Software Developer Solution Guide

TLS and Certificate Integration

VERSION 2.1 | JANUARY 2026

WHITEPAPER

Table of Contents

Table of Contents

The following acronyms and their associated definitions will be used throughout this document.

Acronym	Definition
AD	Active Directory
AD FS	Active Directory Federation Services
CA	Certificate Authority
CLI	Command Line Interface
DNS	Domain Name Service
FQDN	Fully Qualified Domain Name
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaC	Infrastructure as Code
IdP	Identity Provider
IP	Internet Protocol
OS	Operating System
PKI	Public Key Infrastructure
SAML	Security Assertion Markup Language
SCIM	System for Cross-Domain Identity Management
SP	Service Provider
SSH	Secure Shell
SSL	Secure Socket Layer (superseded by TLS)
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VDI	Virtual Desktop Infrastructure
VM	Virtual Machine
WLAN	Wireless Local Area Network
ZCA	Zscaler Central Authority
ZDX	Zscaler Digital Experience
ZIA	Zscaler Internet Access
ZPA	Zscaler Private Access
ZTE	Zero Trust Exchange

Introduction

This Solution Guide provides guidance that will help implement Zscaler Internet Access (ZIA) with TLS interception, thereby reducing friction and lowering security risk(s) and degradation to the Developer's user experience and workflow. This solution guide is intended for use by software development teams, software developer managers, and Zscaler security professionals that support the enterprise environment.

The document focuses on the challenges faced in a developer's environment when using TLS interception leveraging Zscaler Internet Access. It is paramount that the ZIA Administrators and a small group of Developers within the enterprise work together to bridge the gap between the flexibility a Developer needs and the security that ZIA can enforce.

This Solution Guide will discuss the following topics:

- How Zscaler Internet Access provides security to Software developer use cases.
- Zscaler TLS Interception and Public Key Infrastructure (PKI) Certificates.
- TLS interception and configurations to the developer's tools.
- Cloud Applications and configurations to the developer's tools.
- Zscaler Internet Access configurations and TLS Inspection troubleshooting guidance.

Initial Configuration guidance when deploying Zscaler Internet Access (ZIA) TLS Interception can be found in this document: ZIA SSL Inspection Leading Practices Guide

Subsequent editions of this solution guide will include common ZIA configurations and guidance for Zscaler Private Access (ZPA) to secure developer traffic from endpoint devices to server based targets.

1.1 Problem Statement for Software Developers

When securing a developer environment with Zscaler TSL/SSL interception, the biggest blocker to a developer's workflow are applications that do not use the System Certificate Store to verify TLS certificates. Hence, this Solution Guide focuses on two common problem cases:

1. Applications or software packages that do not use the default system certificate store or do not trust Zscaler's certificates. This causes unexpected errors or behaviors that otherwise would not appear.

2. Applications that perform certificate chain checks (Certificate Pinning or mTLS) not allowing Zscaler TLS interception to inspect the traffic.

There are 3 places where the problems can be addressed:

1. Zscaler SSL/TLS Interception Policies and Cloud Applications.
2. Zscaler Client Connector deployment and Configuration.
3. Developer Computer's Settings.

This guide proposes a method to minimize the amount of time spent troubleshooting applications with SSL/TLS issues caused by Zscaler TLS/SSL interception.

1.2 Solution Architecture

This solution presents a 4 step approach to minimize developer's issues with Zscaler. There will be Zscaler Internet Access configurations, Zscaler Client Connector configurations, and Mac OS computer settings that the developer will need to perform to enable full TLS interception and avoid issues.

The following architecture design principles must be maintained for the Developer cohort to be productive when using Zscaler. It is assumed the Developer's local computer will have the Zscaler Client Connector installed and that the developer has Administrator rights on his own machine. These additional assumptions are also applicable:

- Developers are aggregated into a single identity group.

- Internet access is required for the Developer's local computer.

- Internet access is required for the server/application that builds or hosts the application created by the Developer.

- The Developer is required to make connections to the servers to transfer data, upload code and review, test, administer, and troubleshoot the server.

- The Developer is expected to inspect and troubleshoot the inter-server communications.

Figure 1 shows how a typical developer will use multiple tools on his machine that follow different paths to reach the internet. This use case is core to the issues that developers face.

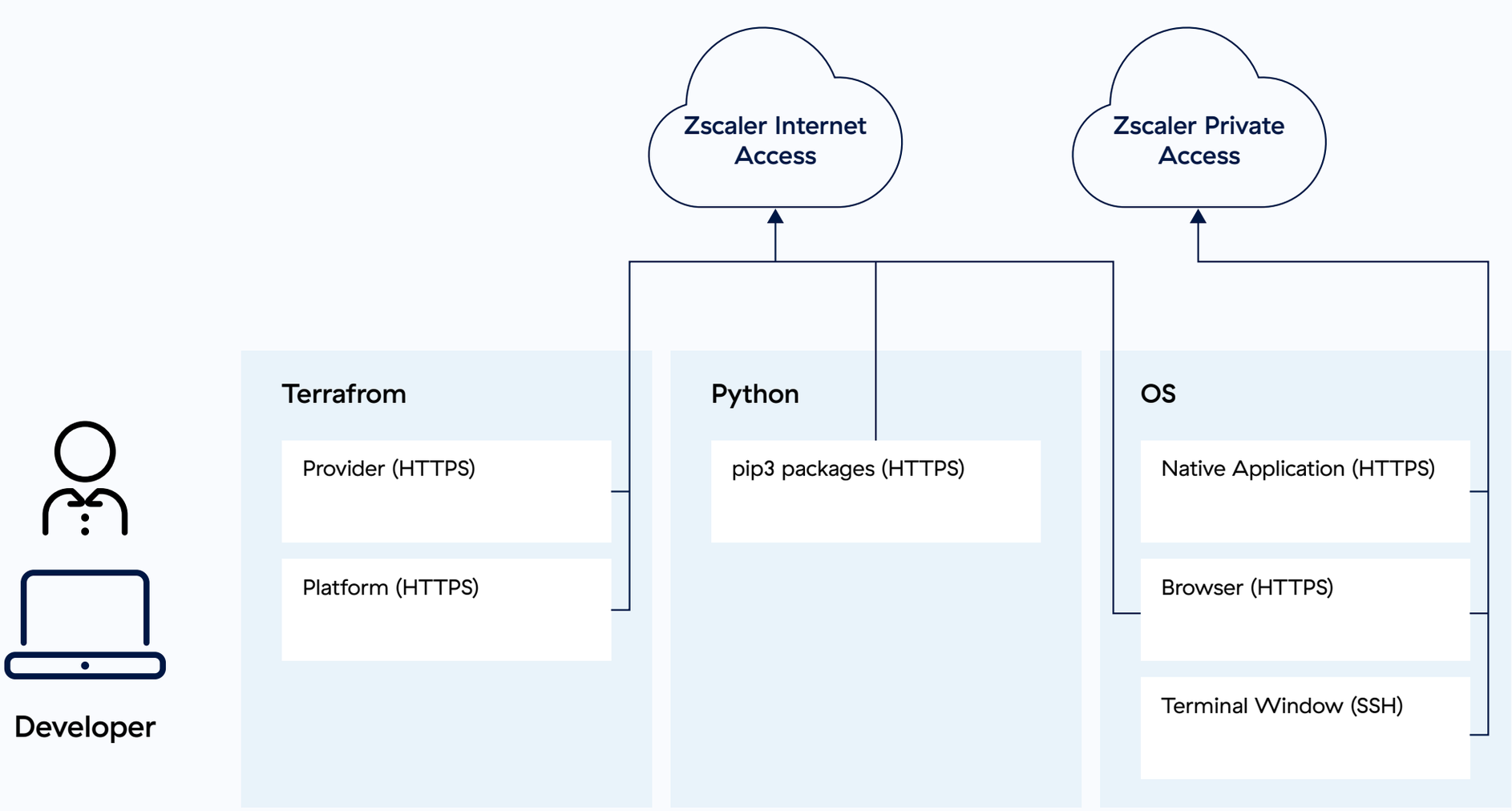


Figure 1: High level Developer Workflow

The 4 steps proposed (explained further in the Solution section) in this document are the following:

- Pre-loaded Dev Tool Policies.
- Dev Tools Visibility.
- Policy Tuning.
- Protection for Developer Tools.

1.3 TLS Inspection Architecture

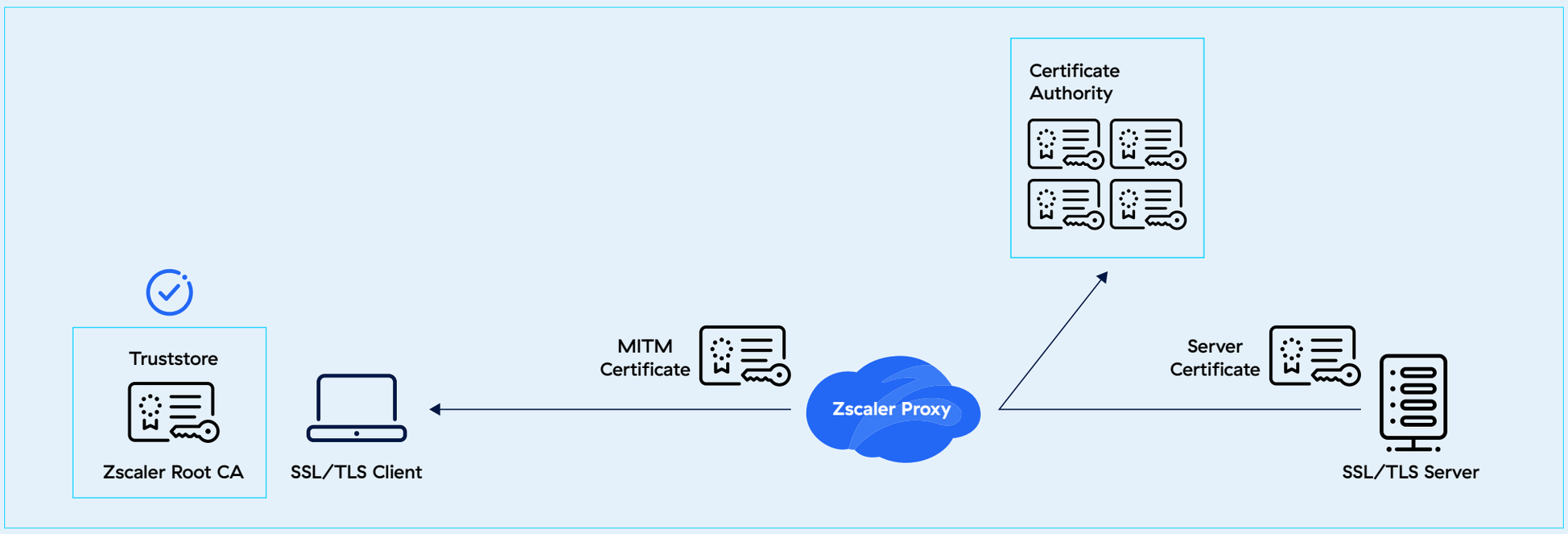
This section provides an overview of how TLS inspection is done, but details about the TLS protocol are outside the scope of this document. When establishing a secure TLS session from a client device to a server, both entities will exchange secure messages, called a handshake. During this exchange, the server sends a certificate to the client endpoint device to prove its identity. The client endpoint verifies this certificate against a pre-installed list of certificate authorities, included within the web browser or operating system's trust store. The client endpoint also confirms that the current date is a value between the "issued" and "expired" dates of the certificate. Before the network layer determines if the TLS handshake was successful, several other steps usually are done.

Certificate validity is checked with either the Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP).[1] In both cases, the client can contact the certificate authority (or its delegate) and verify if a given certificate has been revoked by the certificate authority. The result is success or failure for a given certificate. If a certificate fails the verification, the TLS handshake is completed but no further data is sent and a warning message is shown in the Web browser or other client application.

If an application is using "certificate pinning", it will perform an additional check to validate that a specific hardcoded certificate or its public key is present in the TLS connection. This ensures that the app only accepts that pre-approved certificate, preventing Zscaler Internet Access from working for this application.

Zscaler intercepts all requests made by the client and uses this to make SSL interception decisions (refer to **Figure 2** for a diagram of this process). When a session is selected for interception, ZIA operates as a MiTM (Man in The Middle) proxy which intercepts the encrypted traffic, decrypts the traffic, inspects and applies controls, then re-encrypts the session before sending it to its destination. For ZIA to decrypt encrypted traffic, the Zscaler service dynamically generates, signs and issues a certificate on behalf of the server and thus requires the client endpoint to trust this certificate issued for Zscaler root Certificate Authority (CA). This is achieved by installing a Zscaler Root CA certificate as a trusted root CA in the Developer's local machine certificate trust store. Operating systems use a system-wide trust store to keep the root CA certificates; Linux is managed by OpenSSL, MacOS utilizes Keychain, and the systems trust store on Windows machines are managed by CertMgr. In the Software Configuration Reference and Troubleshooting Connection Problems sections, there are some examples for Windows and Mac on tools that are commonly used by developers.

[1] More information on PKI environments can be found here.



Solution

For this solution there are three Zscaler aspects that need to be considered:

- Zscaler Internet Access (ZIA).
- Zscaler Client Connector.
- Developer client machine.

When configuring ZIA policies please use the order of policy enforcement documentation to understand how ZIA evaluates a given request. If it has been determined that there is a problem at the TLS layer, then proceed to determine if there is a missing certificate trust configuration or a TLS network error. For more information on finding errors, refer to the ZIA reports explained in the Troubleshooting Connection Problems section of this document.

Communication between ZIA Administrators and Developers reporting issues is the most important aspect for swift resolutions. To address this, we propose a feedback loop between the developer and the ZIA administrator in the form of a report and a confirmation. The report leads to a configuration change or to more analysis. This feedback loop starts with the developer reporting that a specific domain is presenting TLS errors. The loop is either finished when the ZIA administrator gets confirmation that the application is working or getting involved if further investigation is needed. We provide examples of how this communication can be done using the configurations and tools provided within ZScaler.

We want to guide the Developer experience so that they will know how to fix their own application SSL errors, and detect when a bypass is required due to Certificate Pinning. It will come down to these 3 common solutions for developers when they encounter an application error:

- **Rare occurrence:** Specific ZIA SSL Inspection Policy to bypass the domain when Certificate Pining is encountered.

- **Common resolution:** Install SSL Interception intermediate certificates into Developer application/OS internal certificate store.

- **Uncommon resolution:** Install or configure the application's specific packages to make use of system trust store or certificate bundle.

When the developer is not able to troubleshoot the root cause of the SSL error, the most common scenarios are:

- A missing configuration for the application to Trust the Zscaler Root Certificate or a custom certificate bundle. This is addressed by a computer configuration or software update. In very rare occasions, an application has no configuration to trust custom certificates.

- Public Key Pinning (PKP) or Certificate Pinning, where an application expects a specific server certificate in order to make the connection. When this happens and ZIA is enabled in the environment, the application will not work and a URL bypass is needed.

All configurations presented in this document can be found in the Terraform format in section 5.5 Terraform Code Examples.

2.1 Gain Visibility on "Developer Tools"

Both Developers and ZIA Administrators are encouraged to leverage pre-defined URL Categories that have been built by the security teams within Zscaler. In this particular case, Zscaler provides the "Developer Tools" and "System Tools" as a list of sites that are commonly used by developers and their workflow.

In this context, we also provide two search tools that allow Developers and ZIA Administrators to match a given URL to the list of URL Categories where they can be found. The final configuration under TLS/SSL interception must be common knowledge by both Developers and ZIA Administrators.

2.1.1 Site Review

Developers can use "Site Review" (https://sitereview.zscaler.com). This will allow them to verify in which pre-defined URL Category a given site can be found. They will be able to "Look Up" each site and see which URL Categories it is included in. **Figure 3** shows an example for apple.com.

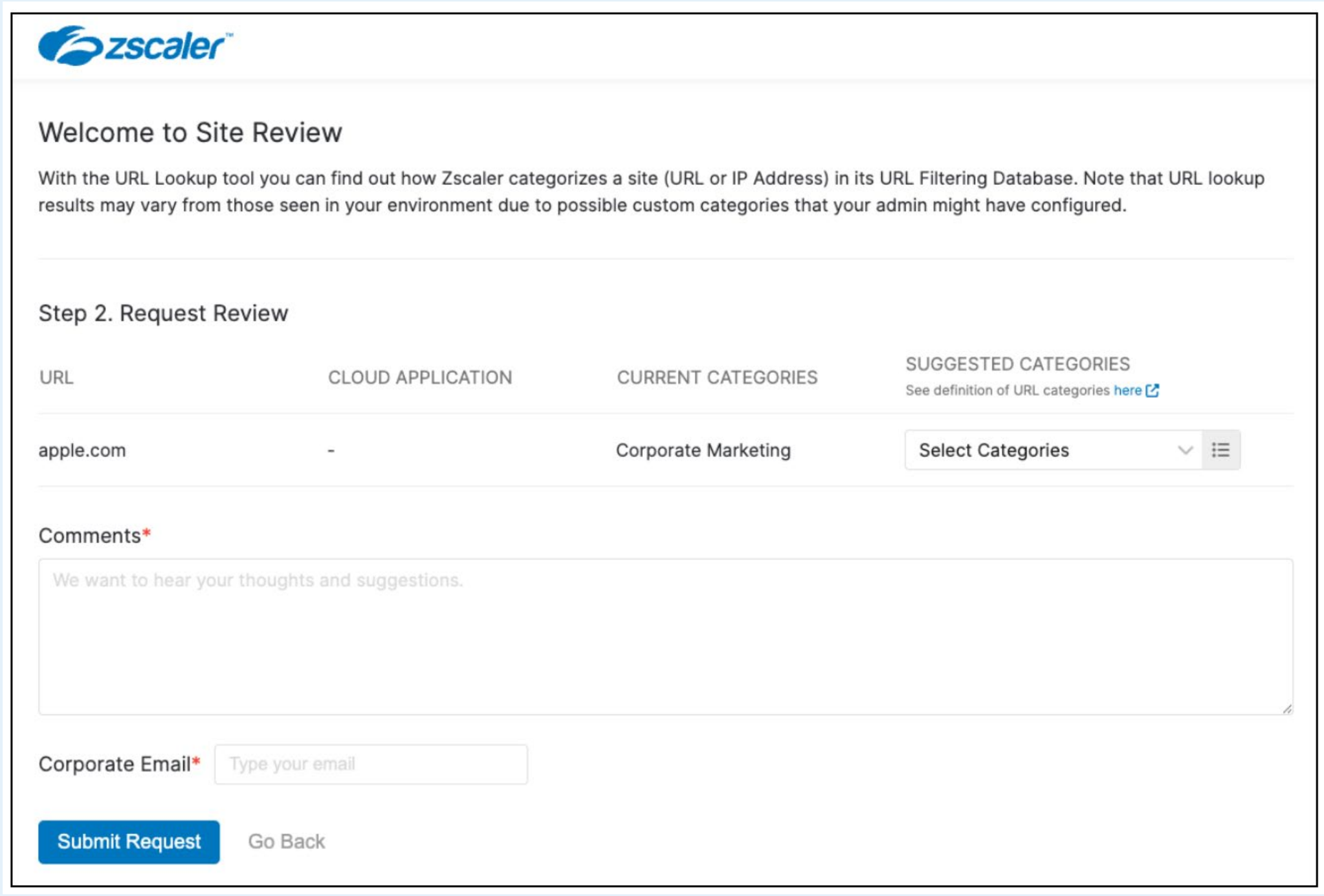


Figure 3: Site Review output for apple.com

Many sites that Developers use should be included in either “Developer Tools” or “System Tools” predefined URL Categories. If a site needed for the Developer’s workflow is not in those categories, then it must be considered in which category it is found or if a custom category is needed.

The following table shows examples (as of October 2025) of some URLs and to which category they belong. These URLs are also used in the next section of this document.

URL	URL Category
app-site-association.cdn-apple.com	CDN
itunes.apple.com	Music and Audio Streaming, Online Shopping
developer.apple.com	Other Information Technology
apple.com	Corporate Marketing
setup.icloud.com	FileHost
codeload.github.com	Professional Services, Other Information Technology, Developer Tools
github.com	Professional Services, Other Information Technology, Developer Tools
httpbin.org	Other Information Technology
api2.cursor.sh	Generative AI and ML Applications

The Developer group will gain the knowledge of which URL Categories contain the URLs that they will be using.

2.1.2 URL Search

For the ZIA Administrator within the administration portal, in the help section you will find the URL Search functionality. This URL Lookup tool effectively serves the same purpose as the Site Review. See an example in **Figure 4**, also using apple.com.

