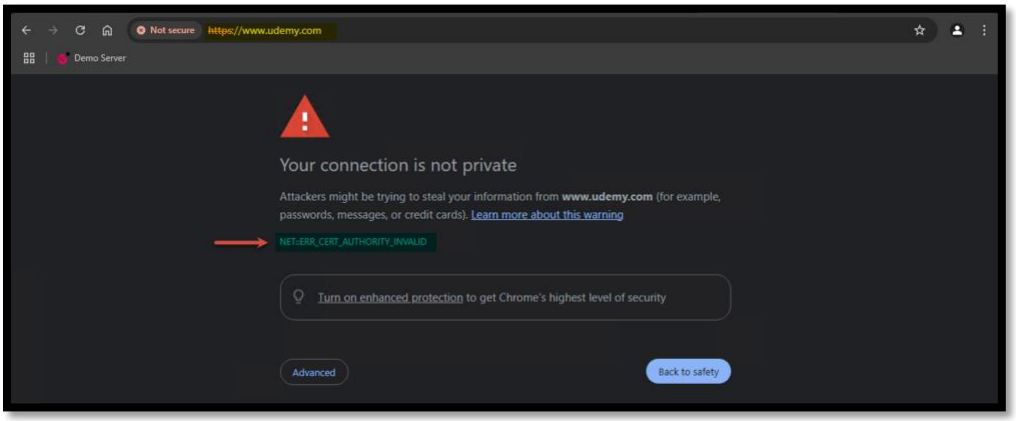
4. Under **HTTPS Inspection**, review the default **Outbound Policy**. Notice the categories bypassed by the first rule. The second rule **inspect** all outbound web traffic by default.



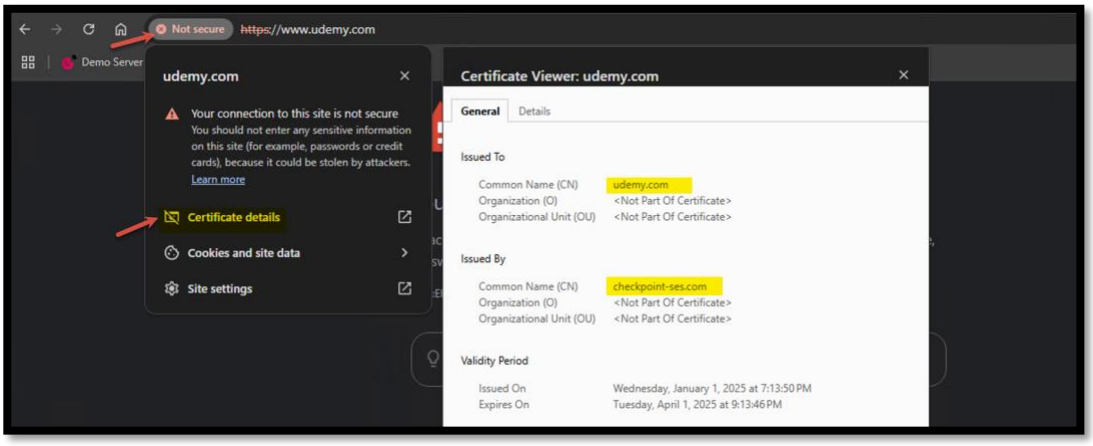
5. We will enable HTTPS inspection for the internal subnet **10.1.1.0/24** only. Modify the second rule and add **net_10_1_1_0_24** as the source of the rule.



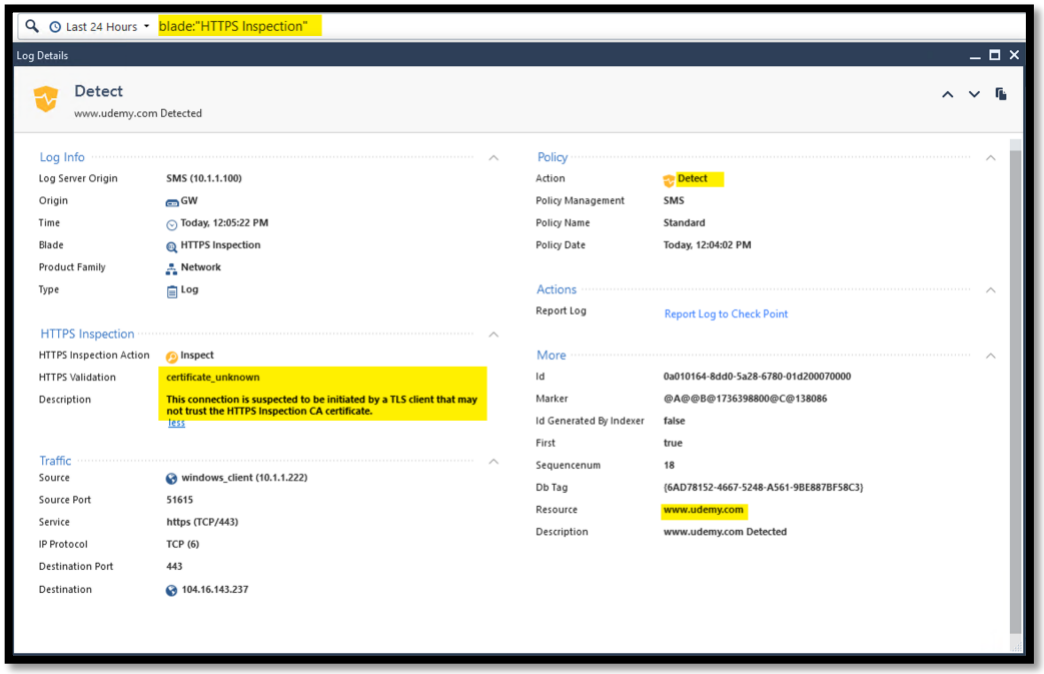
6. Install the Access Control Policy.
7. From **win_client**, open a web browser and try to access any allowed website. E.g. https://www.udemy.org. Notice that we are presented with a certificate trust warning.



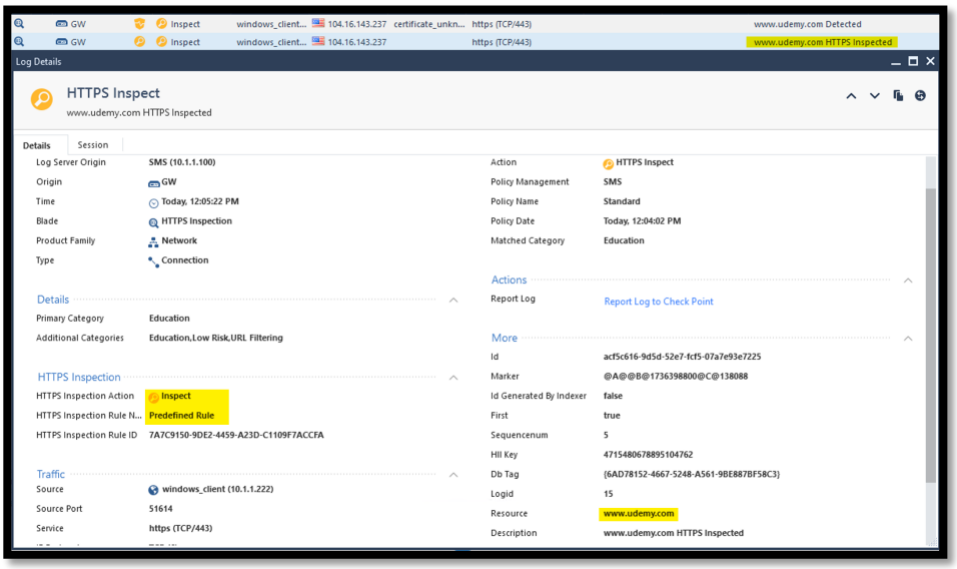
8. Bypass the security warning by clicking “Proceed to …” (if missing type *thisisunsafe* to bypass the warning). Open the certificate and notice that the certificate presented is a certificate issued by the GW using the certificate we created in the steps above.