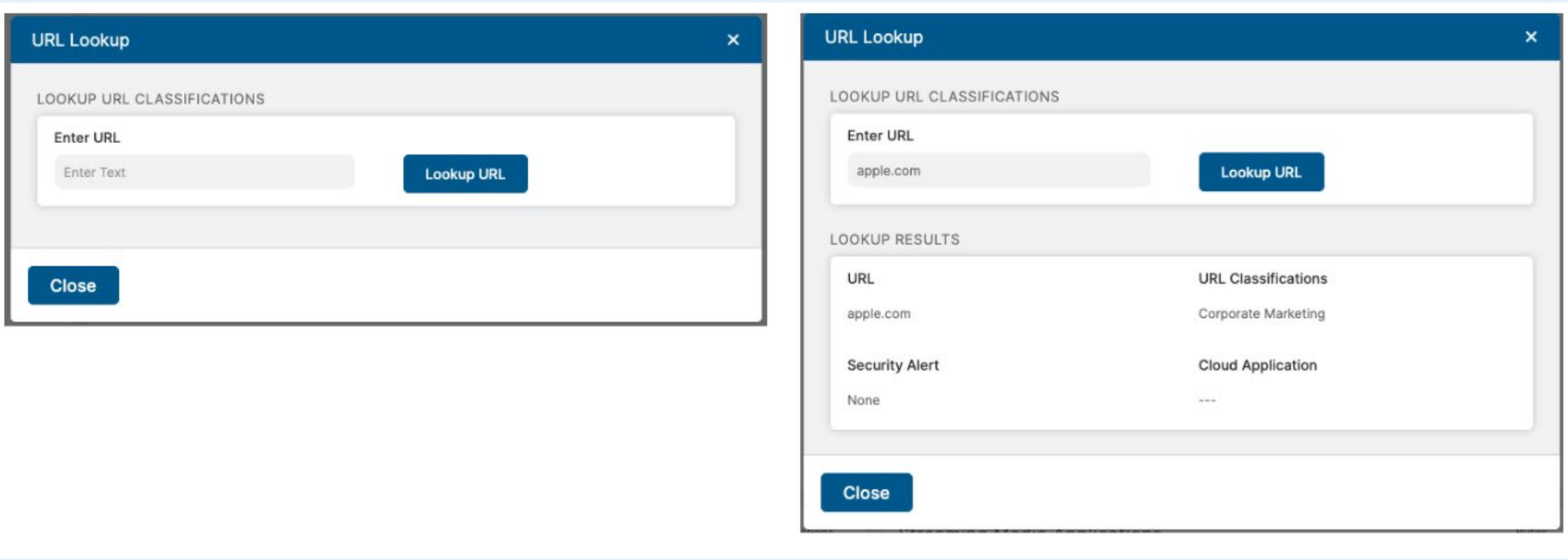


Figure 4: URL Lookup output for apple.com

2.2 Phased Approach to Securing Developer Environments

The following 4 steps outline both how to gain visibility into what URLs Developers need and how to make the necessary changes for Developers to be able to work securely.

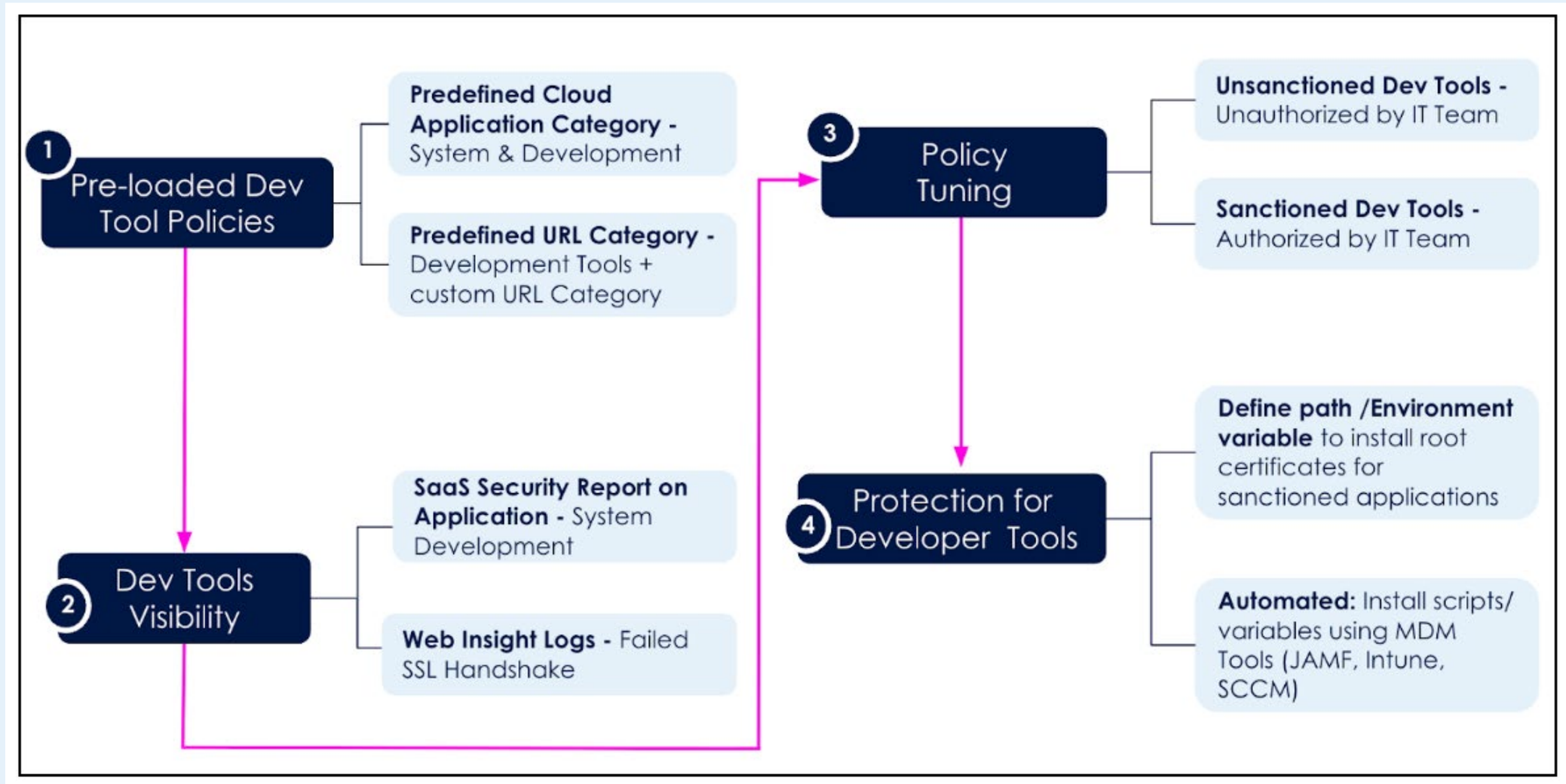


Figure 5: 4 Phases to Securing Developer Environments

2.2.1 Step 1 – ZIA SSL Interception Configuration

This section explores how to configure TLS inspection and Cloud Applications for developers. This first step will include:

- Configure TLS Interception policies and Cloud Applications to gain visibility
- Assess the URLs that will affect Developers
- Make configuration changes to start to intercept those URLs.

We do this to prevent any blockers to the Developer's workflow, gain insight into what sites Developers will need, and possibly detect any future blockers. When applying TLS inspection policies, keep in mind the following principles:

- A single policy can be applied to up to 4 individual users. To create scale, utilizing groups allows an administrator to apply policies to many users.

- Understanding the needs of each developer group allows policies to be applied to more than one developer team in a single group.

- Each group will have at least one URL category for both SSL Interception and URL Filtering.

- Consider using a predefined URL category for all groups and a unique custom URL category for each group.

- Lastly, keep in mind that a TLS Interception bypass for domain github.com would bypass all repositories. But if used as a URL filter (i.e. github.com/github-copilot), it can be used to block as long as the github domain is being inspected by TLS Interception.

Each developer group will have a unique set of applications and URLs that they need. From step 1 to step 3, we expect that Developers will contribute additional URLs that will allow the final change to inspect all the URLs. We expect that some common categories will be shared among groups. Additionally, we are using a Custom Category (Called Bypass-PKP) where you can place any URLs on DO NOT INSPECT when there is no support for certificate bundle on a given application.

We first address URL Categories. Keep in mind that the same URL category can be used for SSL/TLS Inspection and for URL Filtering. The former uses domain names while the latter uses URLs. As stated previously, URL filtering only applies to SSL intercepted traffic or traffic with the SNI header. Therefore, when creating additional Custom Categories, please refer to URL Format Guidelines to include those URLs.

The table below provides examples of the following Custom (User-Defined) URL Category for various Developer use cases:

- Applications that have been identified to have PKP and must not be inspected (ex: Apple Store and Mac Updates).

- Applications that are in other potentially blocking URL Categories for safe handling (ex: Known Developer Sites).

- Applications that company policy wants to block (ex: Github Copilot).

- Applications that have private certificates (ex: customer specific list of sites).

Order	Name	URL Category
Bypass–Apple	amp–api–edge.apps.apple.com amp–api.apps.apple.com api.apple–cloudkit.com app–site–association.cdn–apple.com appldnld.apple.com apps.mzstatic.com bag–cdn–lb.itunes–apple.com.akadns.net bag.itunes.apple.com configuration.apple.com dit.whatsapp.net downloaddispatch.itunes.apple.com entitlements.itunes.apple.com gateway.icloud.com gdmf.apple.com gg.apple.com gs–loc.apple.com gs.apple.com gsa.apple.com h3.media.apple.map.fastly.net humb.apple.com identity.ess.apple.com ig.apple.com init.itunes.apple.com itunes.apple.com lcdn–locator.apple.com lcdn–registration.apple.com mesu.apple.com metrics.icloud.com mmg.whatsapp.net ns.itunes.apple.com oscdn.apple.com p63–acsegateway.icloud.com p63–fmip.icloud.com playgrounds–assets–cdn.apple.com	Other applications and Ap–ple–specific URLs that have Public Key Pinning (PKP) and will be bypassed.

Order	Name	URL Category
	ppq.apple.com profile.ess.apple.com securemetrics.apple.com serverstatus.apple.com setup.icloud.com skl.apple.com suconfig.apple.com swcdn.apple.com swdist.apple.com swscan.apple.com token.safebrowsing.apple updates-http.cdn-apple.com updates.cdn-apple.com xp-cdn.apple.com xp.apple.com	
Developer Safe Sites	github.com/microsoft/vcpkg.git github.com/composer codeload.github.com api.github.com/repos/Behat cursor.com api2.cursor.sh	URL requested by developers, assigned to only developers and applied only to URL filtering.
Github Auto Pilot URLs	github.com/github-copilot github.com/features/copilot/	Blocking copilot for everyone via URL Filtering.
Private CA Sites	testing.httpbin.org	Sites with invalid or private cer-tificates.

As an example, we will now use the above URL Categories and Predefined URL Categories to populate the SSL Inspection policy table. Please refer to the following table as an example of TLS Inspection policies for Developers:

Order	Name	URL Category	Criteria	Action	Description
1	SSL Bypass	Bypass-PKP	Group: Developers	DO NOT INSPECT, Evaluate Other Policies	Permanent bypass list for Developers.
2	Developers Special Cases	"Private CA Sites" or "Github Auto Pilot URLs" or "Developer Safe Sites"	Group: Developers	DO NOT INSPECT, Evaluate Other Policies	This is a rule that will contain sites that are required to be inspected but need special handling. These will eventually be changed to Inspect.
3	Developers Catch-All	URL Category: Developer Tools, Information Technology Cloud Applications: System & Development (All)	Group: Developers	DO NOT INSPECT, Evaluate Other Policies	This rule will eventually be changed to Inspect.
4	Smart Isolation One Click Rule	Smart Isolation One Click Rule		Inspect	Smart Isolate Single Click Rule automatically created upon enabling Smart Isolation from Browser Control.
5	Github_ Exemptions	Developer GitHub Requests	Group: Developers	Inspect, Evaluate Other Policies	Github Specific URL Filtering actions.
6	Private CA	Private CA Sites	Group: Developers	Inspect Allow Untrusted Server Certificate	
7	Zscaler Recommended Exemptions	Recommended SSL Exemptions		Do Not Inspect Bypass Other Policies	
8	Office 365 One Click			Do Not Inspect, Bypass Other Policies	Disabled due to rule name in logging.
	Existing rules				
Default Rule	Default SSL Inspection Rule			Do Not Inspect, Evaluate Other Policies	Default

With the above setup, all Developers should have:

- Developer SSL Bypass URLs
- The catch-all bypass/Inspect rule (depending on the step of the Journey)
- Custom list of special sites
- Two URL Filtering policies to block and allow sites based on their needs

    - » Each group of users would have two SSL Interception policies. Both are defined by URL Categories that are part of the feedback loop that is discussed in this document.

When “Developers All” policy is defined or at the Step 3 of this journey (when you switch to Inspect), there are two configurations that need to be defined for the Developer use case:

1. For policies where the Action is Do Not Inspect, block “Untrusted Server Certificates”. This configuration blocks connections to sites with certificate errors. This is controlled within the SSL Interception Action, as shown in **Figure 6.**

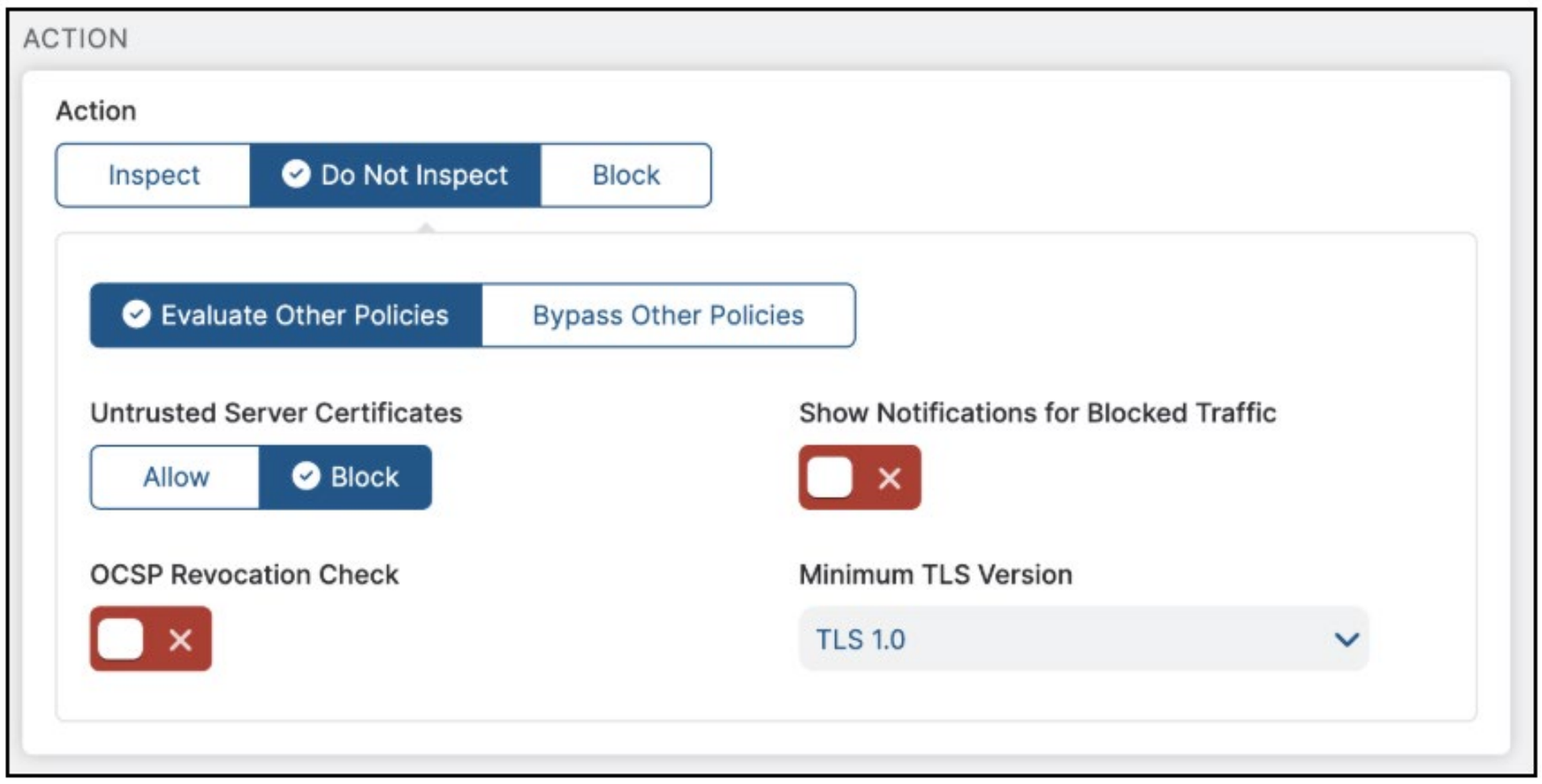


Figure 6: Block Untrusted Server Certificates when Action is Do Not Inspect

2. For policies where the action is Inspect, Allow “Untrusted Server Certifications” for developers. This will allow connections to a site that has a certificate from a private CA, invalid certificate or an expired certificate. This is controlled within the SSL Interception Action, as shown in **Figure 7.**

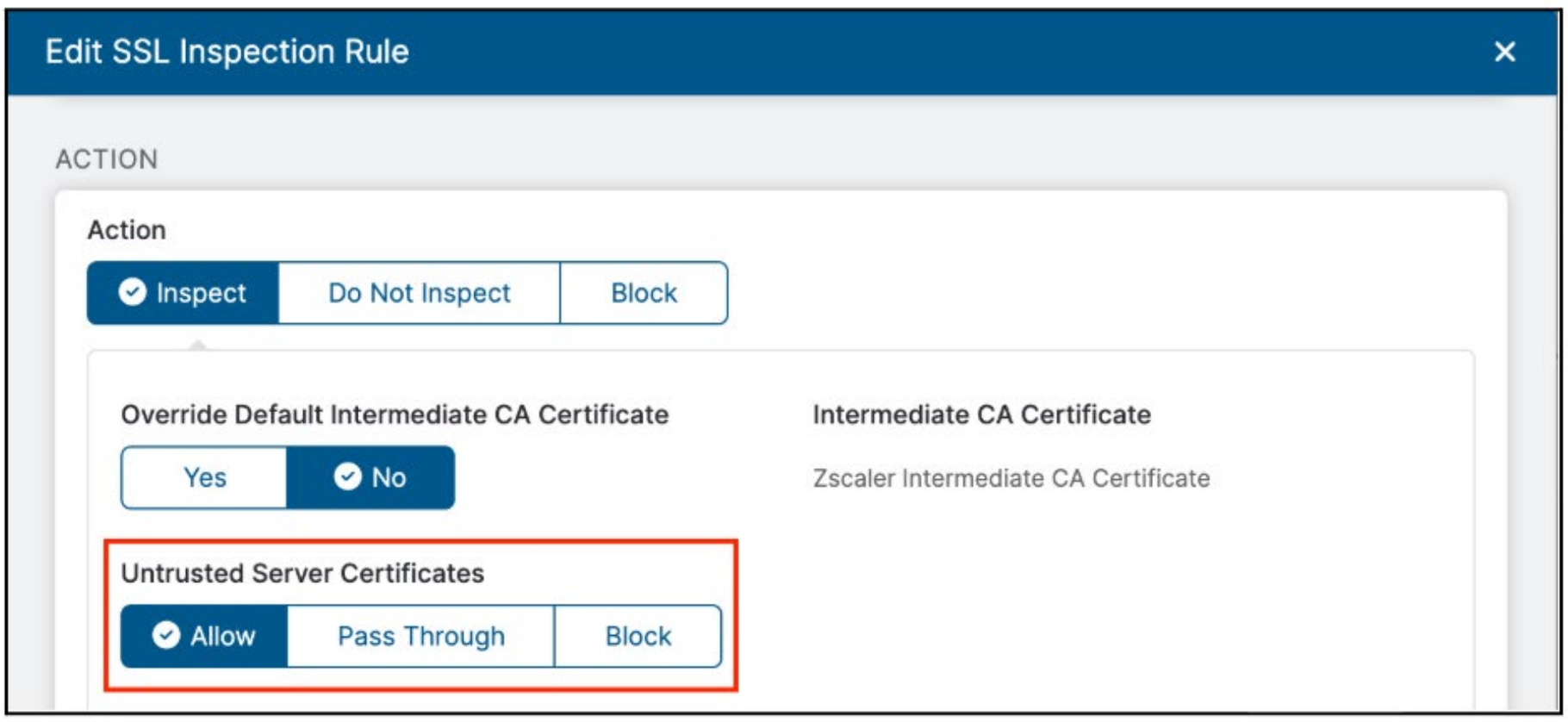


Figure 7: Block Untrusted Server Certificates when Action is Inspect

When connecting to sites protected by the Zero Trust Exchange, you can verify that it has been intercepted by examining the site’s certificate. **Figure 8** shows the Zscaler certificate.