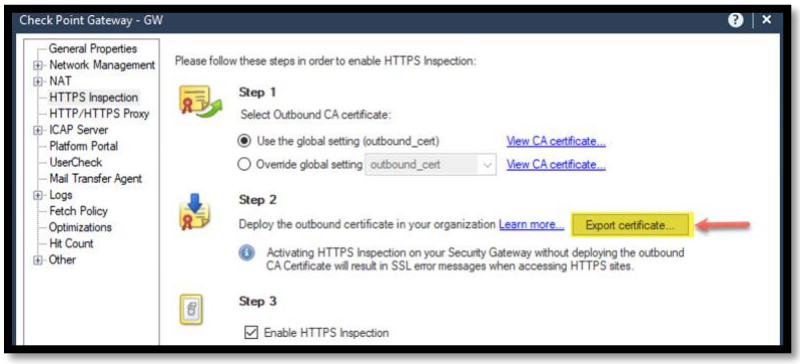
9. From **SmartConsole**, filter the logs to show logs related to HTTPS Inspection **blade: “HTTPS Inspection”**.
10. Review the related HTTP Inspection log and notice that the GW logged a warning related to the certificate being untrusted by the client. Notice that the log type is **Detect**.



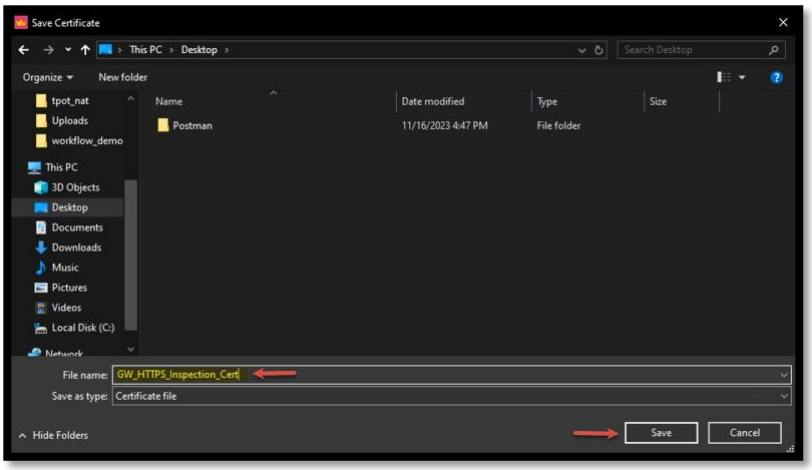
11. Review other security log related to the site we tested. https://www.udemy.com in the example above. Notice that the traffic was inspected successfully.



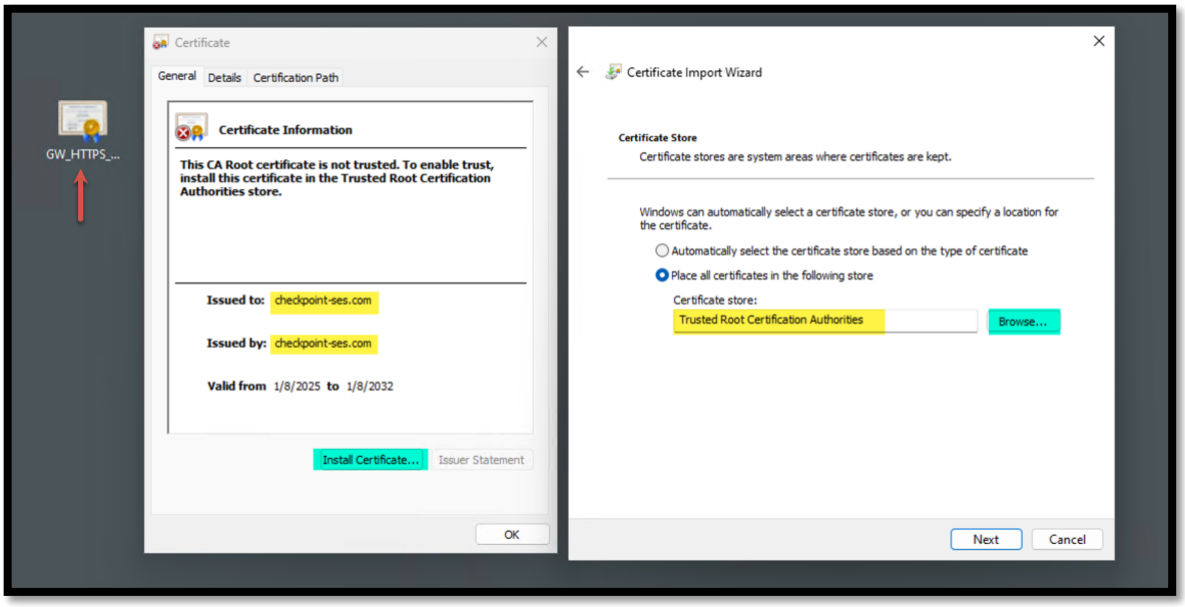
12. To avoid getting the certificate warning, open the GW object and export the HTTPS inspection certificate.



13. Give the certificate a proper name and save it to the Desktop of the Jump Server.

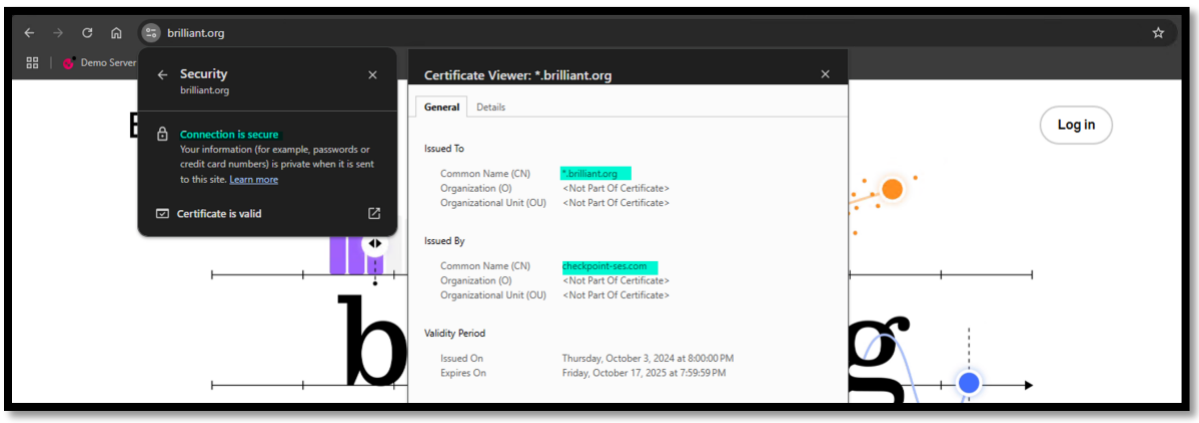


14. Copy the certificate file to the **win_client** host (over RDP) and install it as a Trusted Root Certification Authority. Make sure the certificate is installed successfully.

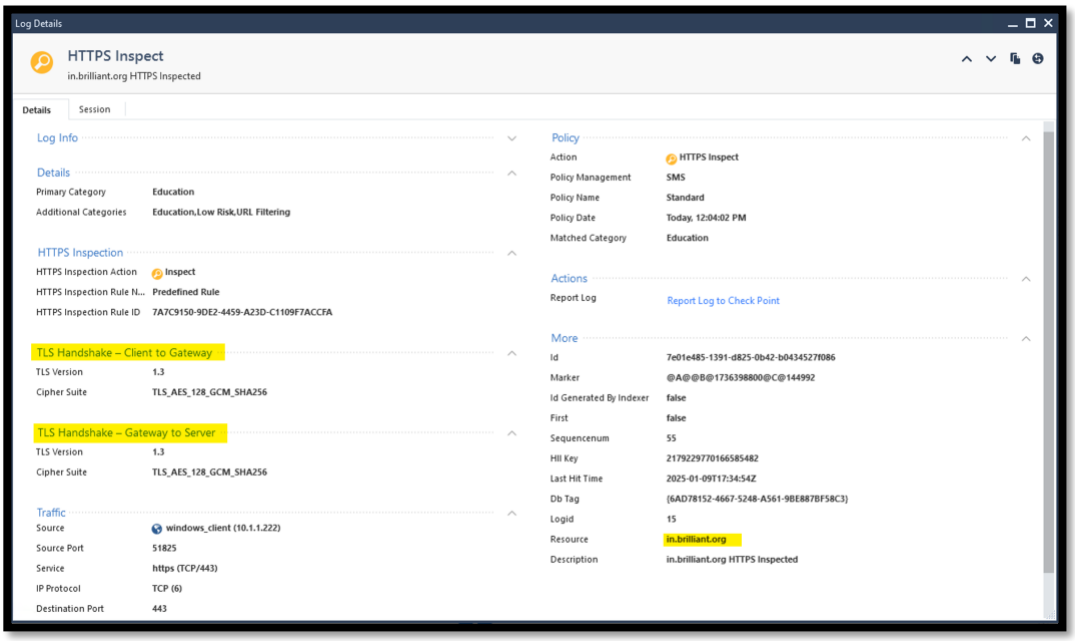


- Note that Chrome uses the Windows Certificate Trust Store while **Firefox** has its own certificate authority store that the CA key must be imported into.

15. Use any web browser like chrome and try reaching any allowed website again, e.g. https://briliant.org. Notice that the certificate warning is no longer present and the Connection is considered secure by the browser.

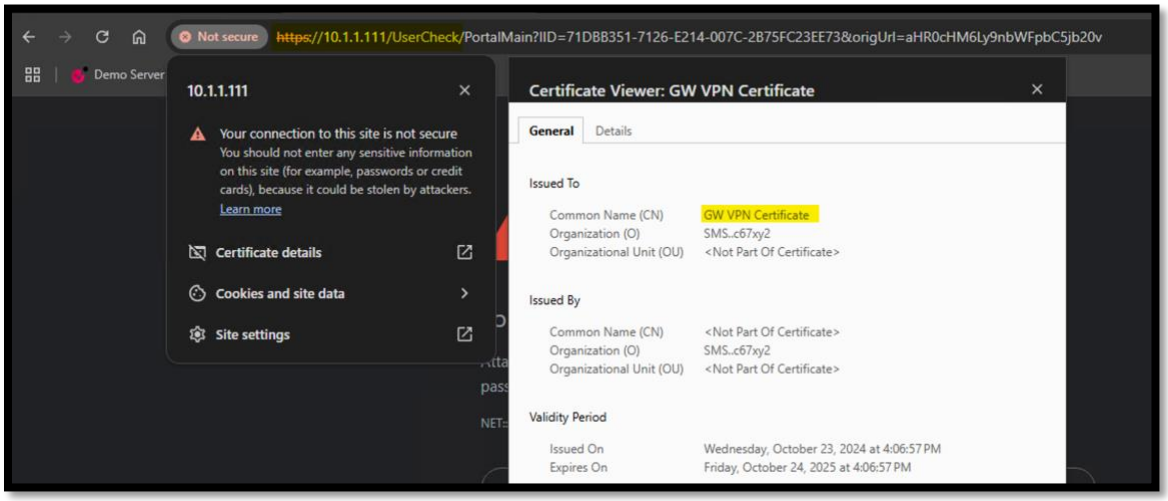


16. Review the HTTPS Inspection log. Notice that we can see details related to the TLS version and the cipher used on the client and the GW side.

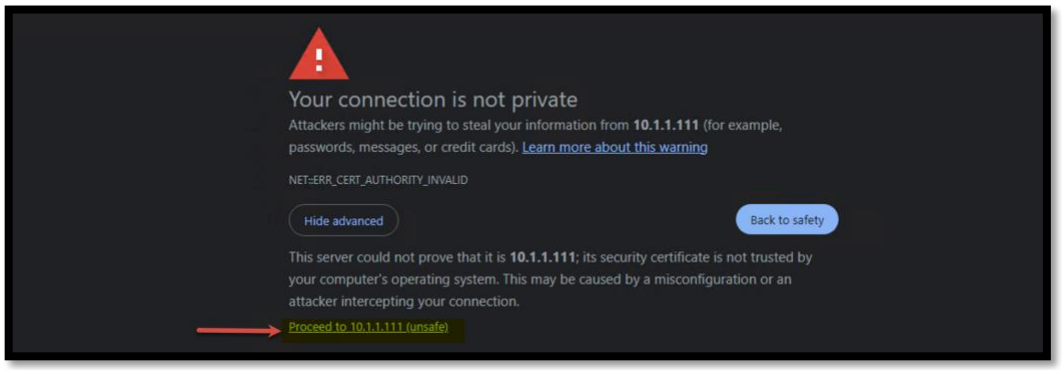


- In the previous lab, we blocked access to public Email server, however, we were not getting a block message because of the inability to redirect HTTPS inspection traffic. This issue should be resolved with HTTPS Inspection enabled.

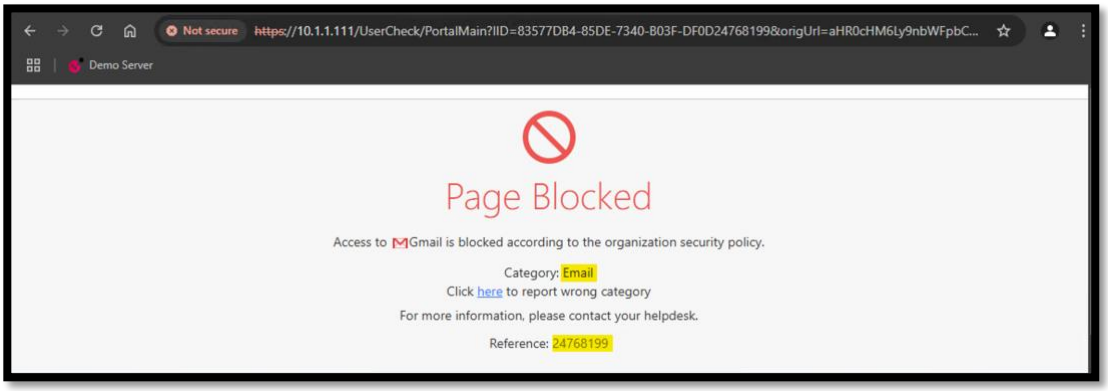
17. From the **win_client** host, try to reach **Gmail**.com or any public Email server. Note that the UserCheck blade is using the default GW VPN Certificate. This is a self-signed internal cert that is not trusted by default. This is separate from the HTTPS Inspection, and it is presented to the user because we are accessing the **UserCheck** portal.