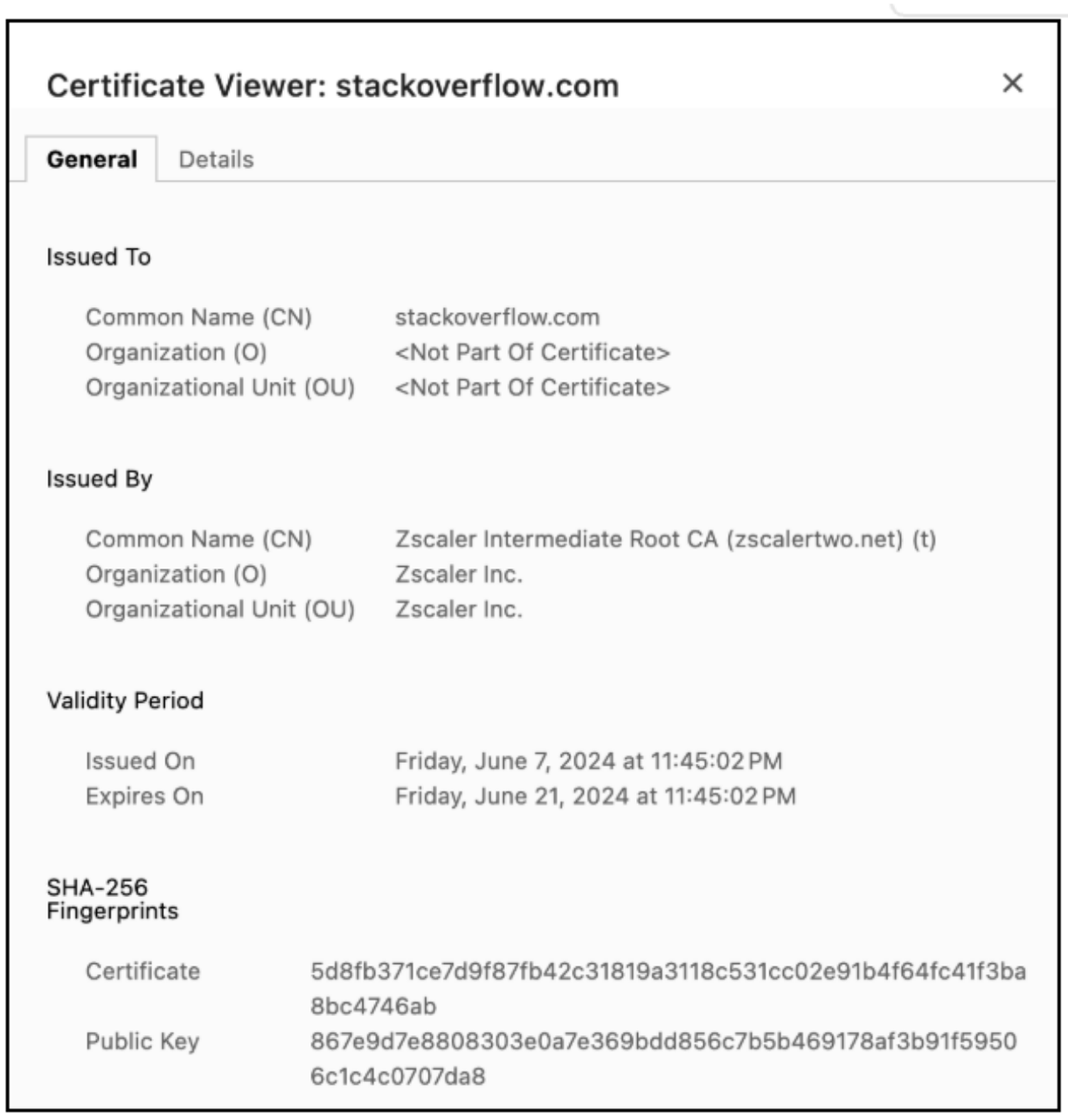


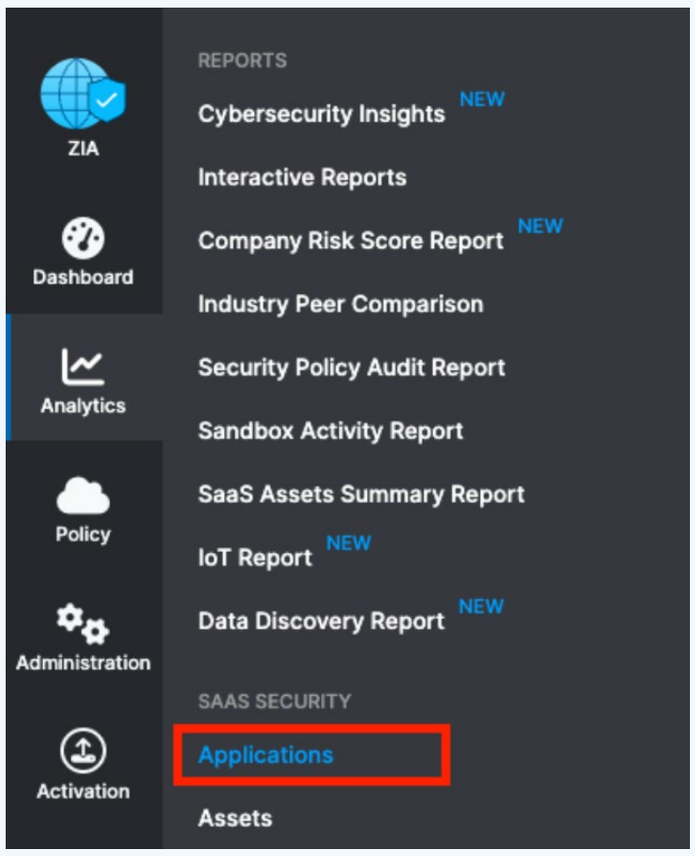
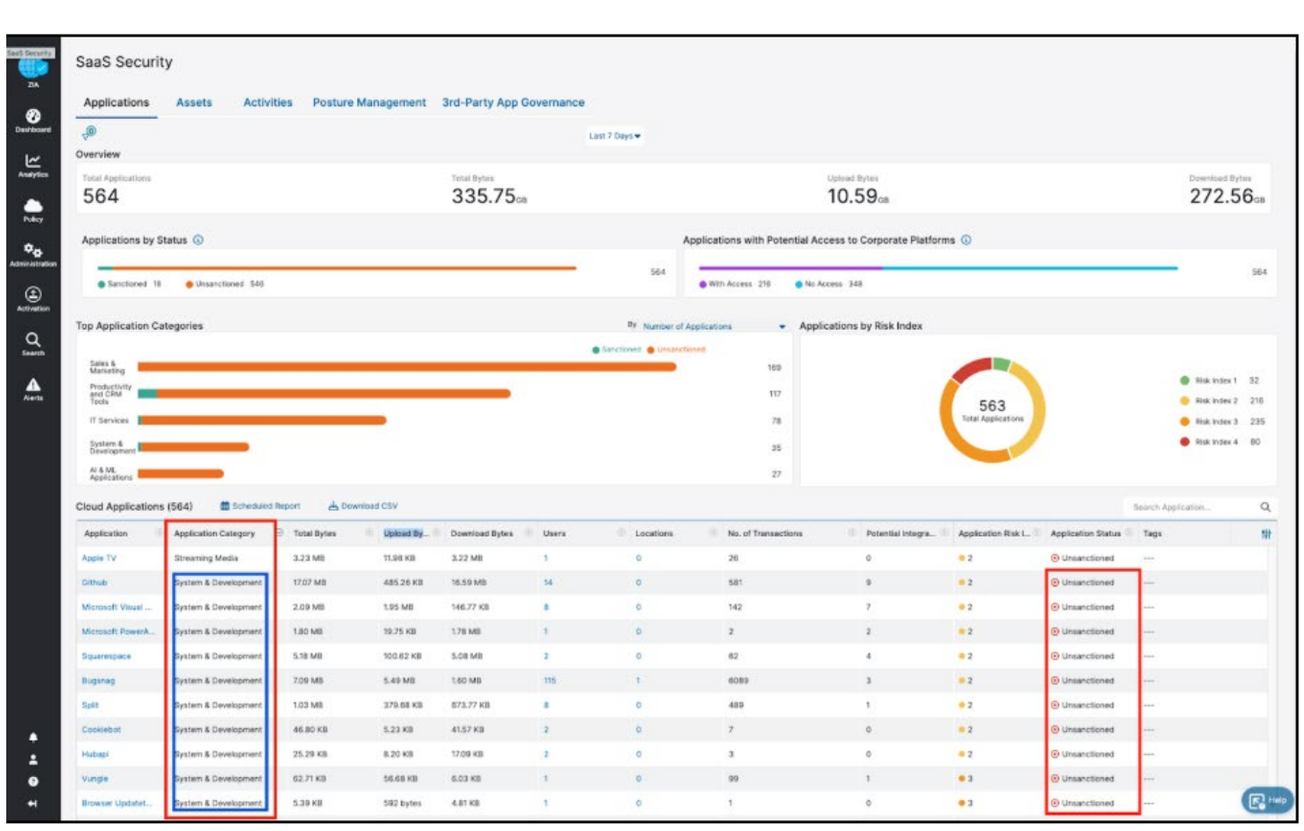
Figure 8: Zscaler Certificate

Refer to Appendix: PKI and Certificate Authorities if a custom intermediate certificate generated for the developers is needed. This section explains how a custom intermediate certificate can be requested and installed in ZIA so that a group of developers use their own certificate instead of Zscaler Root Certificate.

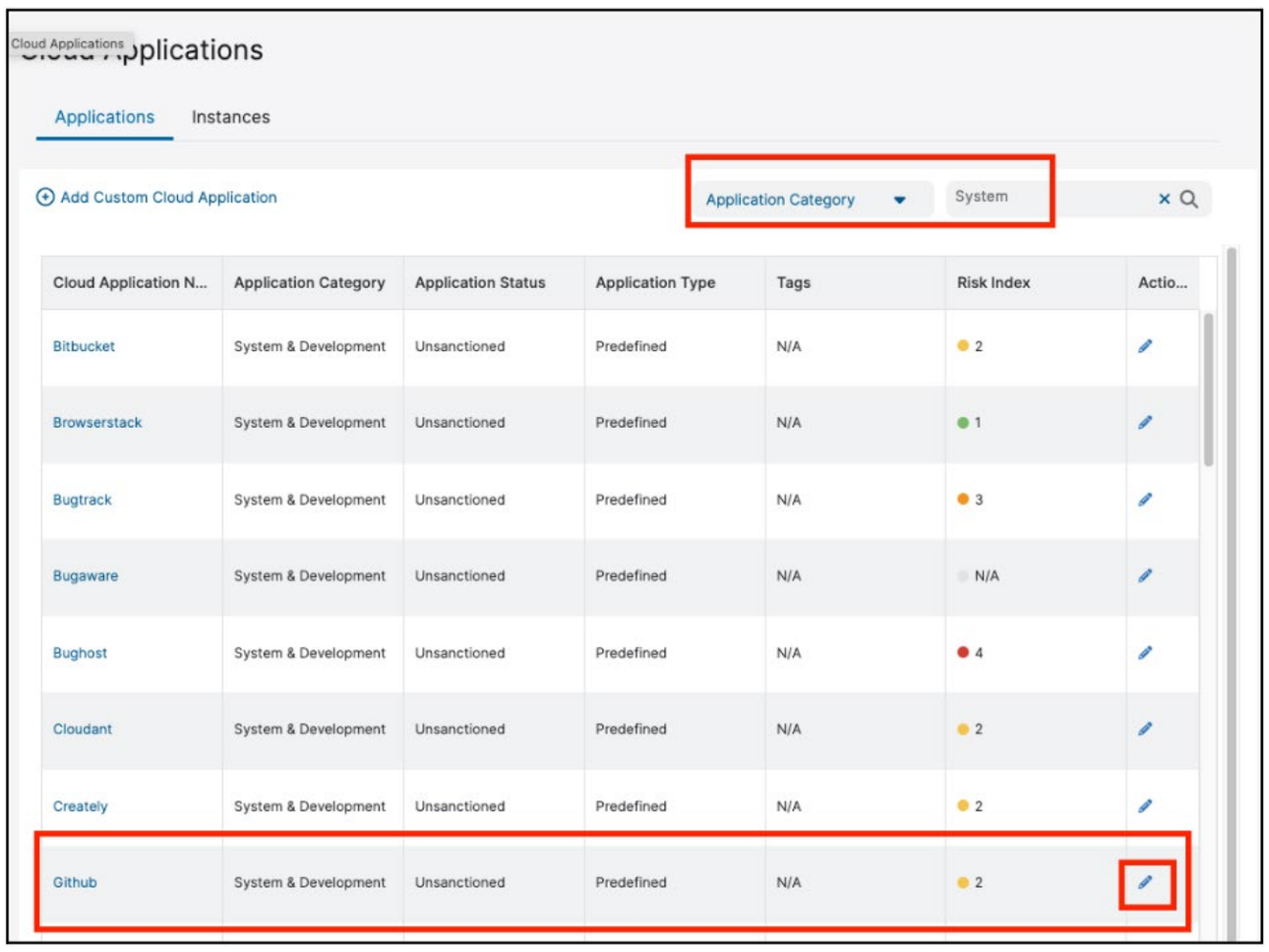
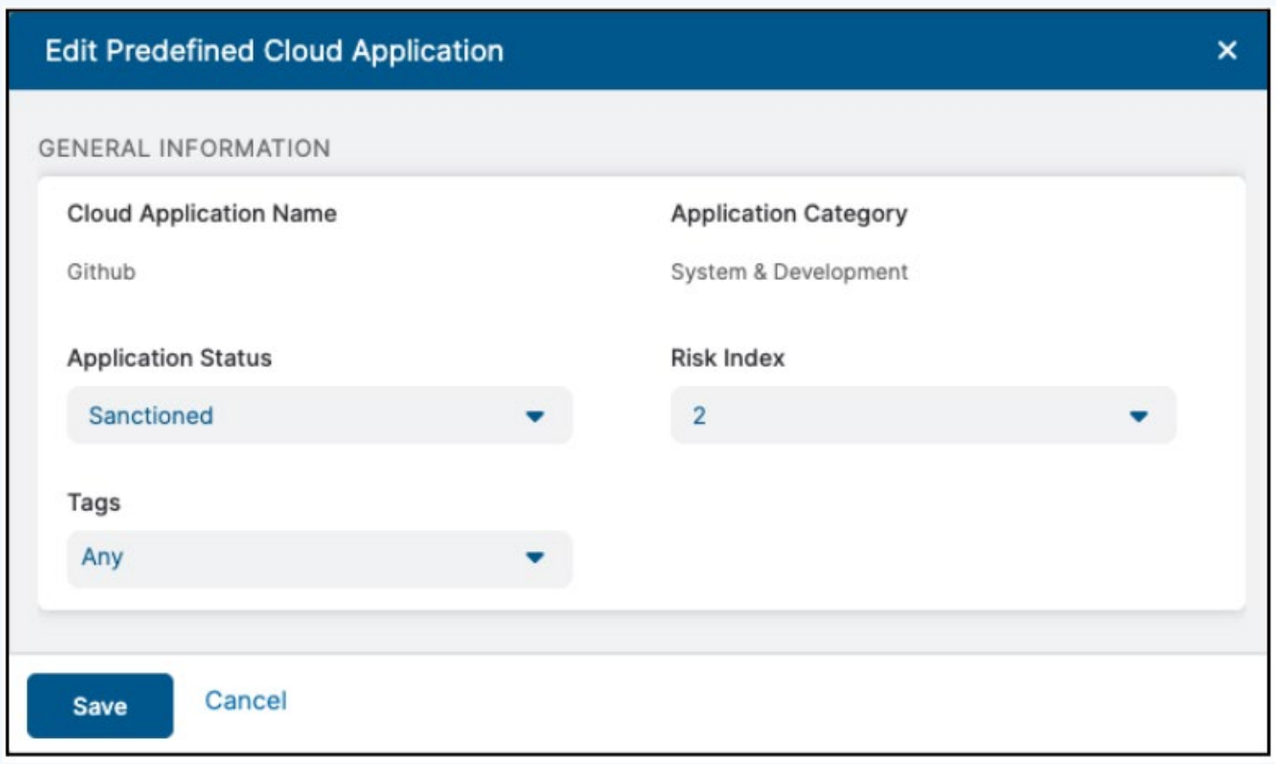
2.2.2 Step 2 – Developer Traffic Insights

It is expected at this point that Developers have installed the Zscaler Client Connector and are using it actively, but the majority of their traffic is not being inspected. With SSL/TLS Interception rules in place for the Developer group, we can use the logs and web insights to explore which sites will be problematic for the Developers. We will gather data from the following places: SaaS Security Applications and Web Insights Logs.

1. SAAS SECURITY APPLICATIONS

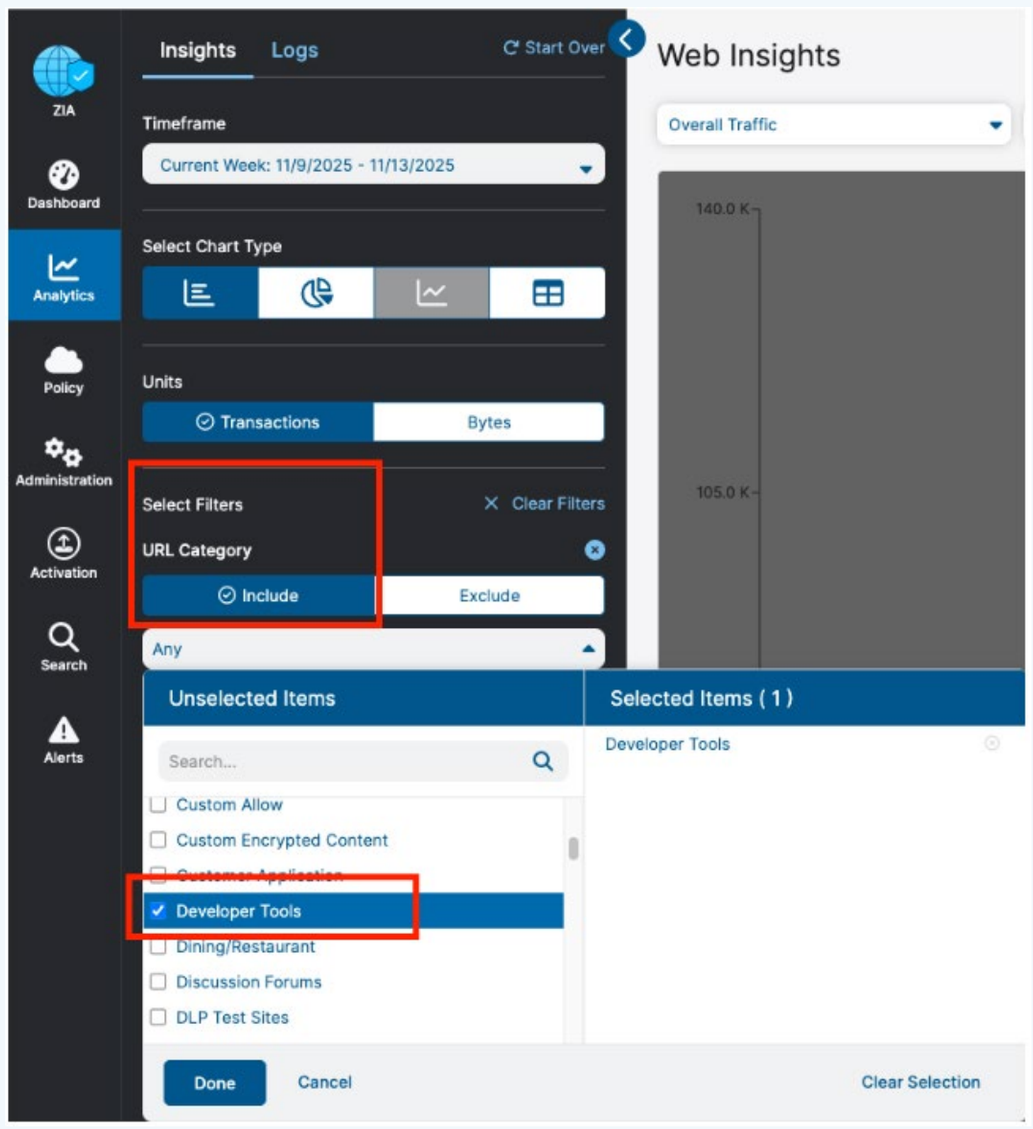
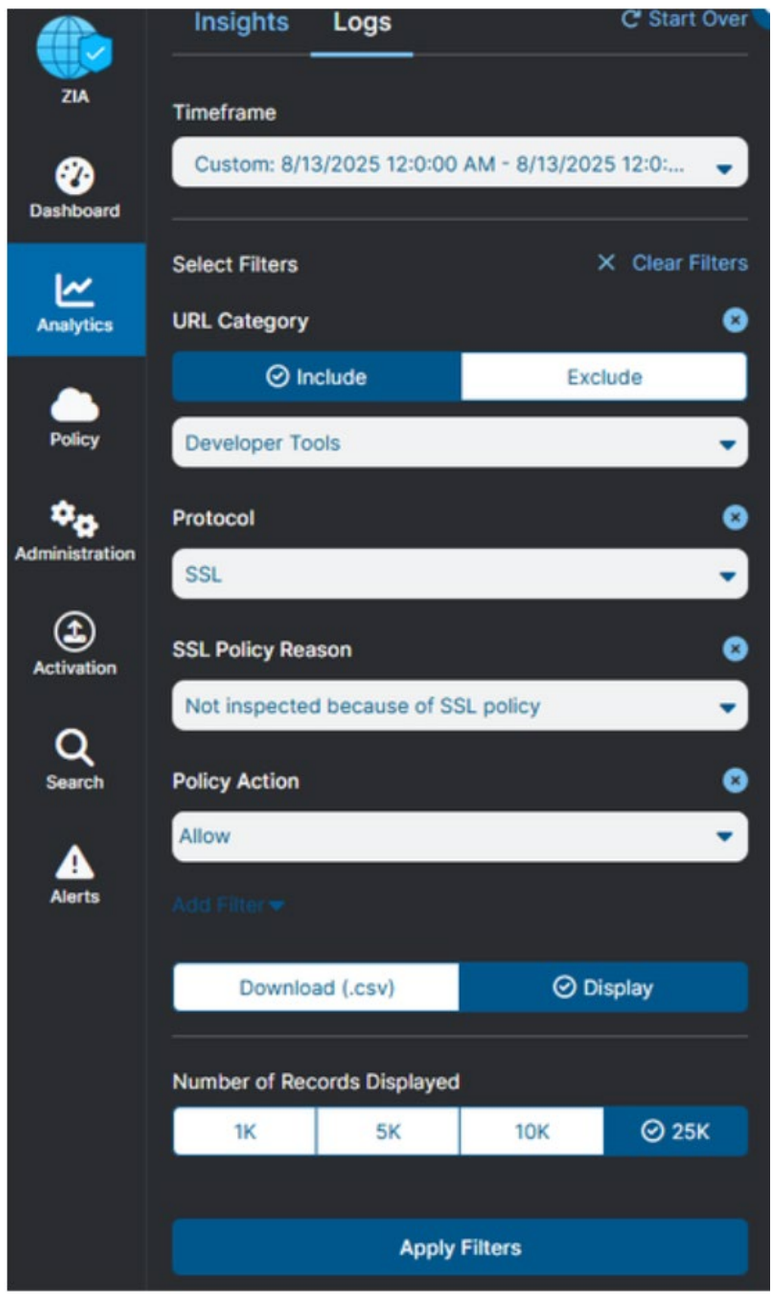
Step	Screenshot Reference
1. From the ZIA Admin Portal, go to SaaS Security applications. Analytics > SAAS SECURITY > Applications	
2. Order the Application Category column so you can easily find all of the Systems & Development applications. Refer to the Application Status column to see if the applications are Sanctioned or Unsanctioned.	