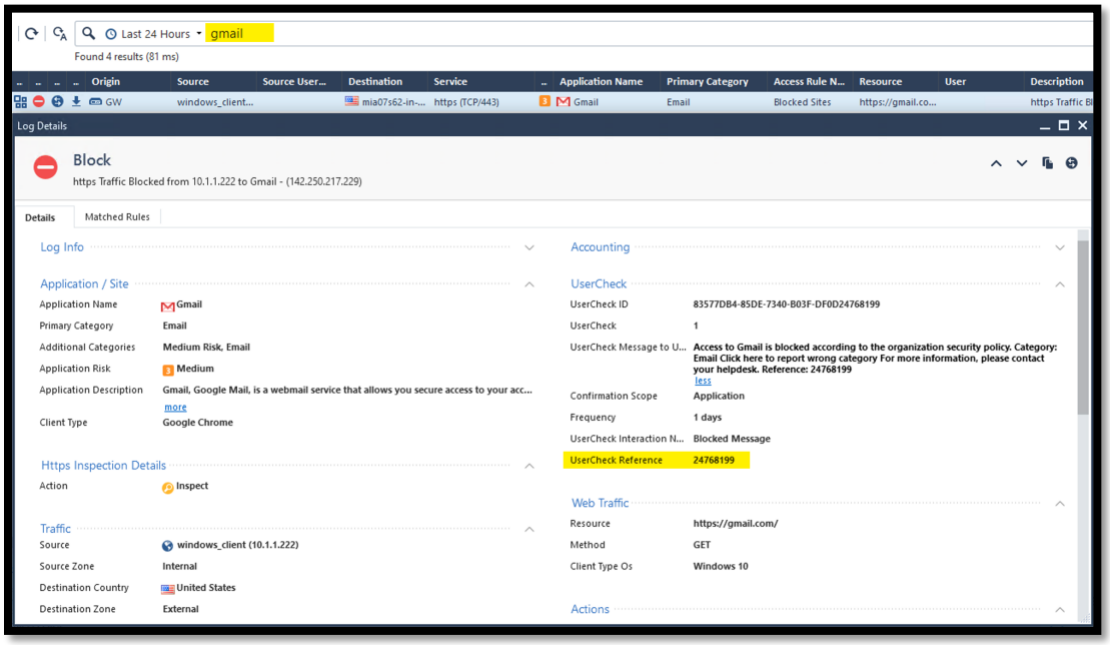
18. Bypass the security warning related to the UserCheck certificate.



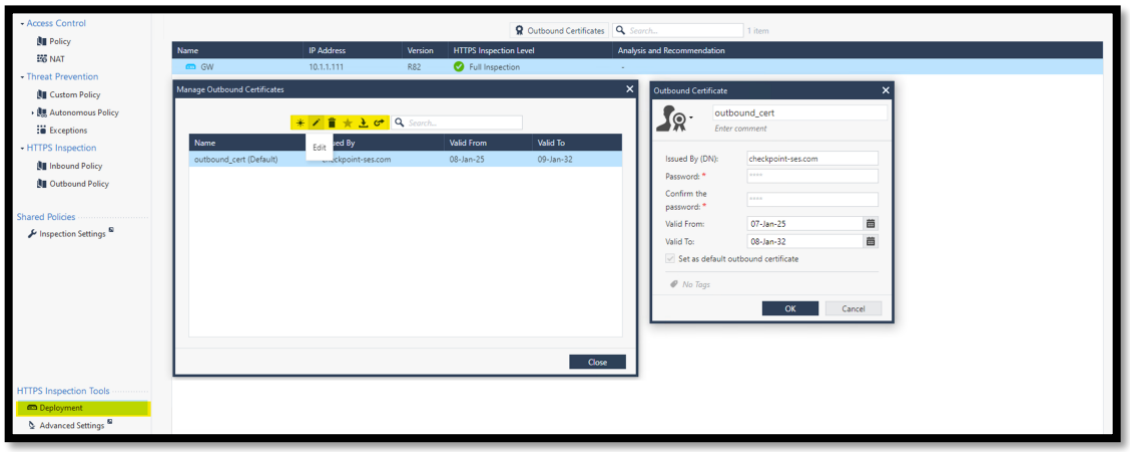
19. notice that block message is now returned successfully.



20. Review the related log, you can filter logs related to Gmail or you can use the UserCheck Reference.



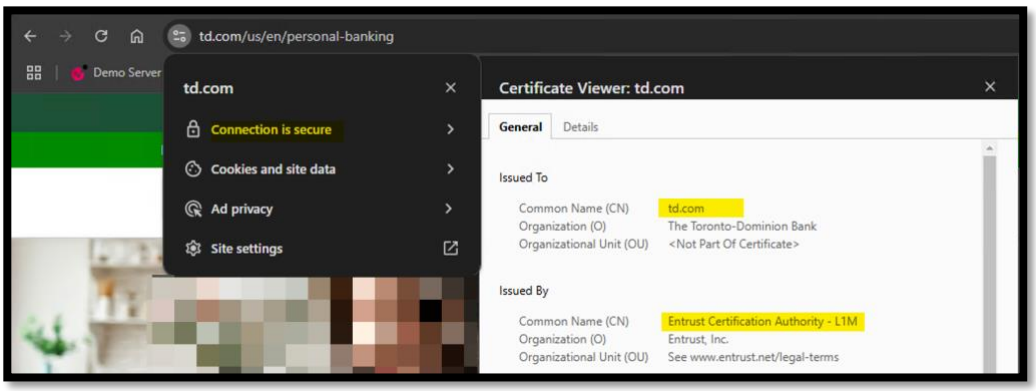
- Note that starting R82, it is possible to **add**, **edit**, **delete** and perform other operations related to the outbound certificates under the **Deployment** settings.



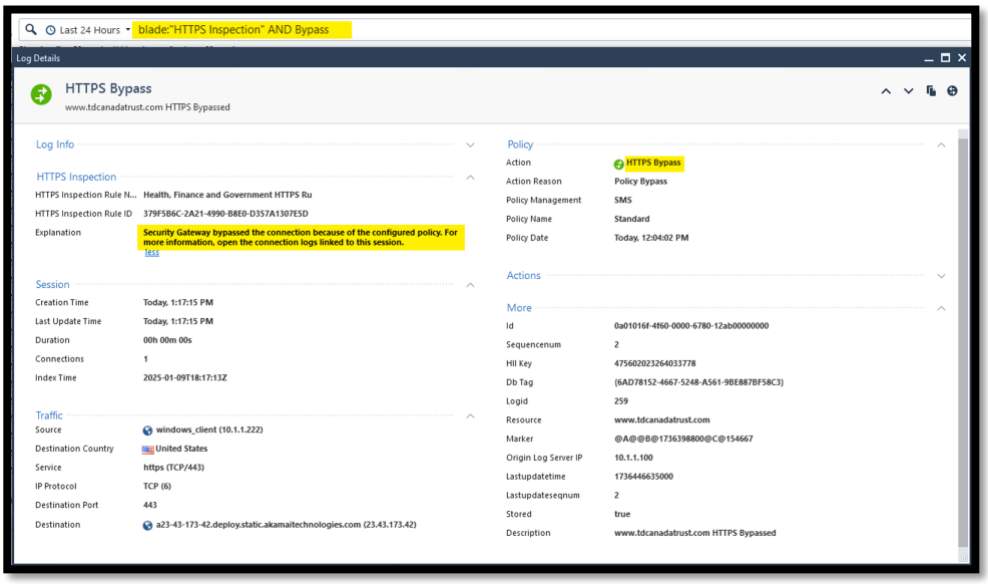
Exercise 2: Bypass Behaviors

There are cases when some websites are not inspected by HTTPS inspection. For example, some servers will not allow the GW to reach the server on behalf of the client (certificate pinning to prevent **MITM** attacks), or servers that do not follow the protocol standards.
There are other scenarios where we bypass due to **regulation** such as in financial and health web sites.

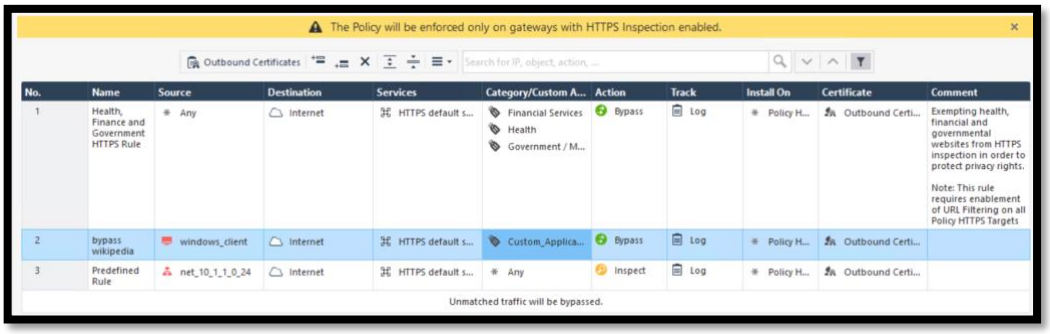
1. From **win_client**, use chrome to browse to a financial institution website. For example, https://www.td.com. Review the issuer of the certificate.



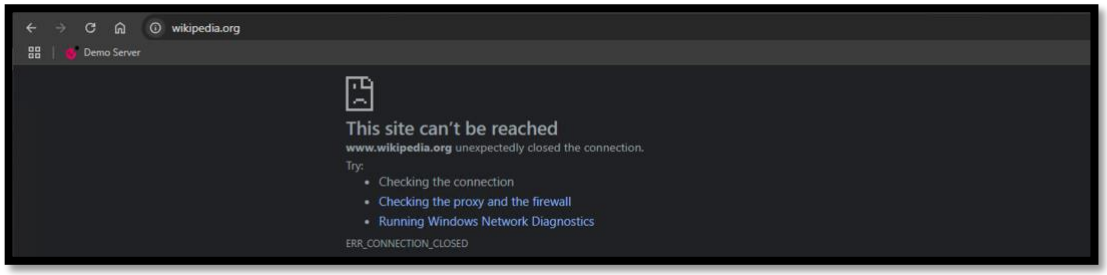
2. Review the HTTPS Inspection bypass log. Notice that this was bypassed as configured in the first rule of the outbound policy.



3. In the previous lab, we created the override categorization object to recategorize https://www.wikipedia.org. Create a new rule below the existing bypass rule, use the **Custom Application** object to bypass inspecting traffic to Wikipedia.org from the win_client host.

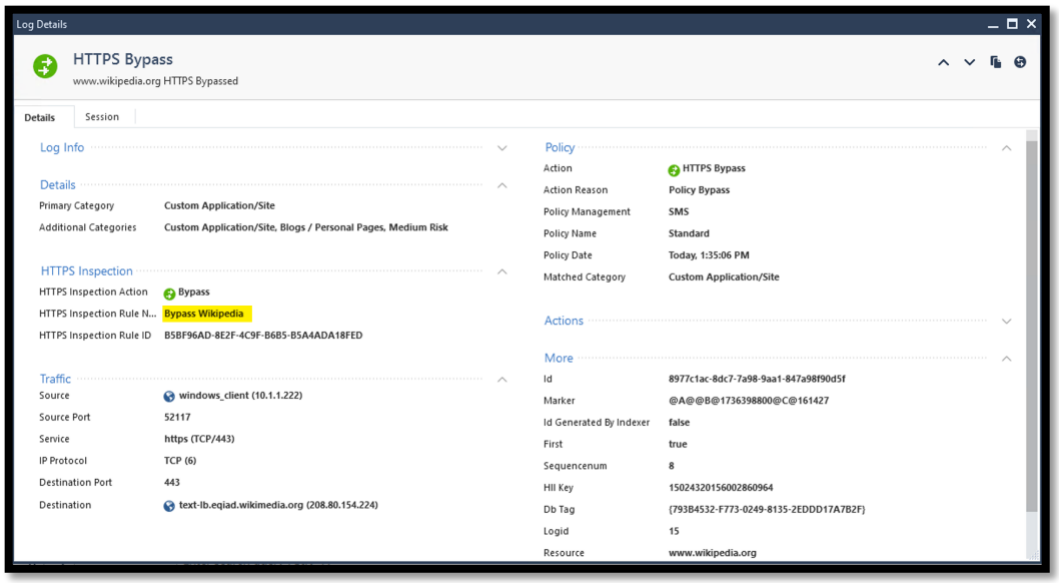


4. From **win_client** try to reach https://www.wikipedia.org.

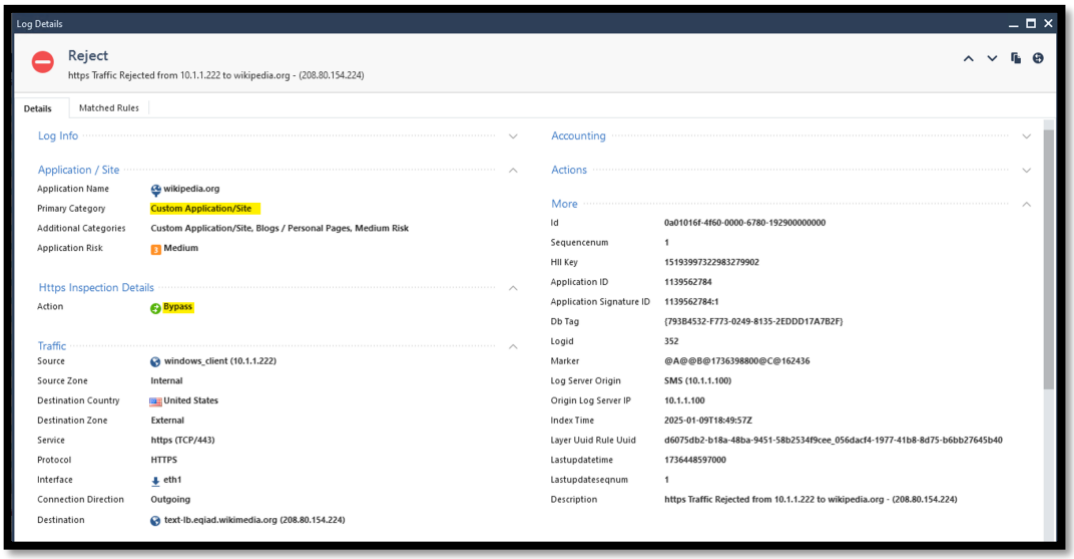


- Notice that once the site is bypassed, the HTTPS inspection blade will not handle the traffic. Hence, the block message will not be presented to the user since redirecting HTTPS sites requires the HTTPS inspection blade to be active for the connection as demonstrated in the previous steps.