1. SaaS Security Applications

Step	Screenshot Reference
3. Record the applications (first column) that have significant traffic for your organization and evaluate together with the Developers if this is indeed an actively used application. If it is then change its status from Unsanctioned to Sanctioned in Cloud Applications.	
4. Use the filter option to narrow down either the application or the category. Click Edit.	
5. Change the application status from Unsanctioned to Sanctioned.v	

6. Repeat this for all applications that together with the Developer group it has been identified as critical to their workflow.

2. Web Insight Logs

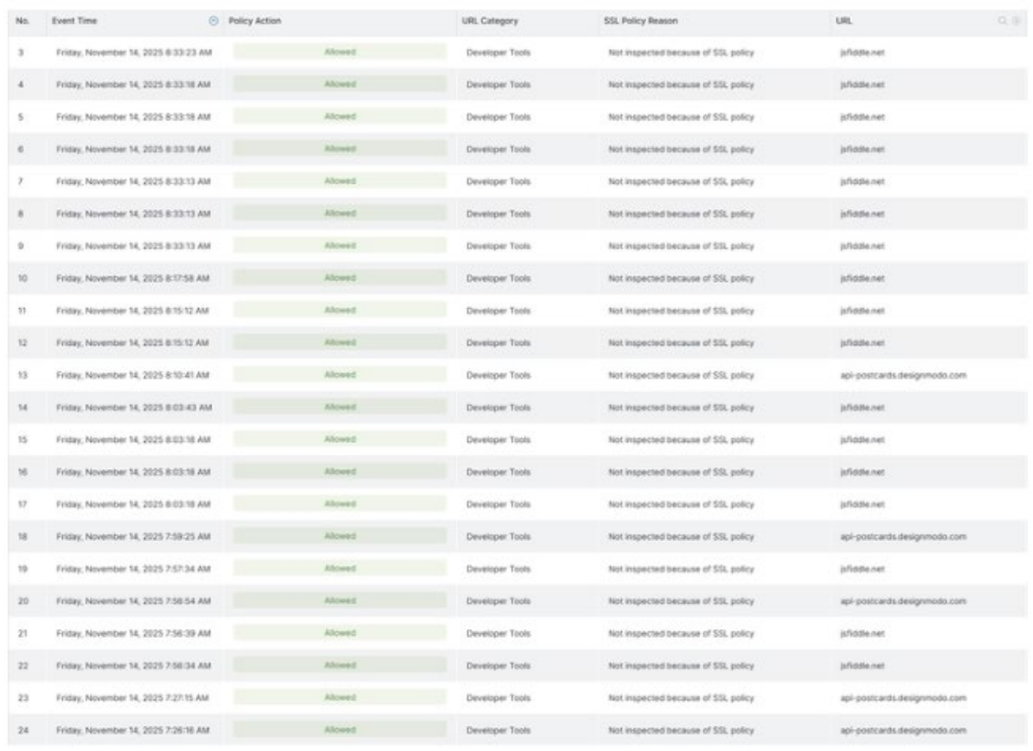
Step	Screenshot Reference
1. From the ZIA Admin Portal, go to Web Insights.	
2. Using the Web Insight Logs we can assess which sites and how many will be inspected. This is a direct correlation to the impact we will have in the Devel–oper's workflow. The following query will show us any sites that match. It is very possible that the Developers do not know what URLs are being used.	

Keep in mind that based on the time frame of the query, it can take some time to run. Once the results are returned, download them and gather only the URLs that interest the Developers. The screenshots below show the result of two queries so you can see what to expect:

Example Query 1

16	Wednesday, August 13, 2025 2:06:06...	Allowed	Developer Tools	Not inspected because of SSL policy	s1.designmodo.com
17	Wednesday, August 13, 2025 2:06:06...	Allowed	Developer Tools	Not inspected because of SSL policy	s1.designmodo.com
18	Wednesday, August 13, 2025 2:03:25...	Allowed	Developer Tools	Not inspected because of SSL policy	140-245-252-53.relay.splashtop.com
19	Wednesday, August 13, 2025 2:01:27 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
20	Wednesday, August 13, 2025 1:59:23 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
21	Wednesday, August 13, 2025 1:33:43 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
22	Wednesday, August 13, 2025 1:25:24 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
23	Wednesday, August 13, 2025 1:09:14 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
24	Wednesday, August 13, 2025 1:01:52 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
25	Wednesday, August 13, 2025 1:01:14 ...	Allowed	Developer Tools	Not inspected because of SSL policy	api.candu.ai
26	Wednesday, August 13, 2025 1:01:13 ...	Allowed	Developer Tools	Not inspected because of SSL policy	cdn.candu.ai
27	Wednesday, August 13, 2025 12:57:5...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
28	Wednesday, August 13, 2025 12:55:11...	Allowed	Developer Tools	Not inspected because of SSL policy	cdn.candu.ai
29	Wednesday, August 13, 2025 12:55:11...	Allowed	Developer Tools	Not inspected because of SSL policy	api.candu.ai
30	Wednesday, August 13, 2025 12:51:0...	Allowed	Developer Tools	Not inspected because of SSL policy	sc2-v3-dc.splashtop.com
31	Wednesday, August 13, 2025 12:50:0...	Allowed	Developer Tools	Not inspected because of SSL policy	sl-v3-univ-sns-win-3744-g3.api.splashtop...
32	Wednesday, August 13, 2025 12:49:3...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
33	Wednesday, August 13, 2025 12:45:0...	Allowed	Developer Tools	Not inspected because of SSL policy	sc2-v3-dc.splashtop.com
34	Wednesday, August 13, 2025 12:44:0...	Allowed	Developer Tools	Not inspected because of SSL policy	sl-v3-univ-sns-win-3744-g3.api.splashtop...
35	Wednesday, August 13, 2025 12:43:5...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
36	Wednesday, August 13, 2025 12:43:2...	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
37	Wednesday, August 13, 2025 12:33:0...	Allowed	Developer Tools	Not inspected because of SSL policy	api.candu.ai

Example Query 2



No.	Event Time	Policy Action	URL Category	SSL Policy Reason	URL
3	Friday, November 14, 2025 8:33:23 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
4	Friday, November 14, 2025 8:33:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
5	Friday, November 14, 2025 8:33:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
6	Friday, November 14, 2025 8:33:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
7	Friday, November 14, 2025 8:33:13 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
8	Friday, November 14, 2025 8:33:13 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
9	Friday, November 14, 2025 8:33:13 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
10	Friday, November 14, 2025 8:17:58 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
11	Friday, November 14, 2025 8:15:12 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
12	Friday, November 14, 2025 8:15:12 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
13	Friday, November 14, 2025 8:10:41 AM	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
14	Friday, November 14, 2025 8:03:43 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
15	Friday, November 14, 2025 8:03:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
16	Friday, November 14, 2025 8:03:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
17	Friday, November 14, 2025 8:03:18 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
18	Friday, November 14, 2025 7:59:23 AM	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
19	Friday, November 14, 2025 7:57:34 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
20	Friday, November 14, 2025 7:58:54 AM	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
21	Friday, November 14, 2025 7:56:39 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
22	Friday, November 14, 2025 7:56:34 AM	Allowed	Developer Tools	Not inspected because of SSL policy	jsfiddle.net
23	Friday, November 14, 2025 7:27:15 AM	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com
24	Friday, November 14, 2025 7:26:16 AM	Allowed	Developer Tools	Not inspected because of SSL policy	api-postcards.designmodo.com

3. Download the results and filter out duplicates and any other URL that has very few hits. Build the list of URLs that will be shared with the Developers for their validation.

2.2.3 Step 3 – Policy Tunning

The initial list provided by the Pre-defined URL Categories and Custom Categories can now be updated to include any other sites Developers will need. In other words, before you move to intercept traffic, the URL categories should have been enriched by the feedback from the developers and the information gathered from the steps discussed above.

The communication plan with Developers should allow for Developers to validate the final list of URLs. Referring to the Software Configuration Reference section below, customize the deployment script to include any additional configurations that are needed. If an application does not provide a programmatic method to fix SSL error but instead must be configured manually, include this information in the communication plan. For example, **Figure 9** shows Eclipse IDE's Site Download list that can only be changed by the running user, and its configuration is part of his session only.

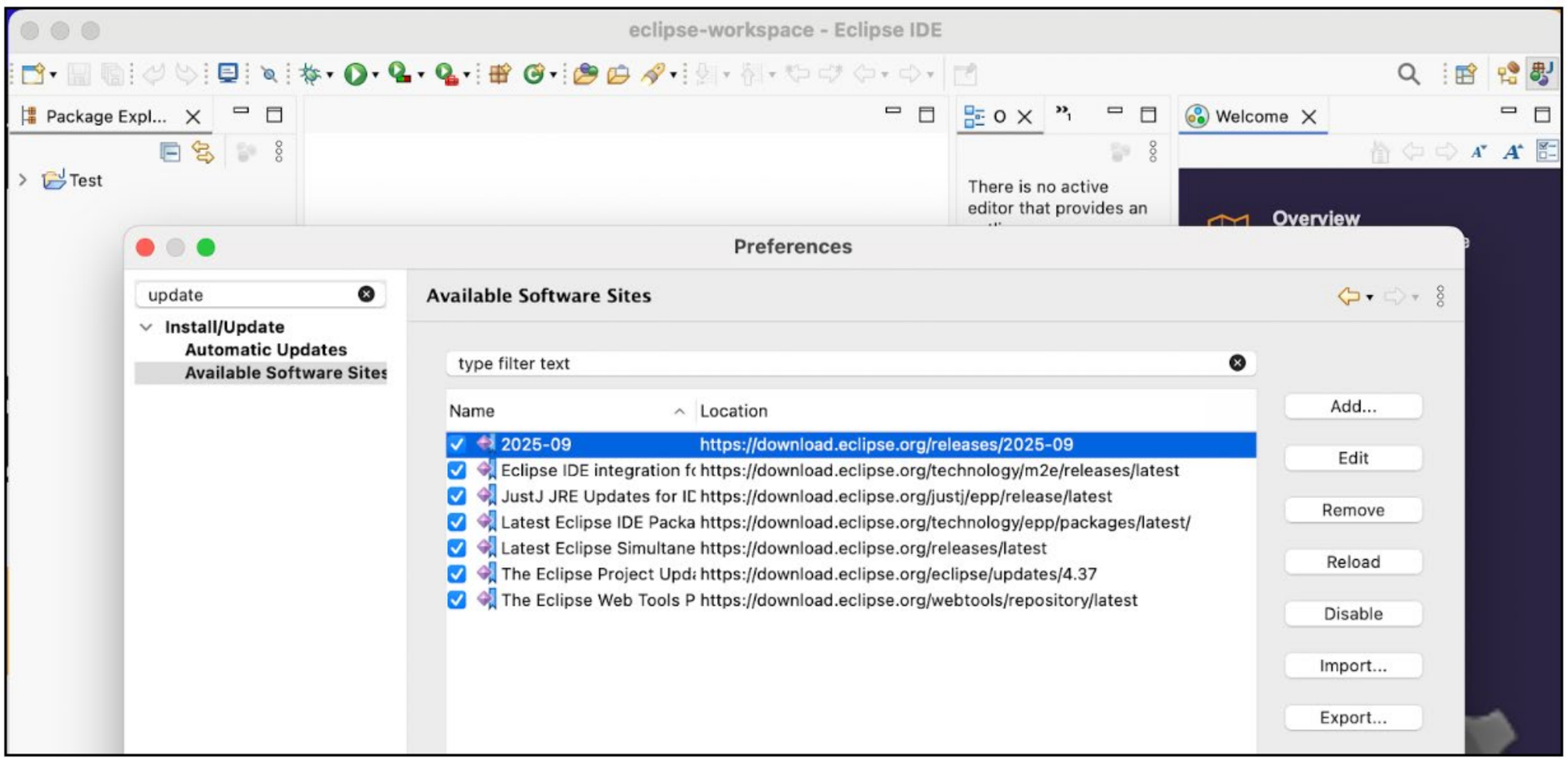


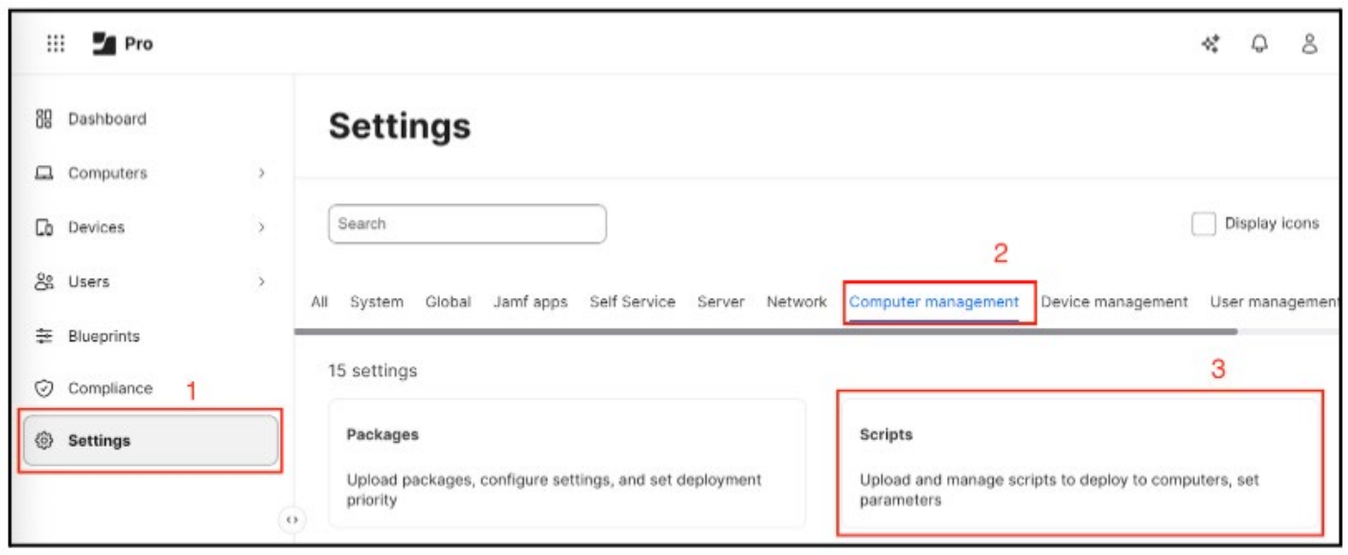
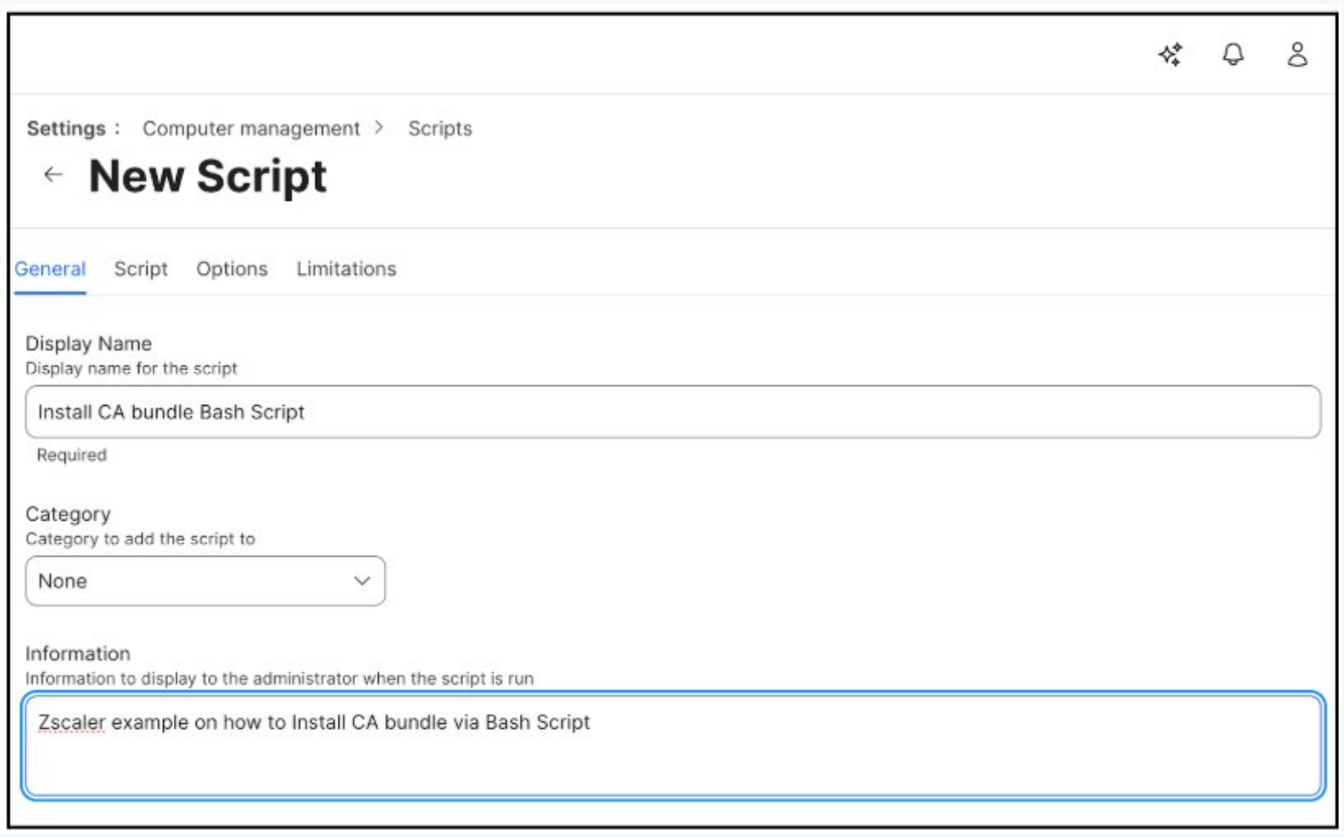
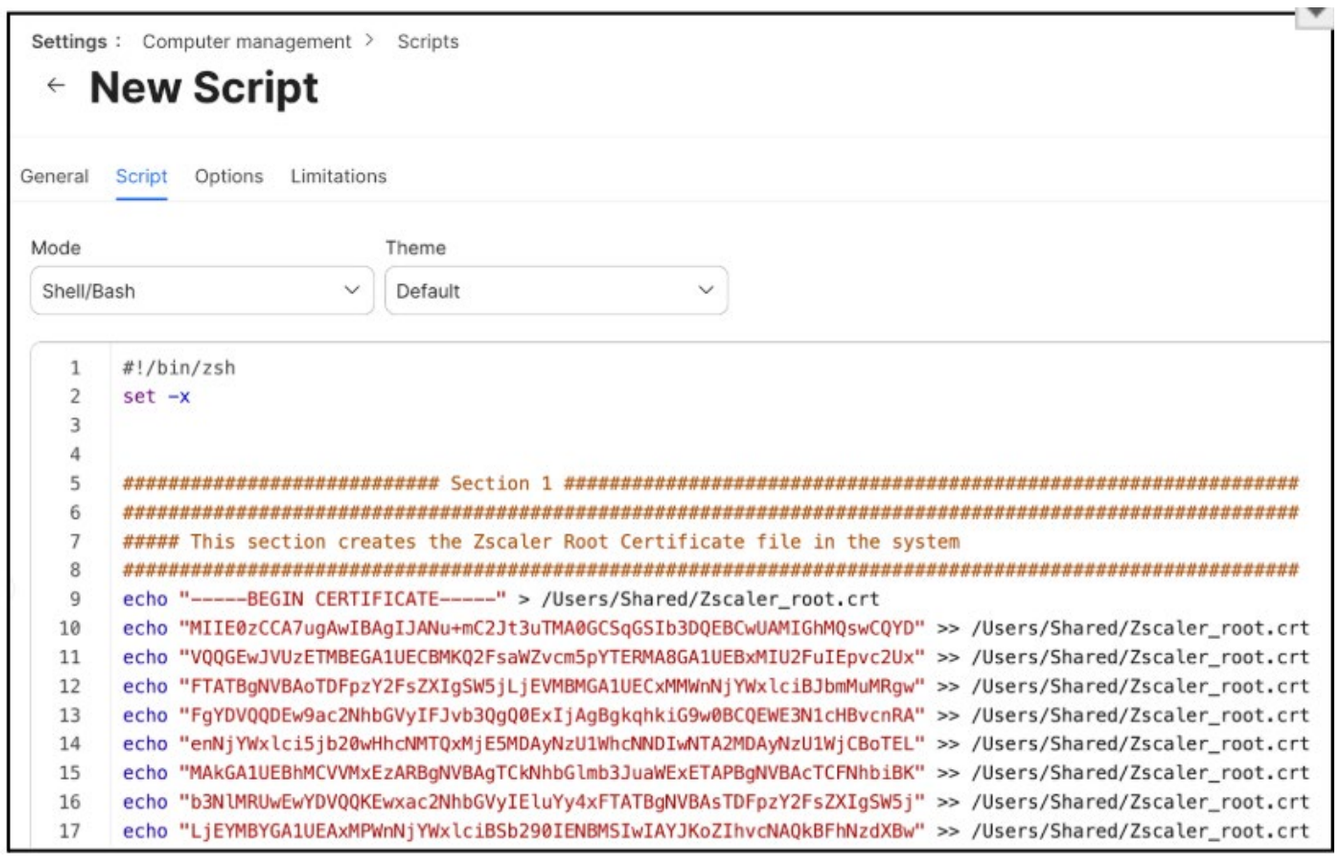
Figure 9: Eclipse IDE Preferences Available Download Sites List