5. Review the HTTPS inspection log and notice the details related to the user related settings.

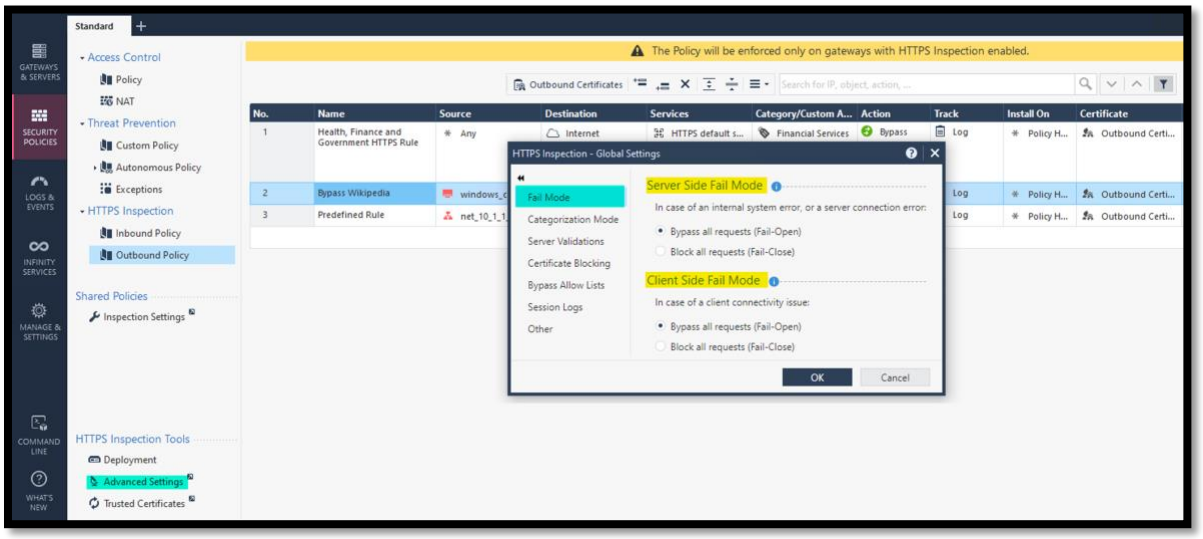


6. Review the URLF log and review the related HTTPS inspection field indicating a bypass.

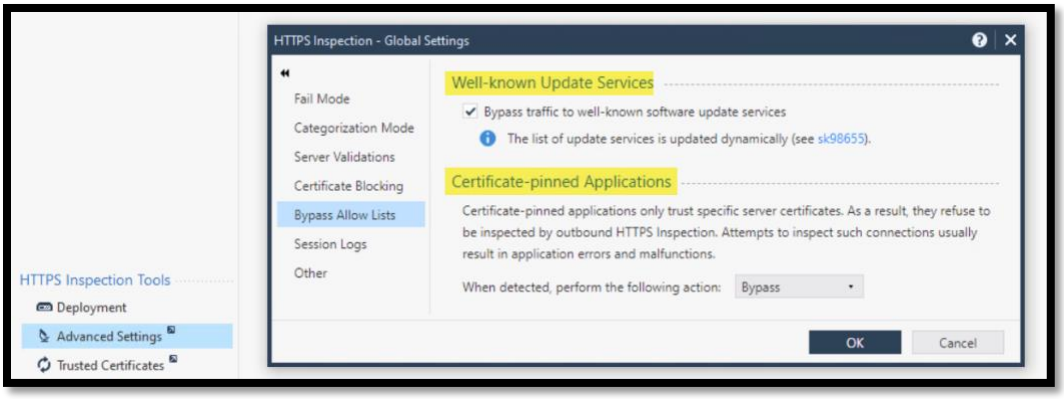


7. From **SmartConsole**, Open **Advanced Settings** and review the default Fail Modes.

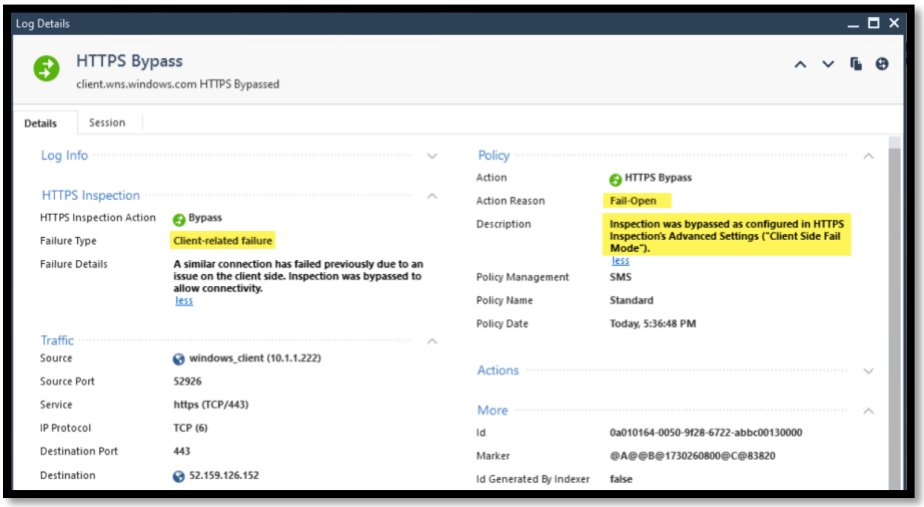
   - The **Client Side Fail Mode** controls issues on the client side
   - The **Server Side Fail Mode** controls:
     - Engine failures
     - Server issues.



8. Review the Bypass Allow Lists. Review the list in SK98655.



- Note that the log below shows a bypass action related to the fail-mode.

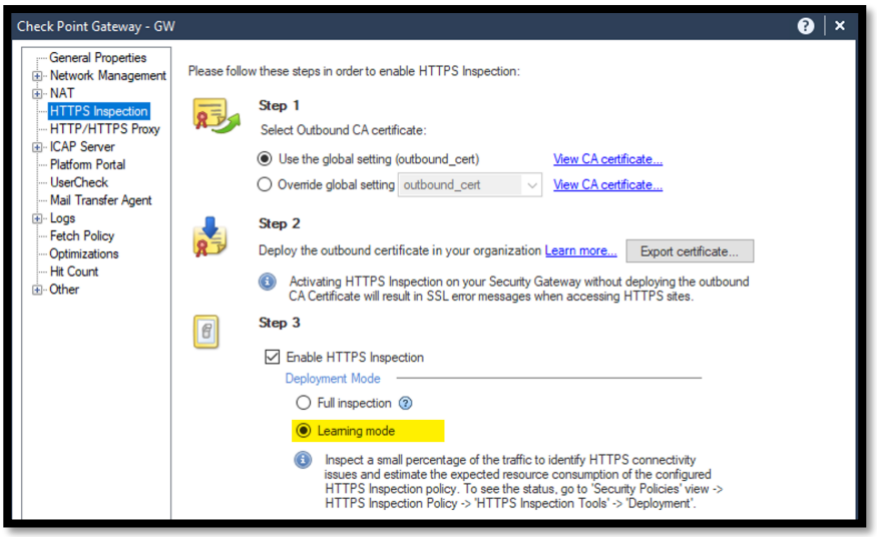


Exercise 3: Deployment Assessment

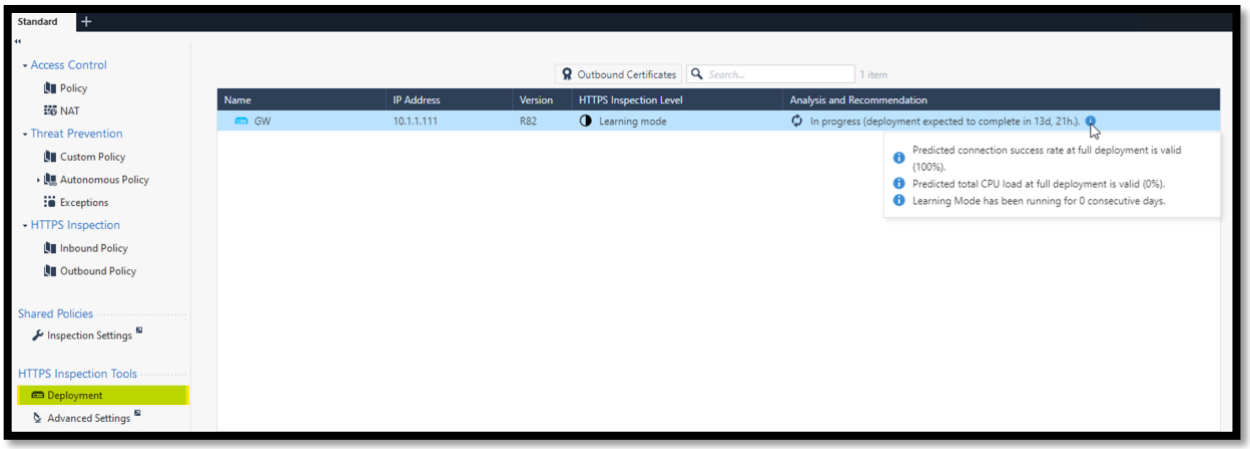
In the previous exercises, we configured the HTTPS Inspection outbound policy to inspect traffic from one network **10.1.1.0/24** while all HTTPS traffic from other network will be bypassed by default. This can be a good deployment strategy to deploy HTTPS inspection gradually.

R82 brings a new deployment assessment feature to monitor the traffic and provide recommendations.

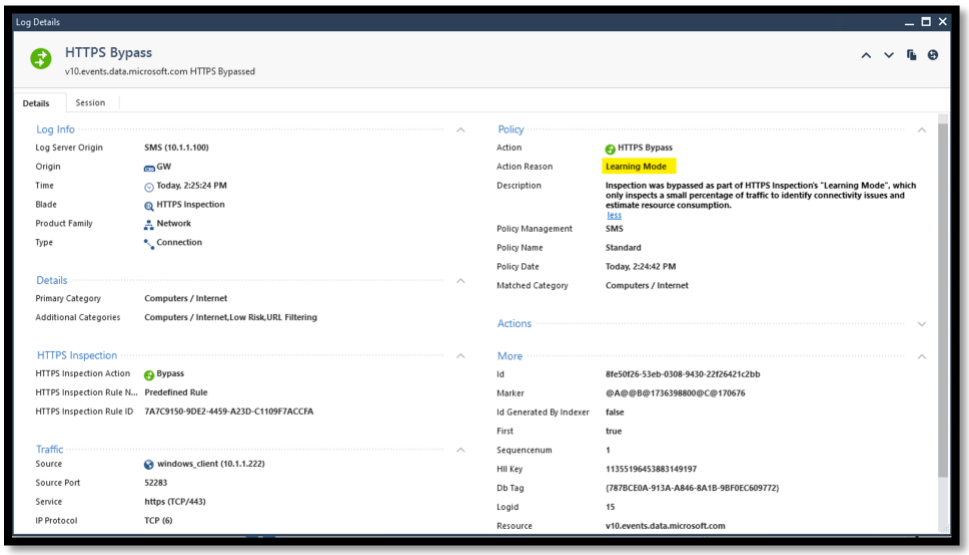
1. Edit the GW object and change the HTTPS Inspection **Deployment mode** to **Learning Mode**.



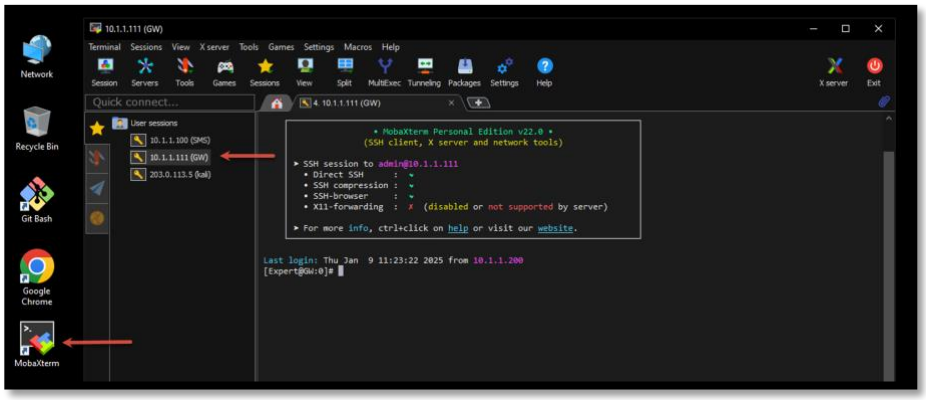
2. Install the Access Policy.
3. Review the **GW** status under Deployment.



4. Review the HTTPS inspection logs and notice that the traffic is bypassed.

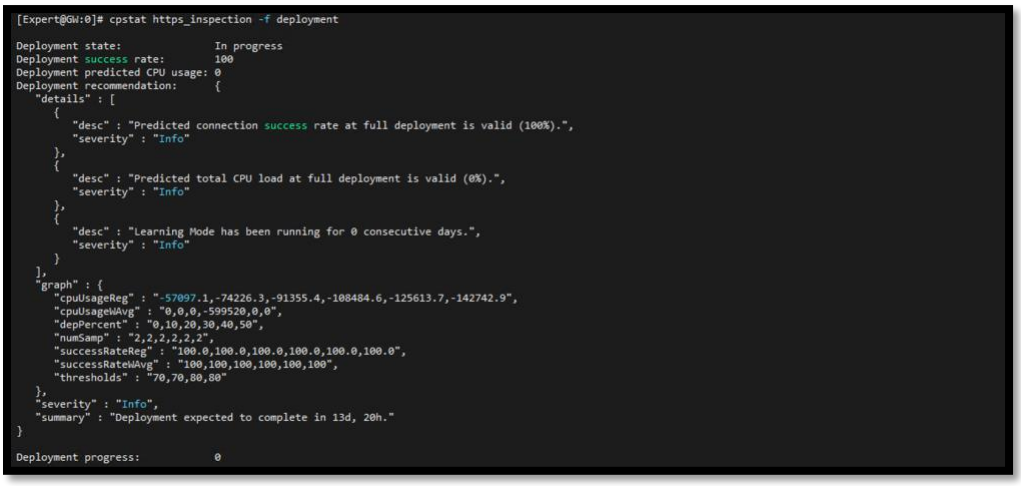


5. To get more details, use the SSH client **MobaXterm**, and open the **GW** saved session.



6. Run the command below to see get the status of the deployment assessment.

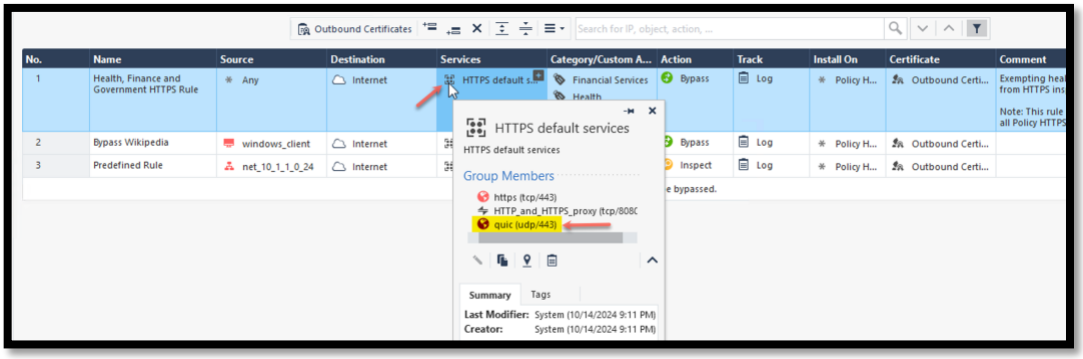
```
cpstat https_inspection -f deployment
```