Ideally any software that can be configured programmatically should use the provided bash script and be delivered to the target device using an MDM.

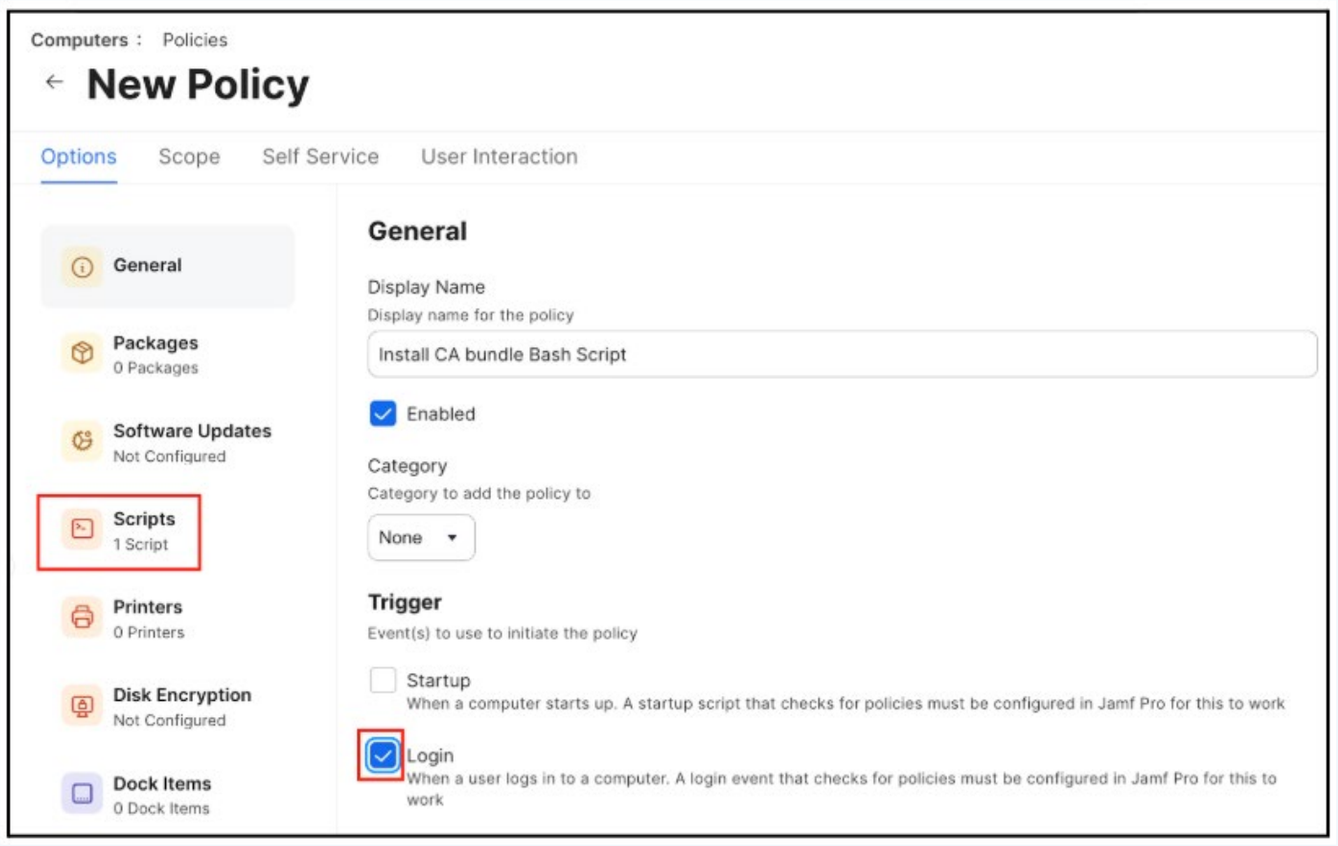
Let Developers know that the SSL Interception Policy Action is being changed from DO NOT ENCRYPT to DECRYPT and that the bash script has been deployed. If any SSL errors are seen after the change they can either fix them or contact the ZIA Administrator to start the troubleshooting process together.

JAMF PRO DEPLOYMENT OF BASH SCRIPT

To deploy a Bash Script using JAMF Pro, follow the steps in the table below.

Step	Screenshot Reference
1. In JAMF Pro, click Settings > Computer management > Scripts. Then click “New” on the top right.	
2. Update the new Script according to your environment. Refer to the screenshot to the right for an example.	
3. Provide the script in the relevant section.	

4. Once completed, the script is available for de-ployment as a policy.



5. Include a parameter value according to:

- --run_with_cron Install launchd daemon and env vars for users
- --run Only install envi-ronment variables for all users
- --only_cron Install the launchd daemon (run once as root)
- --remove_cron Remove the launchd daemon, script and plist

Note: It is important to note that the script should only be executed once per device, as it modifies all users found in the system. If it is executed several times, it will create duplicate configuration commands on the user environment.

An alternative option is to use JAMF Pro Self Service so the Developer has more control of the deployment time. If a Developer uses the Self Service deployment option, they will be able to see the output of the script if done via CLI.

Your Developers should have access to the Software Configuration section of this guide and the final bash script that was deployed. Make sure that Developers have deployed the script before moving to ENCRYPT their traffic. That way, when the switch from DO NOT ENCRYPT to DECRYPT happens, only a small number of URLs will present errors, as all other URLs already have a configuration that prevents the error at the Developer's workstation.

2.2.4 Step 4 – ZIA SSL Interception Final Configuration

Proceed to change the Action from DO NOT ENCRYPT to DECRYPT and let the Developers know of the change. Monitor traffic and follow the communication plan with the Developers to quickly fix any certificate errors that they encounter.

Keep in mind that, there is a chance that some of the software used by the Developers has no way to enable certificate bundles or to install an alternate root CA. In the Software Configuration Reference section, you will find per application guidance on how to enable the Zscaler certificate for correct TLS interception.

2.3 Other Aspects of the Software Developer Experience

The following sections address some other configurations and software components that can affect the Developer experience.

2.3.1 Zscaler Client Connector

Zscaler Client Connector plays a major role in the Developer's experience while using Zscaler Internet Access.

The preferred deploy mode for Zscaler Client Connector is via a Mobile Device Manager (MDM), such as JAMF or Microsoft Intunes. Management via MDM solutions not only enables unattended installations but also allows for advanced configuration like Strict Enforcement or pushing Firewall rules to the device.

If a policy configuration is removed on an MDM enrolled device, depending on its configuration it will simply get replaced by the same policy.

2.3.2 Zscaler Strict Enforcement and Fail-Close

Zscaler Strict Enforcement refers to configuration strategies designed to ensure that internet-bound traffic is unconditionally routed through the Zscaler cloud security platform, preventing any direct-to-internet access or proxy avoidance (more information can be found here). This is typically achieved through a combination of Zscaler Client Connector installation flags, (optionally) a PAC file, Client Connector configurations, and other network related restrictions.

Strict Enforcement ensures that users are not able to access the internet unless they log in via the Client Connector. If any user is logged out of the client, they will get a message to authenticate. Keep in mind that Strict Enforcement requires users to enroll with the app before accessing the internet and blocks traffic in the following situations:

- The user has not logged in after a new install.
- The user logs in and then logs out.
- An administrator removes the device from the ZCC portal.

Strict Enforcement does not affect users who have already logged in and disabled the ZIA service. The Developer needs to be aware if this feature is being actively used in their environment to not cause unexpected blockers in their workflow.

Refer to this help article for more information on how installation flags restrict network access.

2.3.3 Zscaler Client Connector Tunnel Version

Zscaler on the Client Connector offers two versions of the Z-Tunnel. Traffic is forwarded through the tunnel so that the ZIA Public Service Edge can apply the appropriate security and access policies. The Developer should have a discussion with your ZIA administrator on which ZTunnel strikes the correct balance between connectivity and security.

Understanding which Tunnel has been deployed helps provide insight into what conditions errors are seen by the Developer. The following shows some of the advantages of each ZTunnel type:

Z-TUNNEL 1.0

Z-Tunnel 1.0 forwards traffic to the Zscaler cloud via CONNECT requests, much like a traditional proxy. Version 1.0 sends all proxy-aware traffic or port 80/443 traffic to the Zscaler service.

For the developer use case, all internet websites commonly use port 443, and any custom deployed software on the local machine would make use of higher (+1024) port numbers. This allows the Developer to browse securely via ZIA but still deploy software locally and make requests to other machines in the local network.

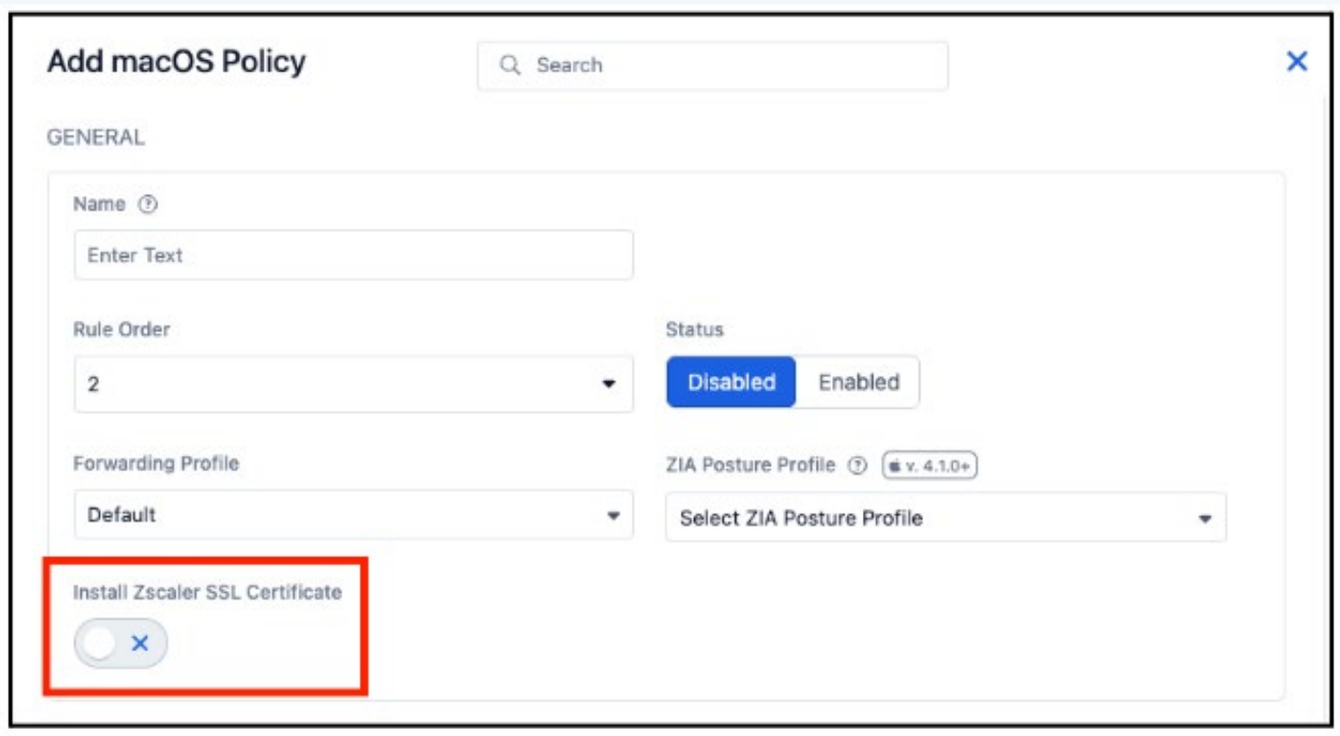
Z-TUNNEL 2.0

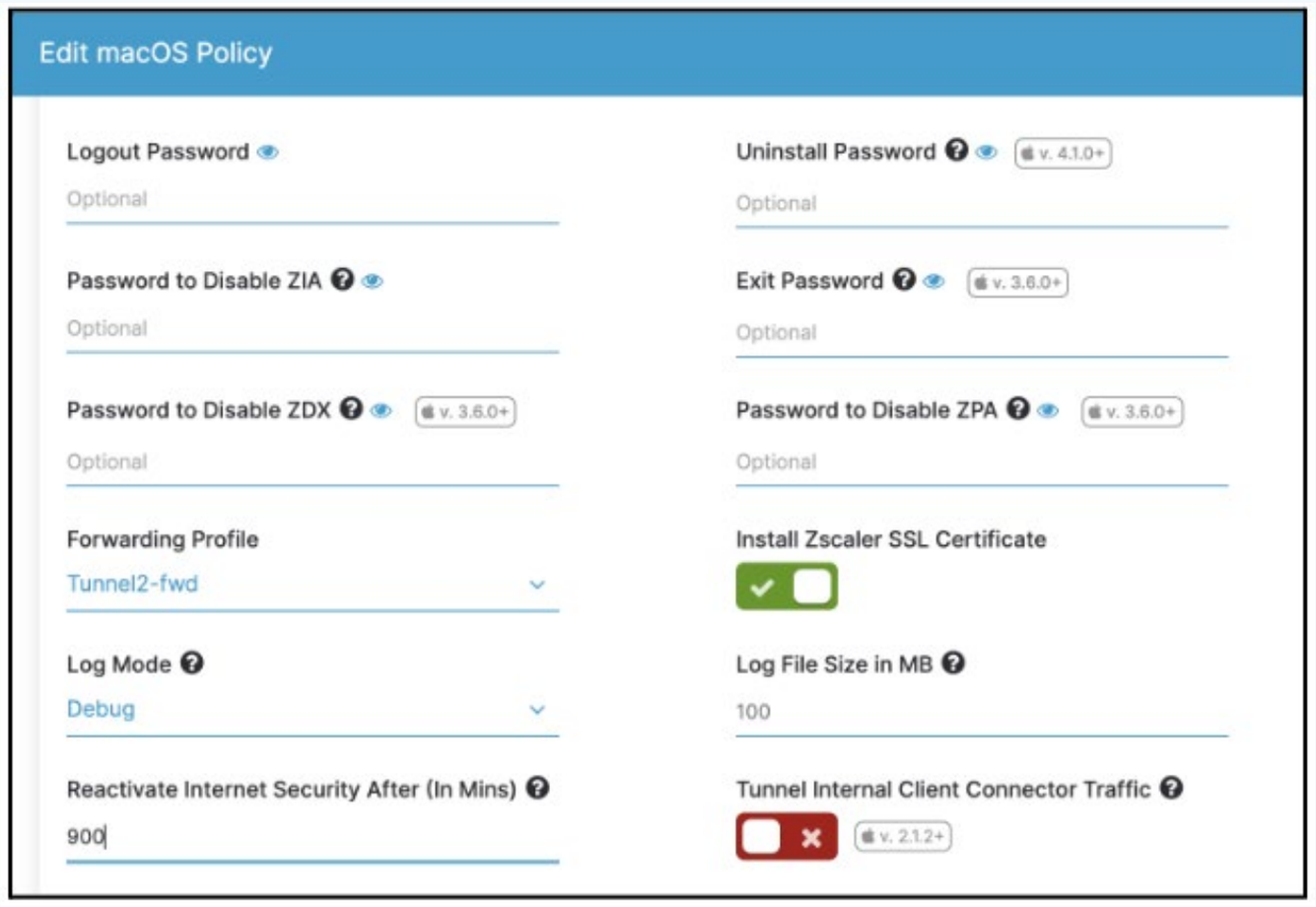
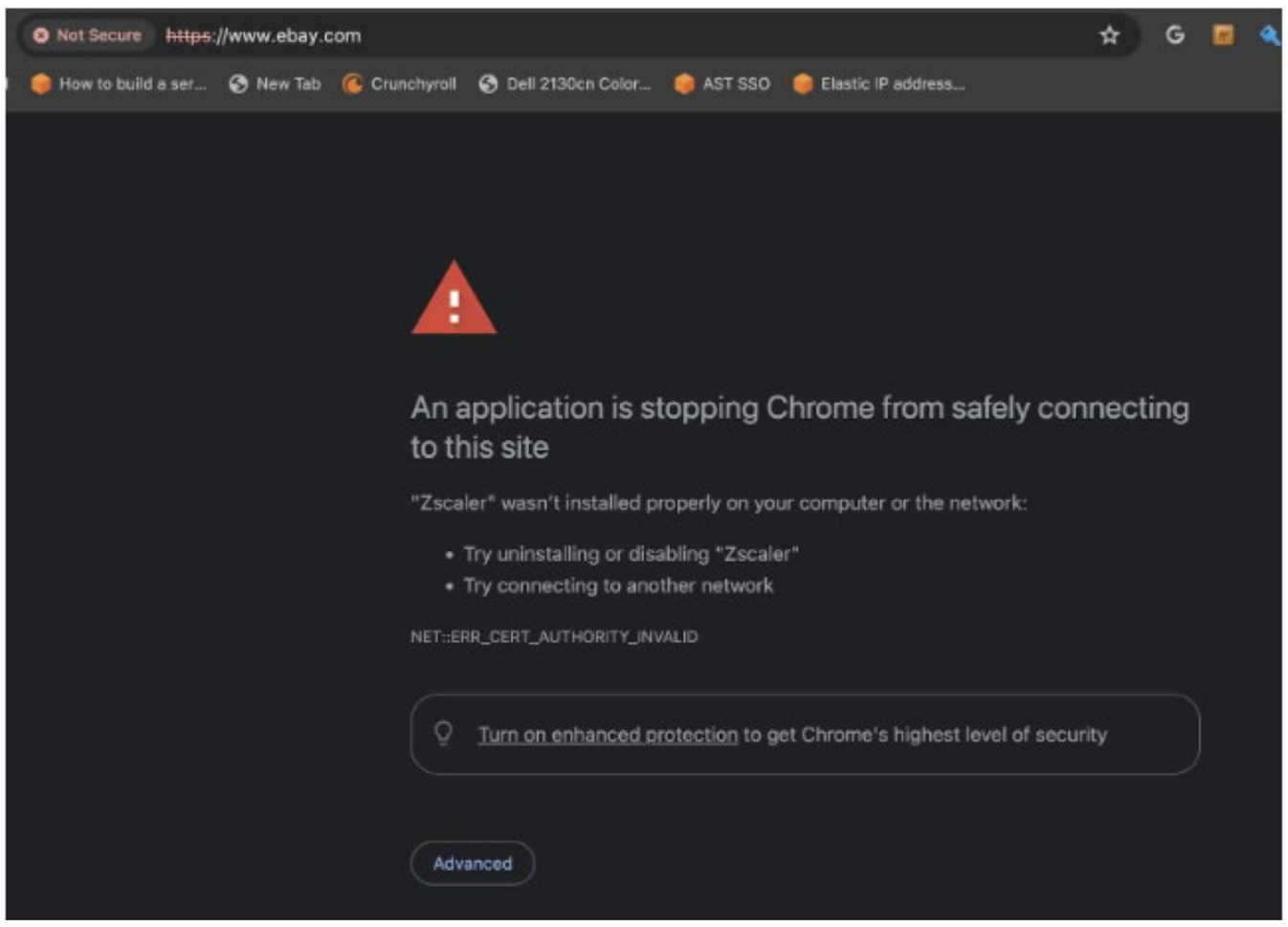
Z-Tunnel 2.0 has a tunneling architecture that uses DTLS or TLS to send packets to the Zscaler service. Z-Tunnel 2.0 is capable of forwarding to Zscaler all ports and protocols where the forwarding policy will determine if a given request is for the local network or for a website on the internet via Zscaler.

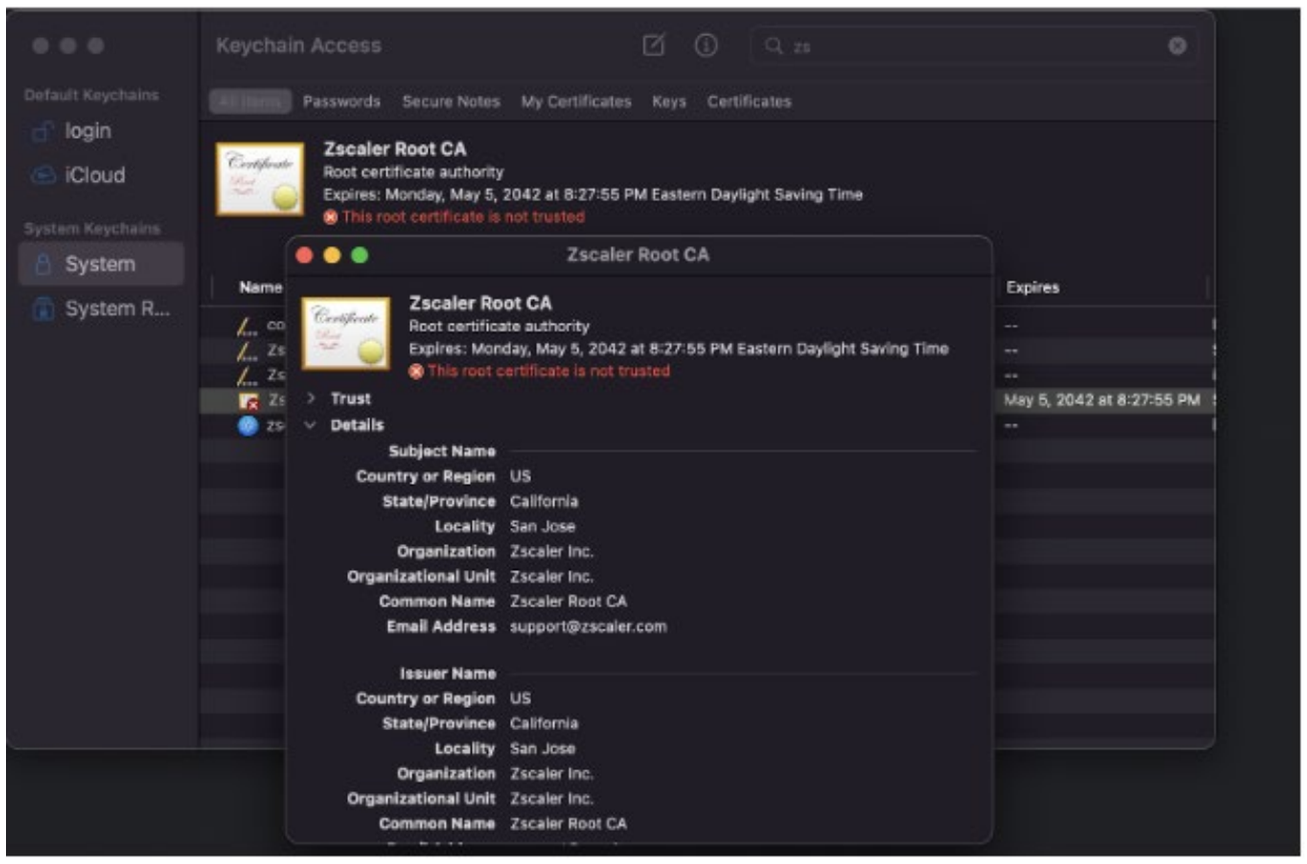
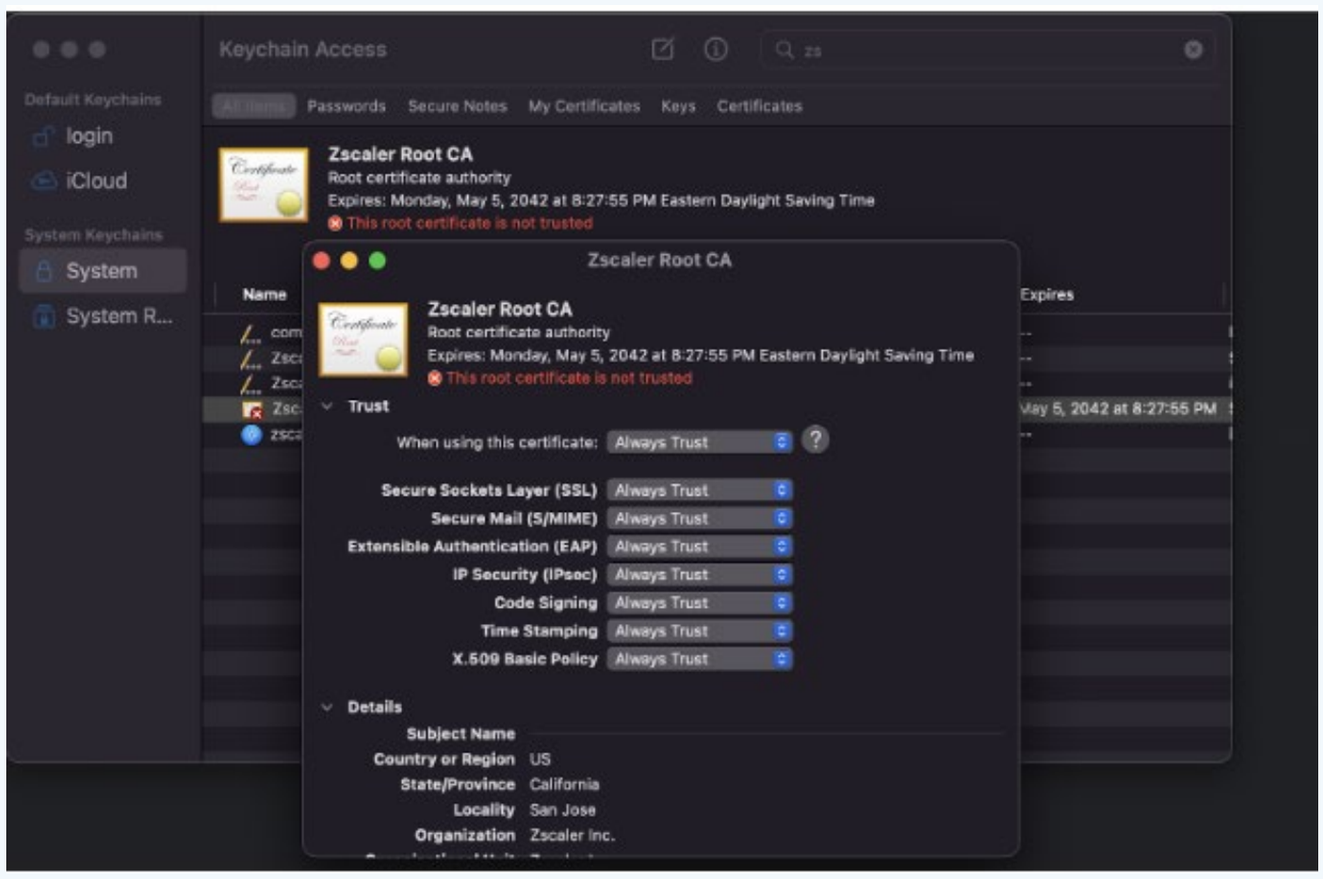
2.3.4 Zscaler Client Connector Automated Certificate Installation

This section will showcase how some configurations will affect the Developer local machine's Zscaler Client Connector. When the Client Connector is deployed using an MDM, it is advised to automatically install and trust the Zscaler root certificate. Therefore, by default the Zscaler Client Connector will not install the intermediate certificate for TLS/SSL Interception. Follow the following steps to accomplish this.

From the Zscaler Client Connector portal, the App Profile controls if the Intermediate certificate is installed. By default, no certificate is installed in the client machine. To leverage the Client Connector Install Zscaler SSL Certificate feature:

Step	Screenshot Reference
1. Download the intermediate certificate from the ZIA administration portal.	
2. Update the new Script according to your environment. Refer to the screenshot to the right for an example.	

Step	Screenshot Reference
3. Proceed to the App Profile to enable the certificate installation from the Client Connector automatically. An App Update and possible browser restart will be necessary. This is configured in the profile in Zscaler Client Connector Portal. Go to App Profiles > select Windows or Mac > add a new policy. You will find an option to enable “Install Zscaler SSL Certificate”. This will install the certificate mentioned above in the developer machine.	
In MacOS (if there is no MDM solution in place), it is necessary to also trust the installed certificate. In Microsoft Windows there is no need to make any changes. For Macs: 4. As per Big Sur Release Notes, improvements on system security are mandatory by requiring an interactive administrator password when changing certificate trust settings on the Certificate Store. If this is not done the following error will be seen:	

Step	Screenshot Reference
To trust the certificate and avoid the error above, follow these steps to Trust the installed certificate. 4a. Open KeyChain Access.	
4b. Open Trust within the certificate and change it to Always trust.	

2.3.5 DNS Control

Zscaler service typically includes some type of DNS modification or inspection. A full discussion of this topic is outside the scope of this document. It is important to have a discussion between the network/security team that manages Zscaler and the developer teams to understand if this will impact common developer activities. As an example, Internet bound DNS requests are often inspected, so a newly registered domain may trigger an action that a developer may not expect (possibly dropped or modified). In another example, internal zones may be handled differently to enable Zero Trust Network Access (ZTNA) through the Zscaler service (internal IP may be modified in the DNS response). Have the conversation up front to be certain that the developer community understands the architecture and the implications.

The following two rules are recommended for Developers on DNS Control Configuration:

Rule Order	Rule Name	Criteria	Action	Description
9	Block H/3 QUIC & ECH	DNS Request Type: HTTPS	Block with Response Code: “2 – Server Failure”	Blocks the fast path from DNS to H/3 or QUIC protocols.
10	Block DoH	Protocol DNS Over HTTPS	Block with Response Code: “2 – Server Failure”	Zscaler can apply poli–cy to DOH queries, but it is blocked here to ensure that ZPA is not bypassed in situations where Client Connector is in use.

2.3.6 Browser Isolation

This feature allows the Developer to browse URL categories that are considered risky, yet there is valuable content that can be found on those sites.

A Developer needs to be aware when a site is being rendered by Browser Isolation. Here is an example for how a HTTP request looks like without browser isolation:

```
% curl https://openai.com/chatgpt/

<!DOCTYPE html><html lang=”en-US”><head><title>Just a
moment...</title><meta http-equiv=”Content-Type” content=”text/
html; charset=UTF-8”><meta http-equiv=”X-UA-Compatible”
content=”IE=Edge”><meta name=”robots” content=”noindex,nofollow”><meta
name=”viewport” content=”width=device-width,initial-scale=1”>
```

Here is an example of the same web site with browser isolation:

```
% curl https://openai.com/chatgpt/

<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML 4.01
Transitional//EN” “http://www.w3c.org/TR/1999/REC-
html401-19991224/loose.dtd”>

<html>

<head>

<meta name=”description” content=”Zscaler makes the internet safe for
businesses by protecting their employees from malware, viruses, and
other security threats.”>
```

Zscaler Browser Isolation renders the risky web page on an airgapped computer that isolates it from the Developer computer, yet it does allow certain configurable interaction options that are relevant for a Developer use case:

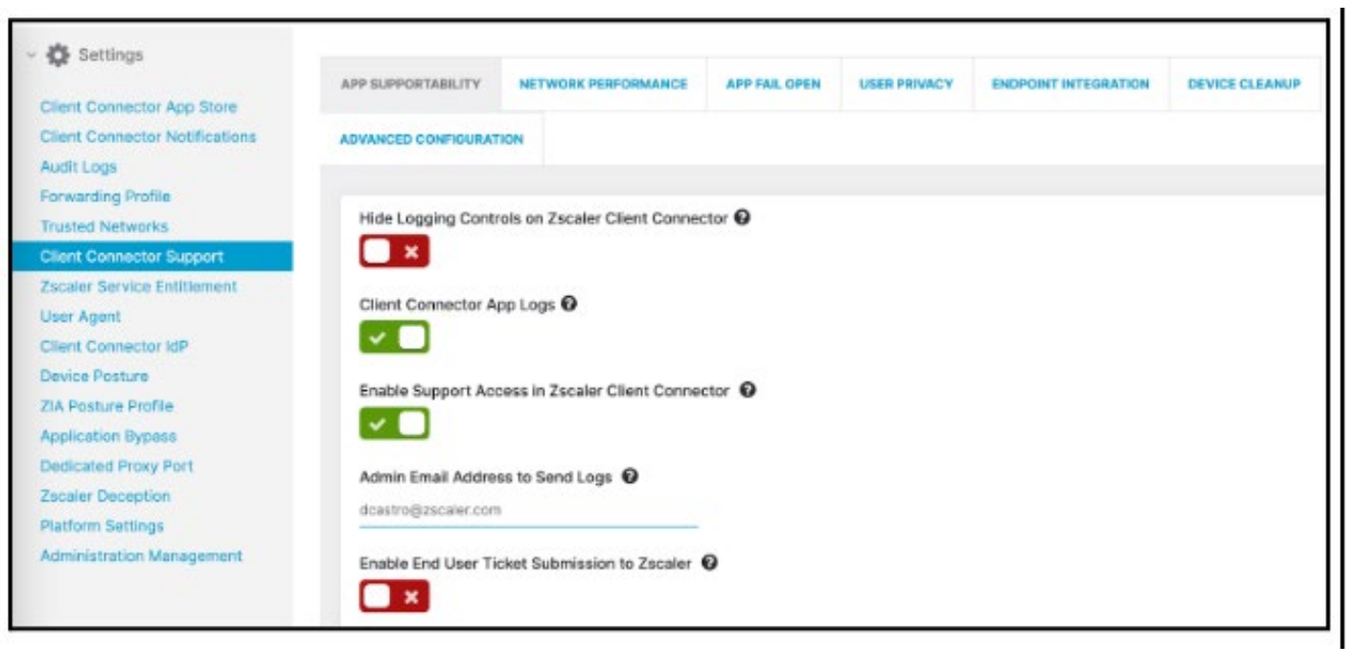
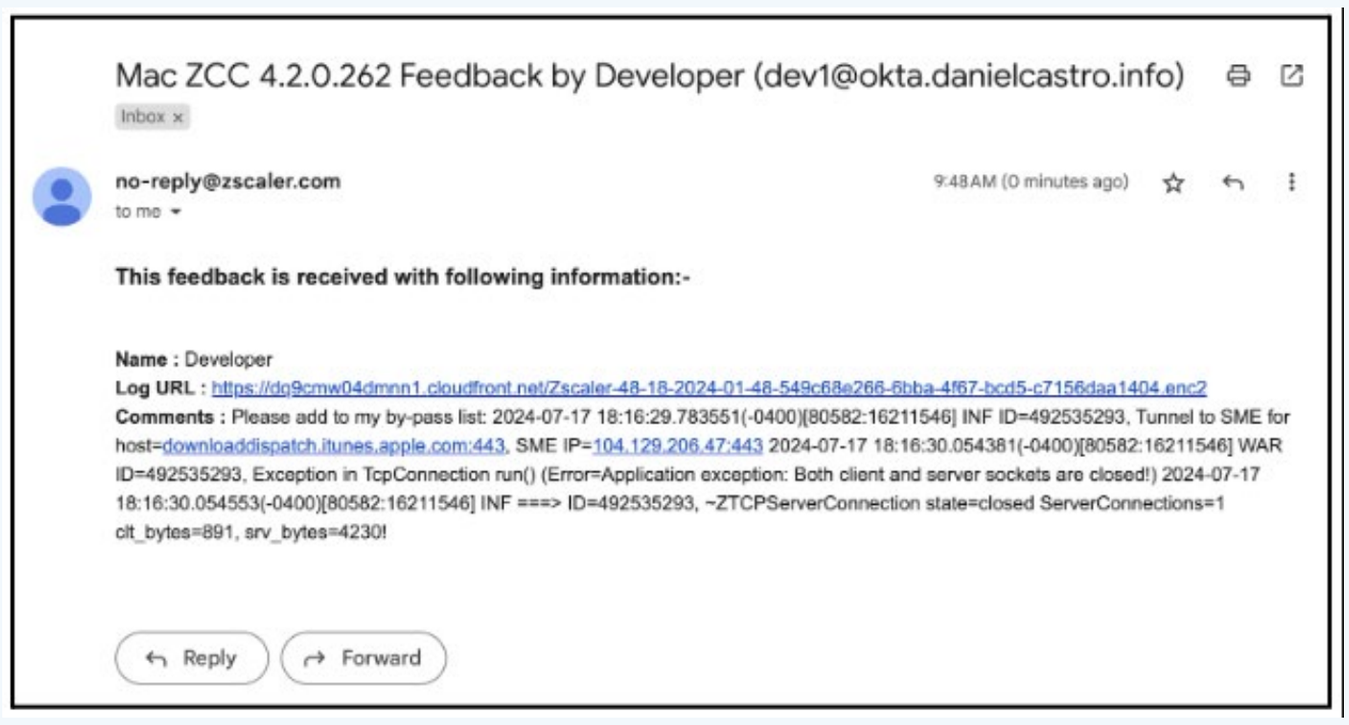
- **Copy / Paste:** This feature can be turned on/off and will allow the Developer to copy and paste text between the browser isolation and their local machine.

- **File Transfer:** This configuration allows the Developer to upload files or download files (or both) to the isolated browser session running within Zscaler.

- **URL Obfuscation / Original URL:** The URL that was browsed to can be obfuscated or shown as is to the Developer. This matters as this URL might be shared between Developers and only the original is real.

2.3.7 Enable feedback from User to Administrator

The Zscaler Client Connector has the ability to send an email with the relevant data that will allow the ZIA Administrator to work with the Developer to create bypass rules in a short time frame.

Step	Screenshot Reference
1. Administrator for Zscaler Client Connector must to turn on “Enable Support Access in Zscaler Client Connector” and specify an email address that will re-ceive the communication.	
2. Once a user provides feedback via the feature, it will be received in the configured email address.	
3. The email will also contain When, Who and What URL failed. This can be corroborated in the Web Insights Logs.	