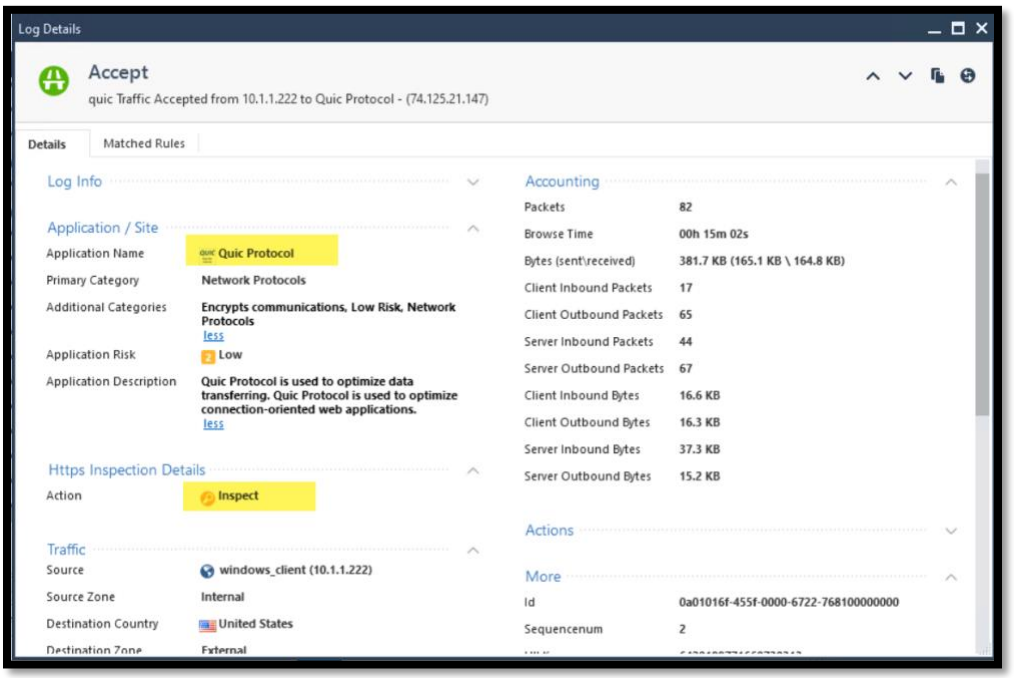
Exercise 4: HTTP3 Protocol over QUIC

Starting **R82**, the Check Point **GW** is capable of inspecting **HTTP3/QUIC** traffic. In this exercise, we will review the logs and changes related to this feature.

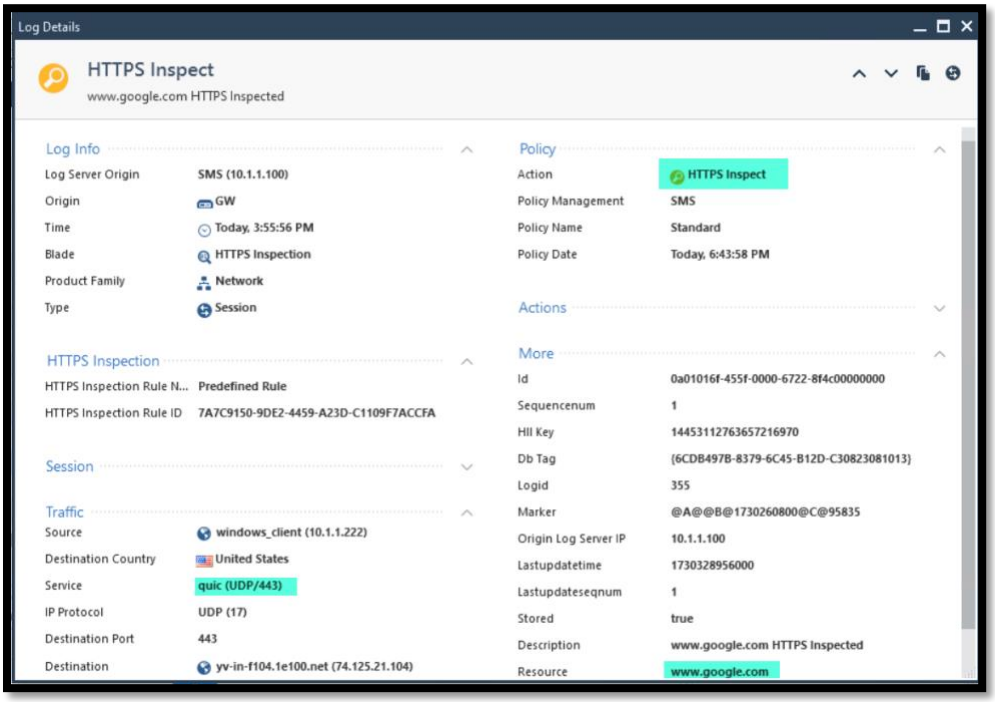
1. Open the service group used in the HTTPS Inspection Policy **HTTPS Default services**. Notice that **QUIC** is now inspected by default.



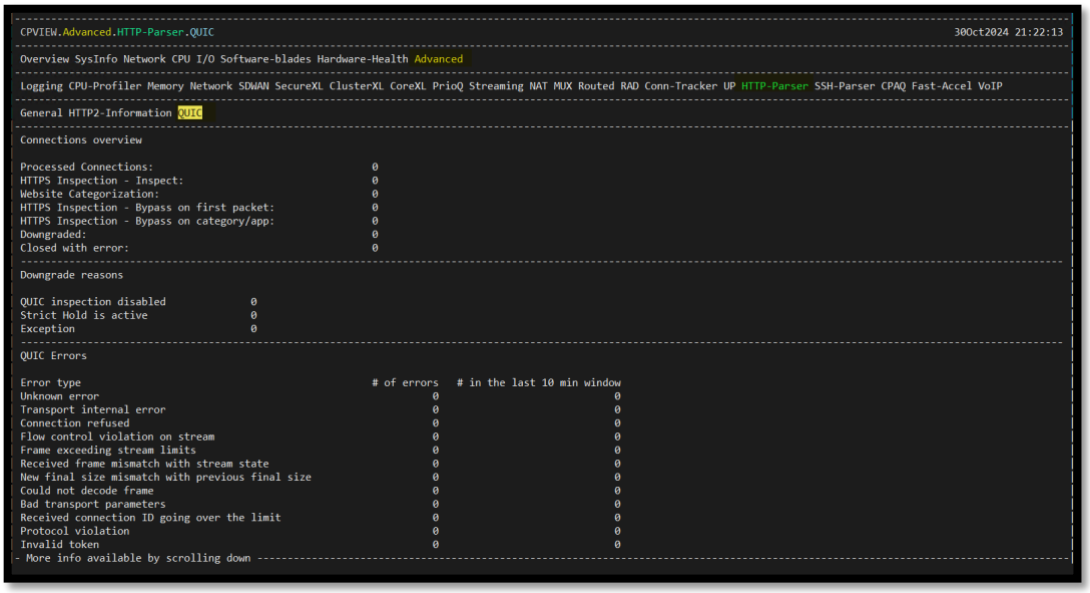
2. Filter the logs to find **QUIC** related logs. Notice that we can see logs from the Application control inspecting **QUIC**.



3. Review the HTTPS inspection logs, notice that the **QUIC** traffic is being inspected like how HTTPS traffic is logged.



4. Login to the GW over SSH and use CPVIEW to see more details regarding **QUIC** protocol inspection.



5. Edit the GW object and change the deployment mode for HTTTPS Inspection to **Full Inspection**.
6. Install the Access Policy.

End of Lab 2