Note: When dealing with very large files and looking for a particular entry in Web Insight Logs, it is easier to download the CSV file and find relevant entries using grep or a similar search application.

The administrator can now add the URL to the user/group bypass list and allow the app to run normally.

Software Configuration Reference

Some applications use their own application trust store instead of using the default system store. When this is the case, the application is not able to validate the TLS interception certificate. If an application does not use the system certificate store, it usually provides a way to load a certificate bundle that indicates which root certificates to trust. This section presents a Bash script for MacOS that serves as an example on how to configure individual certificates that are installed on a system.

A certificate bundle or "ca-bundle" file commonly refers to a file that contains a list of root certificates and normally replaces the system certificate store for a set of applications. We will provide an example to build a "ca-bundle.pem" file with the Zscaler root certificate and the Mozilla common system root CAs.

In each application subsection, you will find the required steps to use the generated ca-bundle file to be loaded into the application specific trust store. The general guidance is to use environment variables as this is the most broad approach to enabling certificate bundles. In some cases, individual configurations are employed for the specific tool or libraries.

Zscaler's online documentation has the initial list of applications: Adding Custom Certificates to Applications. In the following subsections, you will find applications that are not on that list or more details are provided on a particular application.

3.1 Build a Certificate Bundle

In order to build the ca-bundle file, start from the Mozilla Firefox CA Bundle (cacert.pem can be downloaded here) and add the Zscaler root certificate in PEM format at the end of the file. The ca-bundle filename extension must match what format is required by each application – this is commonly "pem" or "crt". If your administrator follows this guide, your root Zscaler certificate will be installed already in your machine's certificate store and can be exported (for example, as ZscalerRootCertificate-2048-SHA256.crt).

- For Mac you would use Keychain Access, right click on the certificate in question (Zscaler or custom), and choose Export from the menu.

- For Windows you would open the Certificate Manager MMC snap-in, select System Certificates and Right-click the certificate. Select All Tasks, and then select Export. Finish the Export Wizard and use Base 64 Encoded Format. If this option is grayed out, then select PKCS#12 DER format, but this will need to be converted to PEM format.

Merge the two files using a text editor or with the following command in Mac or Linux:

```
cat cacert.pem ZscalerRootCertificate-2048-SHA256.crt > ca-bundle.pem
```

3.2 Installing a CA Bundle with Bash

Download the script from Appendix 5.1 Install CA bundle Bash Script. In this section we explain the example Bash script that contains 6 sections within it.

1. The Zscaler Root Certificate file in the system.
2. Make a CA bundle file.
3. Exports the generic system env variables, and based on the existence of pre-determined software, it stores an appropriate env variable in a file.
4. Mac-specific configurations.
5. Creates the environment variable with path value that were detected and Iterates over all users and installs the environment variable list.
6. Iterate over all users in the system and modify the bash_profile and zshrc files with application specific instructions.

The script will take the Zscaler root certificate and Mozilla Firefox certificate bundle to create a new certificate bundle that is installed in some applications and for all users found in the system. This script does not include all possible applications, as it is expected that each administrator will modify this script to its individual environment.

This bash script is designed to be pushed via an MDM solution (JAMF Pro Deployment of Bash Script), but can also be executed locally by the user or administrator.

3.3 Custom Certificate Application Enablement

The following subsections will show you how to enable the ZScaler Root Certificate on a per application basis.

3.3.1 AWS-CLI v1 and v2

This section describes the SSL configuration of the two AWS CLI versions.

The Zscaler certificate error will be seen similar to this:

```
SSL validation failed for https://sts.amazonaws.com/ [SSL:
CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get
local issuer certificate (_ssl.c:1145)

SSL validation failed for https://ec2.us-west-2.amazonaws.com/ [SSL:
CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get
local issuer certificate (_ssl.c:1145)
```

AWS – CLI v1 and v2		
Tested Version	**Documentation**	**Notes**
aws-cli/2.15.33 Python/3.11.8	Version 1: https://docs.aws.amazon.com/cli/v1/userguide/cli-configure-envvars.html Version 2: https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-options.html	
Configuration Options		
Command Line Arguments		
`--ca-bundle <string>`		
Environment Variable		
`export AWS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem`		
Configuration File		

`% aws configure`
`When the above is run the following changes are done in the configuration file`
`ca_bundle="file.pem"`
`Specifies a CA certificate bundle (a file with the .pem extension) that is used to verify SSL certificates. Can be overridden by the AWS_CA_BUNDLE environment variable or the --ca-bundle command line option.`		

3.3.2 Curl

This section describes the SSL configuration for the curl application.

Curl		
Tested Version	**Documentation**	**Notes**
8.4.0 with libreSSL 3.3.6	https://curl.se/docs/sslcerts.html	On Mac OS X makes use of the system certificate store On Debian Linux distribution curl uses ca-certificates. See more information on 3.14 AP.
Configuration Options		
Command Line Arguments		
`--with-ca-file=<folder> or --with-ca-bundle=<file> OR --cacert <file> or --ca-path <folder>`		
Environment Variable		
`[System.Environment]::SetEnvironmentVariable("CURL_CA_BUNDLE", "<Path to Certificate>\ca-bundle.pem", "Machine")`		

3.3.3 Docker

This section describes the SSL configuration of various Docker components.

Docker		
Docker Image	**Docker Daemon**	**Notes**
Tested software version: Docker 26.1 Alpine latest (3.20.1) Ubuntu latest (24.04), Windows Server Core (2022) Official Documentation Link: https://docs.docker.com/build/building/best-practices https://docs.docker.com/reference/dockerfile/#copy https://docs.docker.com/reference/dockerfile/#run	Tested software version: 26.1 Official Documentation Link: https://docs.docker.com/engine/security/certificates	Docker daemon also accesses the certificate store of the operating system by default, so that a separate configuration of Docker is not required.

Configuration Details

Ubuntu Linux

```
FROM ubuntu:latest
# Don’t ask any command line questions
ENV DEBIAN_FRONTEND=noninteractive
# Copy the certificate to the docker image file system in a trusted location.
On Ubuntu the file needs to have the extension .crt otherwise it will no be
accepted
COPY ca-bundle.crt /usr/local/share/ca-certificates/
# Create the ssl directory because it does not exists at this point
RUN mkdir -p /etc/ssl/certs
# Append the ca certificate to the main certificate file manually because oth-
erwise it is probably not possible to reach the internet
RUN cat ‘/usr/local/share/ca-certificates/ca-bundle.crt’ >> /etc/ssl/certs/
ca-certificates.crt
# Update package list
RUN apt-get update
# Install certificate management
RUN apt-get install -y ca-certificates
# Update the certificates
RUN update-ca-certificates
# More commands…
```

Alpine Linux

```
FROM alpine:latest
# Copy the certificate to the docker image file system in a trusted location
COPY ‘ca-bundle.pem’ /usr/local/share/ca-certificates/
# Create the directory because it does not exists at this point
RUN mkdir -p /etc/ssl/certs
# Append the ca certificate to the main certificate file manually because oth-
erwise it can be impossible to reach the internet
RUN cat ‘/usr/local/share/ca-certificates/ca-bundle.pem’ >> /etc/ssl/certs/
ca-certificates.crt
# Install ca-certificates package
RUN apk add --no-cache ca-certificates
# Update CA certificates
RUN update-ca-certificates
# More commands…
```

Windows Server Core 2022

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022
# Copy the certificate to the docker image file system
COPY ca-bundle.pem ./
# Import the certificate into the trust store of the computer
RUN powershell -Command Import-Certificate -FilePath ca-bundle.pem -CertStore-
Location Cert:\LocalMachine\Root\
# More commands…
```

3.3.4 Git

This section describes the SSL configuration for the git applications.

Git		
Tested Version	**Documentation**	**Notes**
2.38.1	https://git-scm.com/docs/git-config	Git can be configured per individual repository or for all repositories. On Mac OS X git uses the system certificate store.
Configuration Options		
Command Line Argument		

```
# one repository
git config --global http.”https://repos.sample.com”.sslCAInfo /path_to_file/
cert.pem <string>
# all repositories
git config --global http.sslCAInfo /path_to_file/ca-bundle.pem
```

Windows System Store
`git config --system http.sslbackend schannel`

3.3.5 NPM

This section describes the SSL configuration of various node package manager applications.

NPM		
Tested Version	**Documentation**	**Notes**
10.2.3	https://nodejs.org/api/cli.html#cli_node_extra_ca_certs_file	Curl on Mac OS X makes use of the system certificate store.
Configuration Options		
Command Line Argument		
`--with-ca-file=<folder> or --with-ca-bundle=<file>`		
Environment Variable Windows		
`[System.Environment]::SetEnvironmentVariable(“NODE_EXTRA_CA_CERTS”, “<Path to Certificate>\ca-bundle.pem”, “Machine”)`		
Environment Variable Mac		
`launchctl setenv NODE_EXTRA_CA_CERTS <Path to Certificate>/ca-bundle.pem`		
Environment Variable Linux		
`echo “export NODE_EXTRA_CA_CERTS”=<Path to Certificate>/ca-bundle.pem” >> $HOME/.bashrc`		

3.3.6 Oracle Java

This section describes the SSL configuration of various Java components.

JAVA		
Tested Version	**Documentation**	**Notes**
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)	https://docs.oracle.com/en/java/javase/12/tools/keytool.html	Oracle Java uses its own certificate store, which is normally stored in the cacerts file in the Java Runtime installation directory. To manage the certificate store, Java provides the command line tool keytool, which is also located in the Java installation directory. The default password for the Java certificate store is changeit and is required for changes to the certificate store.

Configuration Details

Command Line Argument

```
keytool -importcert -file "<Path to Certificate>/ca-bundle.pem" -alias mycert -keystore "<Java Runtime Install Path>/lib/security/cacerts" -storepass changeit
```

Note: The ca-bundle.pem file must have system-trusted certificates stored along with Zscaler CA certificate or customer custom CA.

3.3.7 Python

This section describes the SSL configuration of various Python components and libraries. Use the following table to modify python in your local system.

Python 3.4.1 or later		
Tested Version	**Documentation**	**Notes**
3.14.0aO	https://www.python-httpx.org/environment_variables/#ssl_cert_file	Python 3.4.1 and earlier do not do certificate validation by default. If you are using virtual environments, then use export on the terminal instead of launchtcl.
Configuration Options		
Environment Variable Windows		
`[System.Environment]::SetEnvironmentVariable("SSL_CERT_FILE", "<Path to Certificate>\ca-bundle.pem", "Machine")`		
Environment Variable Linux		
`echo "export SSL_CERT_FILE=<Path to Certificate>/ca-bundle.pem" >> $HOME/.bashrc`		
Environment Variable Mac		
`launchctl setenv SSL_CERT_FILE <Path to Certificate>/ca-bundle.pem`		

If the developer would rather include the bundle in the python code follow the following guidance:

To verify if indeed the environment is being routed through Zscaler ZIA, the following Python code below can be used to verify the peer certificate:

```
from socket import socket

import ssl

import M2Crypto

import OpenSSL

# M2Crypto

cert = ssl.get_server_certificate(('www.google.com', 443))

x509 = M2Crypto.X509.load_cert_string(cert)
```

```
print x509.get_subject().as_text()

x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM,
cert)

print x509.get_subject().get_components()
```

The following code will return the ssl_cert_dir CA certificate store path:

```
python -c “import ssl; print(ssl.get_default_verify_paths())”
```

If behind ZIA you will get the following error:

```
File “~/.pyenv/versions/3.14-dev/lib/python3.14/urllib/request.py”,
line 1322, in do_open

 raise URLError(err)

urllib.error.URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_
FAILED] certificate verify failed: Missing Authority Key
Identifier (_ssl.c:1020)>
```

There are multiple ways to manually specify the ca bundle within the python code. Depending on the package needed this will vary. In the following example, no additional packages are installed:

```
import certifi

import ssl

from urllib.request import urlopen

ctx = ssl.create_default_context(ssl.Purpose.SERVER_AUTH) ctx.load_
verify_locations(cafile=certifi.where()) # modify to your location

response = urlopen(“https://httpbin.org”, context=ctx)When
distributing the software, include the ca-bundle with the python
files and point cafile to a relative location in relation to the
python file.
```

3.3.8 Python PIP / Conda

This section describes the SSL configuration for PIP command.

Pip		
Tested Version	**Documentation**	**Notes**
pip 24.0 with python 3.14	https://pip.pypa.io/en/latest/topics/https-certificates/	By default, pip will not use the system certificate store but, instead, uses a bundled CA certificate store from certifi. Version 22.2+ use system trust store with --use-feature=truststore. This will replace the bundled packaged certificates for verifying HTTPS certificates. Requires Python 3.10+.
Configuration Details		
Command Line Argument		
`python -m pip install SomePackage --use-feature=truststore`		

3.3.9 Python urlib3 library

This section describes the SSL configuration for the urllib3 python library and packages.

Tested software version: 2.2.1

Official Documentation

Link: https://urllib3.readthedocs.io/en/2.2.1/user-guide.html#certificate-verification

If certificate validation is needed, starting on version 1.25 you can use the PoolManager configuration option ca_certs

```
import certifi
import urllib3
http = urllib3.PoolManager(
cert_reqs='CERT_REQUIRED',
ca_certs=certifi.where() # Specify your location
)
```

3.3.10 Python requests library

This section describes the SSL configuration for the request python library. Requests verifies SSL certificates for HTTPS requests, just like a web browser. By default, SSL verification is enabled, and Requests will throw an SSLError if it's unable to verify the certificate.

There are two main methods to enable a certificate bundle for requests when making the request: using the verify option, or via environment variables.

```
import requests
cafile = 'cacert.pem' # adjust to your location
r = requests.get(url, verify=cafile)
```

Python Requests		
Tested Version	**Documentation**	**Notes**
Python 3.14 with Requests 2.32.3	https://requests.readthedocs.io/en/latest/	This list of trusted CAs can also be specified through the REQUESTS_CA_BUNDLE environment variable. If REQUESTS_CA_BUNDLE is not set, CURL_CA_BUNDLE will be used as fallback.

Configuration Details

Environment Variable Windows

```
[System.Environment]::SetEnvironmentVariable("REQUESTS_CA_BUNDLE", "<Path to Certificate>\ca-bundle.pem", "Machine")
```

Environment Variable Linux

```
echo "export REQUESTS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem" >> $HOME/.bashrc
```

Environment Variable Mac

```
echo "export REQUESTS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem"
```

3.3.11 Python venv

A Python virtual environment (venv) is a self-contained directory that allows you to manage dependencies for different projects separately, preventing conflicts between package versions. It creates an isolated environment where you can install packages without affecting the global Python installation.

We have chosen the simplest form to insert environment variables into python virtual environments. There are more advanced methods like postactivate but these are not covered here.

1. Locate your activate based on your virtual environment. You can use the following command to list all environments in a Mac system:

   ```
   find $HOME -name "*activate" -type f
   ```

2. Select the environment that you want to modify and edit the activate command:

   ```
   nano PATH_TO_YOUR_ENV/bin/activate
   ```

3. Be sure to replace the correct path. Then, add the environment variables to the end of the file in the form:

   ```
   export KEY=VALUE
   ```

4. Based on the version of python of your virtual environment select the correct value for the KEY and use the bundle file path as the value, if there are spaces use double quotes, for example:

   ```
   export SSL_CERT_FILE="/User/Private/Zscaler Bundle/ca-bundle.pem"
   ```

5. Save the file and exit.

Other methods can be found in this Stack Overflow Post.

3.3.12 Xcode Simulator

The following steps need to be taken in order to install Zscaler Root Certificate to Xcode simulator.

Xcode Simulator		
Tested Version	**Documentation**	**Notes**
xcrun version 68. Xcode 11.4	https://support.apple.com/en-us/HT204132 https://developer.apple.com/documentation/xcode/installing-additional-simulator-runtimes	Use this guide to troubleshoot connection problems: Identifying the Source of Blocked Connections
Configuration Details		
Command Line Argument		

```
xcrun simctl keychain booted add-root-cert <Zscaler Root Certificate path in
pem format>
```

3.3.13 Android Emulator

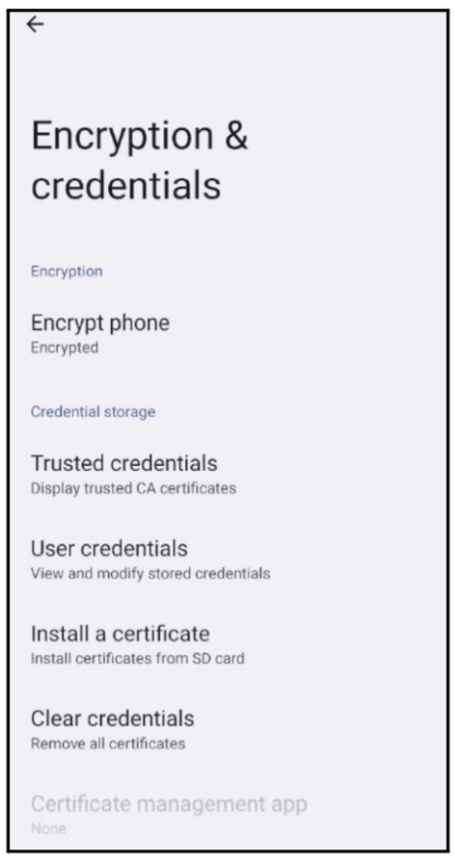
The following steps were tested in Koala (Build #AI-241.18034.62.2411.12071903, built on July 10, 2024), in order to install Zscaler Root Certificate to Android Emulator. This will allow Developers to test their applications when running emulated. But keep in mind the following:

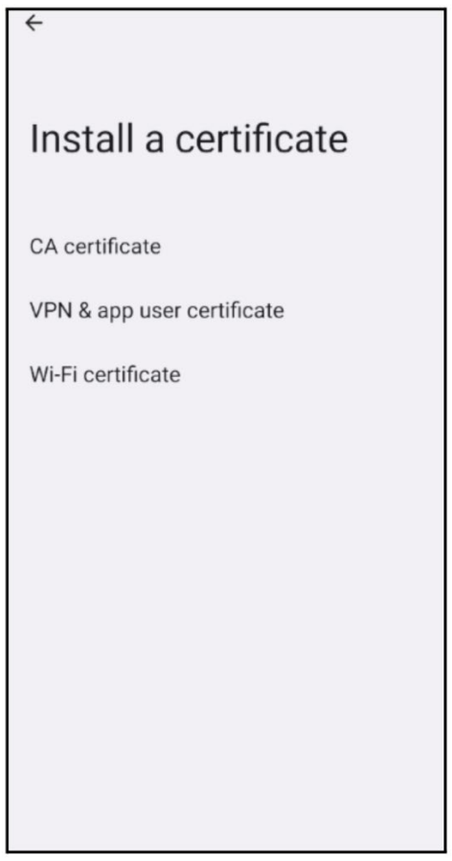
- The network_security_config.xml file needs to be set up so that the applications trust User installed certificates.

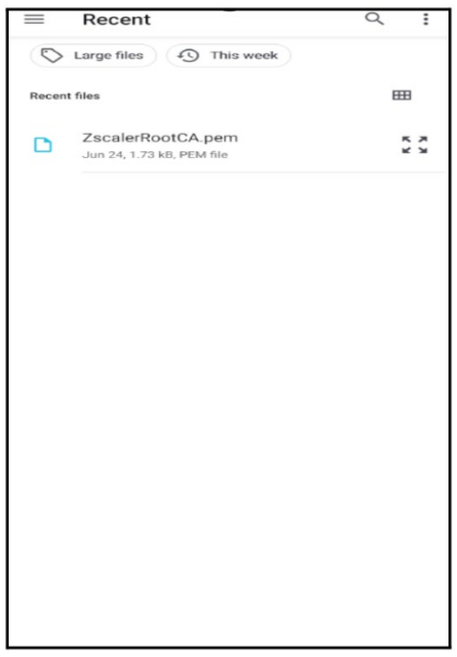
- Starting with version 7 of Android, directly adding certificates to the system trust store on unrooted devices is no longer possible through standard methods.

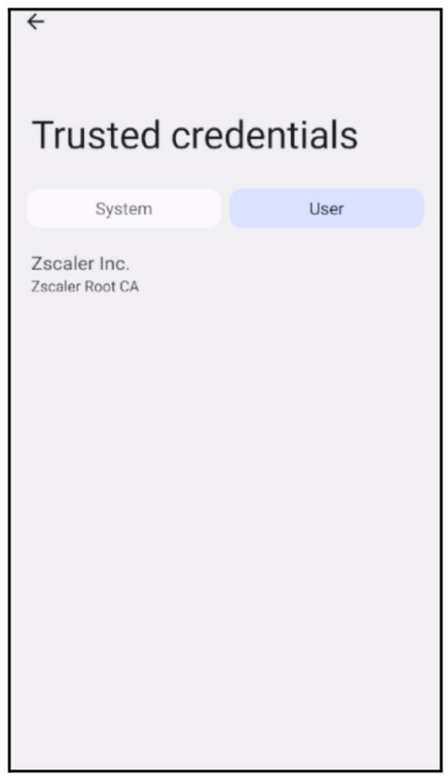
To install the certificate, open the CER file and clean the additional information that it may contain. Rename it to the PEM file format. You can drag and drop the Zscaler Root Certificate PEM onto the Emulated Android and it will be stored in the files. Then install the certificate by following these steps:

Step 1	Step 2	Step 3	Step 4
Open Settings > Security > More Security Settings > Encryption & Credentials.	Click on Install a Certificate > CA Certificate to install Zscaler Root Certificate from files.	Select the certifi–cate from files. On the warning, click on “Install Anyway”.	See the notification “CA certificate installed” at the bottom of the emulator screen. The Zscaler Root certificate should be installed. Verify the certificate is installed under Settings > Security > More Se–curity Settings > En–cryption & Credentials > Trusted Credentials > User.









Step 5

Starting in version 7 of Android, you must also include the following entry in your applica–tion’s network_ security_config.xml configuration file.

This settings will allow the running application within the Emulated Android to trust Zscaler’s root certificate.

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

3.3.14 APT

The command-line tool `apt-get` is the most popular package management tool used in our Debian-based Linux operating system. Starting on apt 1.5 HTTPS is natively supported, yet most sources are configured on HTTP by default.

The following procedure was tested on Ubuntu 20.04 LTS. Official documentation on update-ca-certificates can be found here.

If you encounter an error like this:

```
Err:5 https://us-west-1.ec2.archive.ubuntu.com/ubuntu focal Release
Certificate verification failed: The certificate is NOT trusted. The
certificate issuer is unknown. Could not handshake: Error in the
certificate verification. [IP: 54.241.183.81 443]
```

Depending on the distribution and version of the package, ca-certificates will enable a centralized certificate store for the Linux distribution located at /etc/ssl/certs. The package comes with CLI commands to interact with the store.

1. Make a folder to contain the Zscaler root certificate:

   ```
   mkdir /usr/share/ca-certificates/zscaler
   ```

2. Then copy the certificate as an individual file within that folder and use the extension crt.

3. Then modify the ca-certificates configuration file to include the certificate. At the last line include:

   ```
   zscaler/Zscaler_root.crt
   ```

4. Then issue the command:

   ```
   update-ca-certificates
   ```

5. The output of the command will look like this:

   ```
   ubuntu@ip-192-168-99-133:/etc/ssl/certs$
   sudo update-ca-certificates
   Updating certificates in /etc/ssl/certs...
   rehash: warning: skipping ca-certificates.crt,it does not contain
   exactly one certificate or CRL
   1 added, 0 removed; done.
   Running hooks in /etc/ca-certificates/update.d...
   done.
   ```

This update CA store will enable ZIA for more applications than just APT.

3.3.15 YUM

YUM is a software package management utility used in many popular Linux distributions, including Fedora, CentOS and Amazon Linux 2.

The following procedure was tested in Amazon Linux 2.0.20240719.0

If you encounter an error like this:

```
[ec2-user@ip-192-168-99-237 tls]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Could not retrieve mirrorlist https://amazonlinux-2-repos-us-west-1.
s3.dualstack.us-west-1.amazonaws.com/2/core/latest/x86_64/mirror.list
error was
14: curl#60 - “SSL certificate problem: unable to get local
issuer certificate”
```

1. Place the Zscaler root certificate in this location:

   ```
   /etc/pki/ca-trust/source/anchors/zscaler.pem
   ```

2. Run the following command:

   ```
   sudo update-ca-trust extract
   ```

3. You can now update your system via YUM with the following command:

   ```
   sudo yum update
   ```

3.3.16 Cursor

There are two aspects of Cursor that should be configured. In Cursor IDE the http/2 configuration should be disabled with the following checkbox (Settings->General):

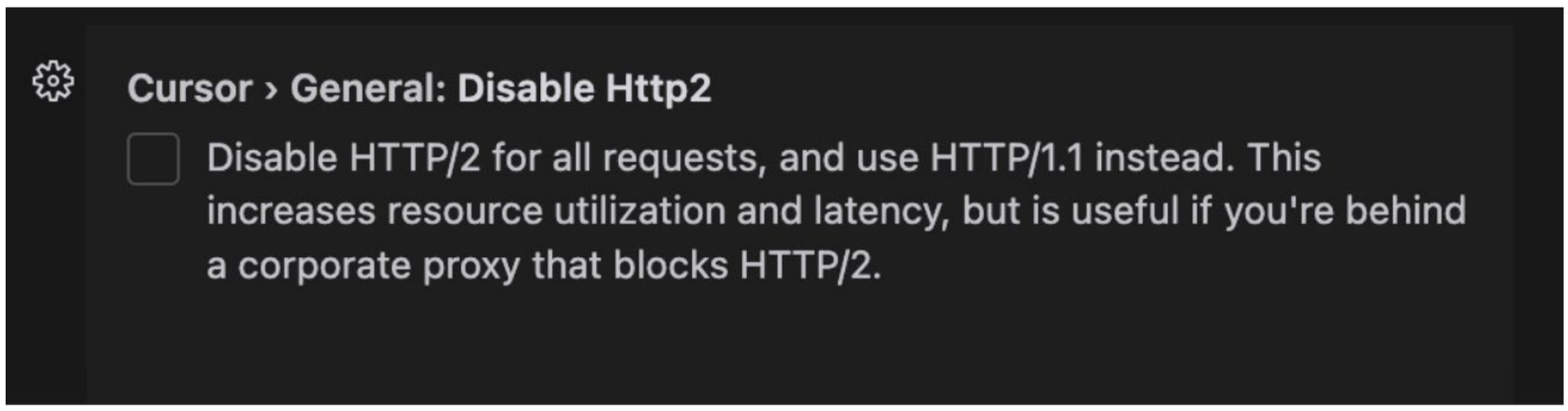


Figure 10: Disable Http2 on Cursor

If you still have problems and need to force Cursor to use the System Certificate store, navigate to the following file:

~/Library/Application\ Support/Cursor/User/settings.json

Inside the file you need to include the following 3 settings making sure the JSON format is not broken and not removing any other settings:

```
JSON
"http.systemCertificates": true,

"http.experimental.systemCertificatesv2": true,

"cursor.general.disableHttp2": true
```

The second aspect is that Cursor has published a list of applications endpoints that should be identified and explicitly allowed:

Usage	Endpoint	Recommended Action
Used for most API requests.	api2.cursor.sh	Inspect
Used for Cursor Tab requests (http/2 Only).	api3.cursor.sh	Inspect/Block
Used for codebase indexing.	repo42.cursor.sh	Inspect
Used for Cursor Tab requests location based (http/2 only).	api4.cursor.sh us-asia.gcpp.cursor.sh us-eu.gcpp.cursor.sh us-only.gcpp.cursor.sh	Inspect / Block (block undesired regions)
Used for downloading extensions from extension marketplace.	marketplace.cursorapi.com cursor-cdn.com	Inspect
Used for checking for and downloading updates.	download.todesktop.com	Inspect

3.3.17 VSCode

If you are using VSCode with AI Code Assistants you need to enable the experimental feature:

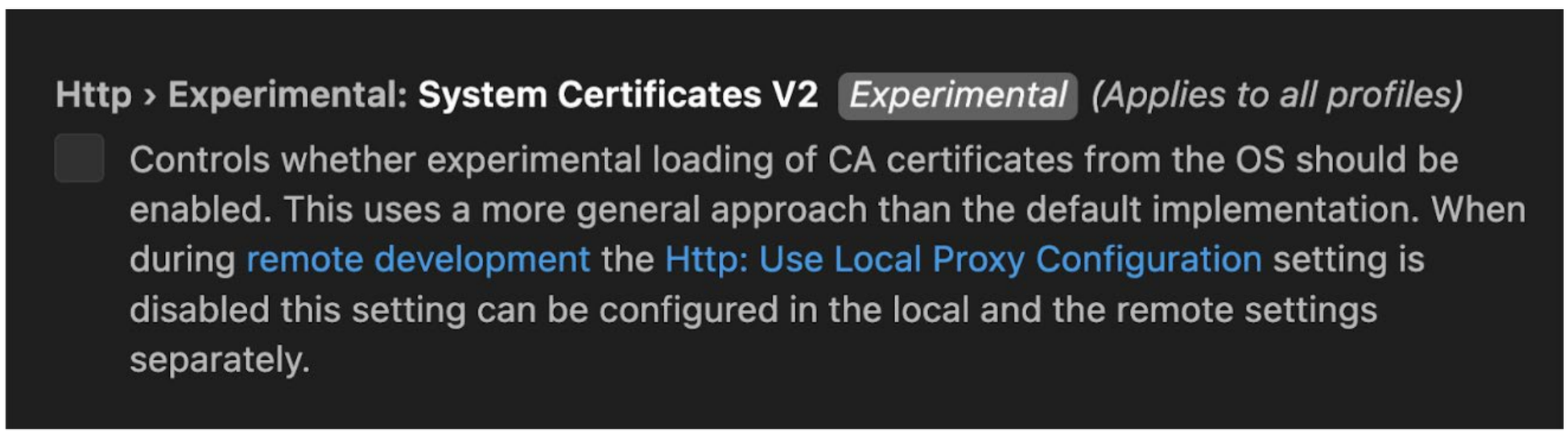


Figure 11: Enable Experimental Feature on VSCode

Troubleshooting Connection Problems

This section describes ways to diagnose connection problems when ZIA is activated on the Zscaler Client Connector. Connection problems with individual applications often manifest as applications that do not function correctly, no longer respond when starting, or do not start at all during execution. The problems only occur if the network traffic is routed via ZIA.

For the ZIA Administrator, the easiest way to diagnose connection problems is via the Analytics function in the ZIA Admin Portal. To check the logs in the Admin Portal, navigate to Analytics → Web Insights → Logs, Analytics → Firewall Insights → Logs and Analytics → DNS Insights → Logs. Filter the log file using the username and timestamp of the blocked traffic and look for “not allowed” connections.

If a diagnosis via the ZIA Admin Portal is not possible, an analysis can also be carried out via the Zscaler Client Connector or via network diagnosis tools such as Wireshark. The procedure for Mac OSX and Windows is described in the following sections.

As of version 4.3 you can find local logs for the Zscaler Client Connector in the following locations:

OS	Log Location	Details
macOS	/Library/Application Support/Zscaler/	Common Logs
	~/Library/Application Support/com.zscaler.zscaler/	User specific logs
Windows	C:\ProgramData\Zscaler %ALLUSERSPROFILE%\Zscaler	**Common Logs** ZSAService ZSAUpdater ZSATunnel (MT)
	C:\ProgramData\Zscaler\log-[User SID] %ALLUSERSPROFILE%\Zscaler\log-[User-SID]	**User Specific Logs** ZSATrayManager ZSATrayHelper ZSATunnel (User) ZSAUpm DBs
	C:\Users\[User Folder]\AppData\Local\Zscaler %LOCALAPPDATA%\Zscaler	**User AppData Logs** ZSATray ZSATrayHelper

4.1 Mac OS X

The application that is having trouble could be grouped in applications that open and don't show the correct or incomplete content, or applications that fail to open or quit unexpectedly. Regardless, use the Zscaler Client Connector Packet Capture with ZIA disabled as a comparison when the application works as expected.

As an example, look at the Apple App Store. This application has certificate pinning, so it will not work with default settings with ZIA. In fact, **Figure 12** shows what the app will look like when opened.

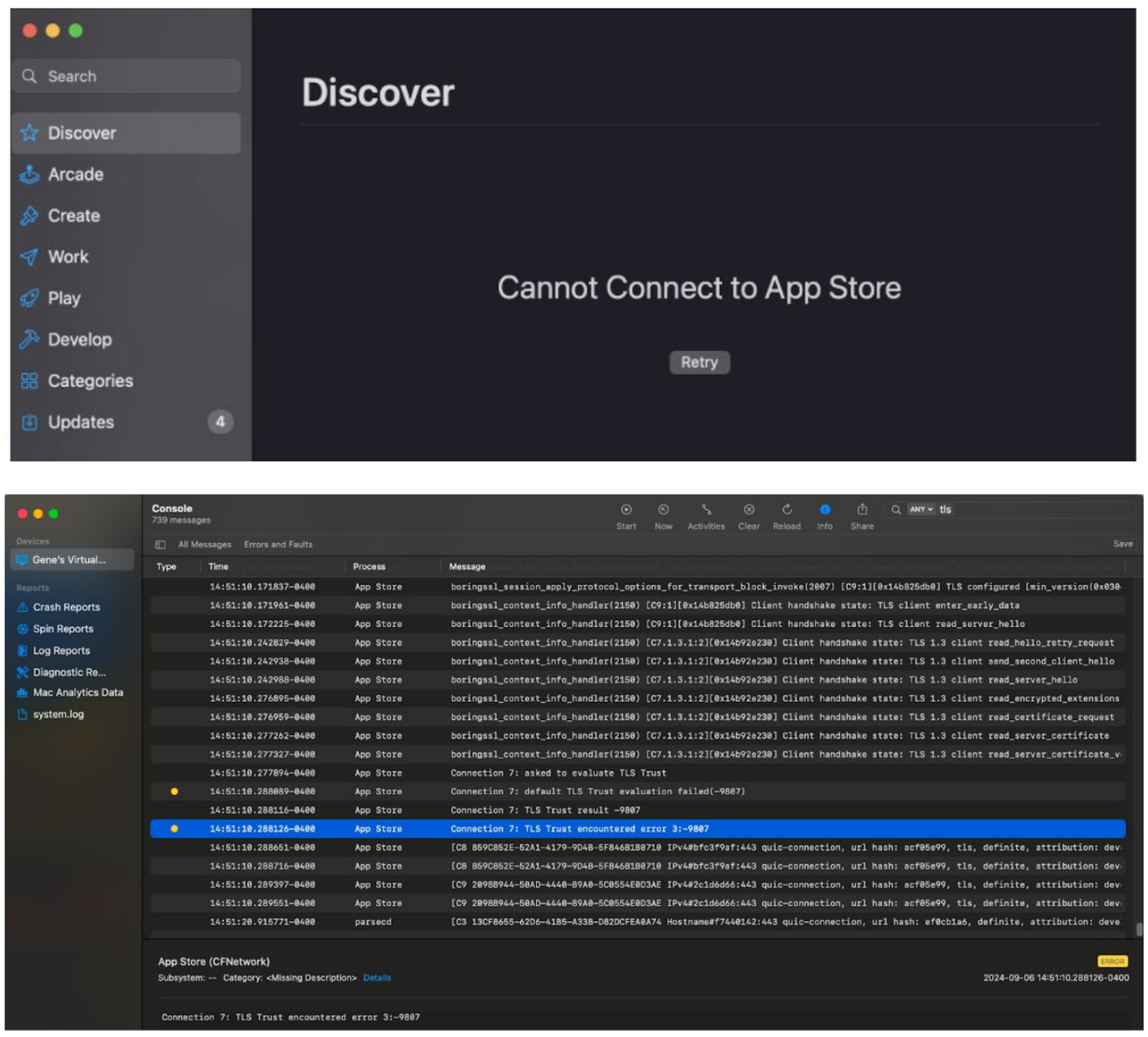
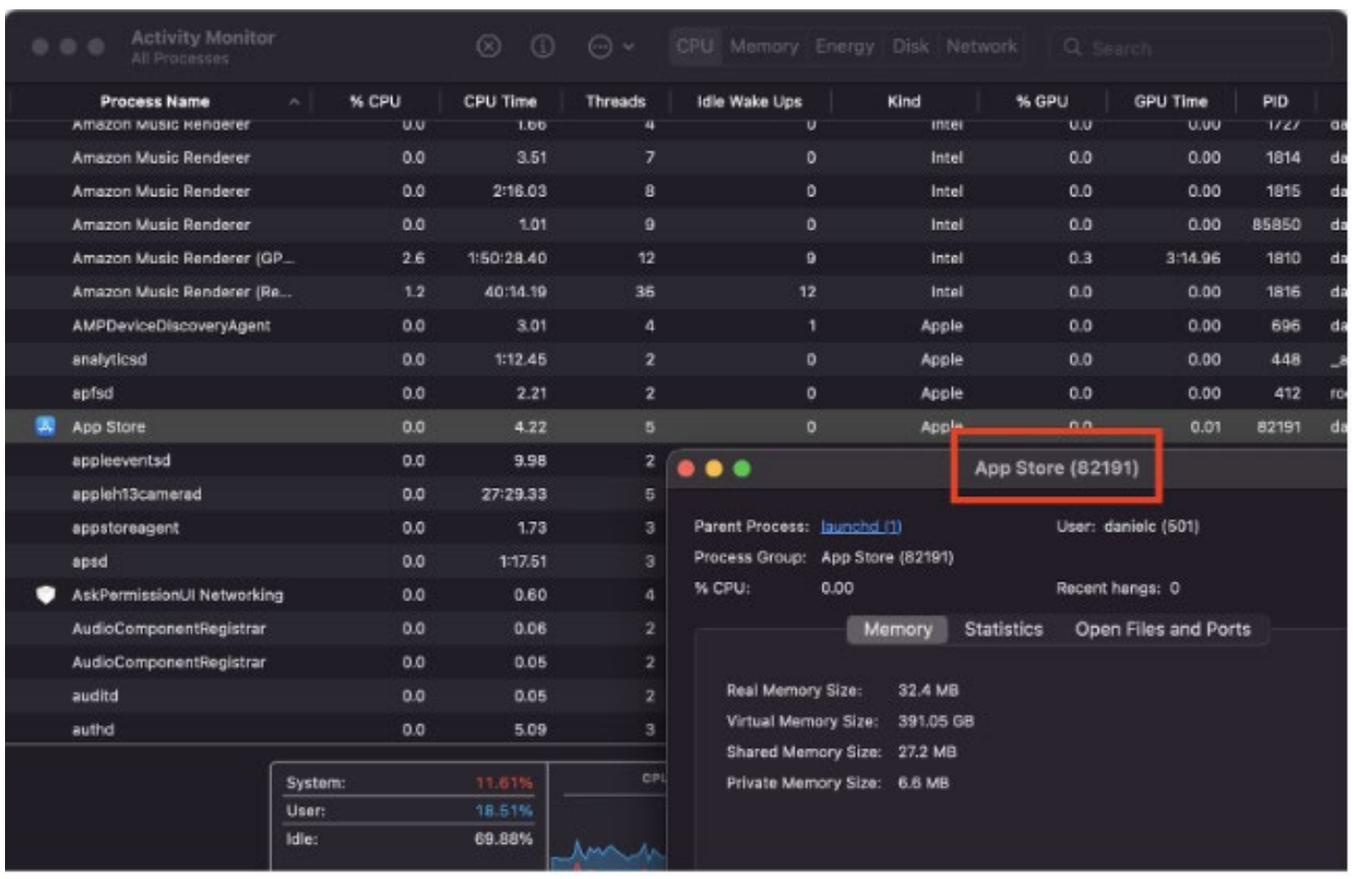
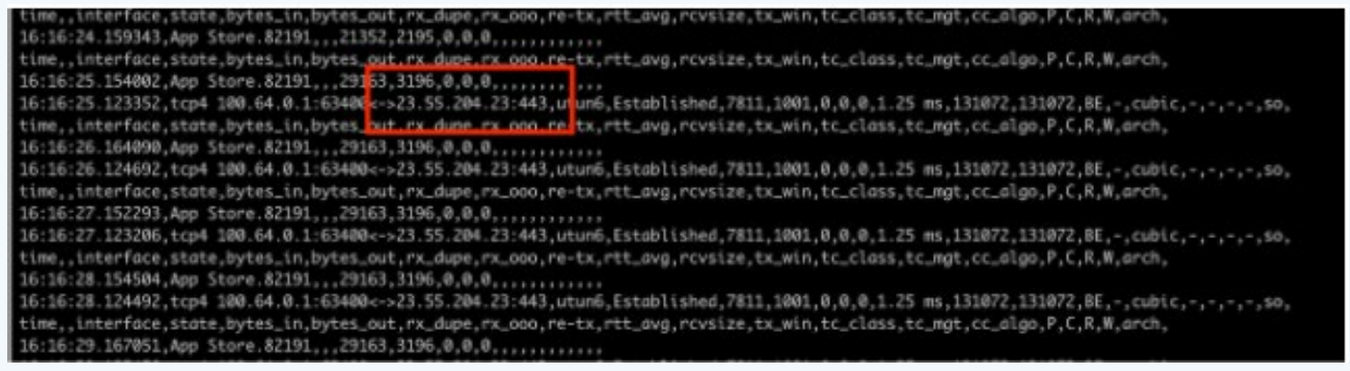
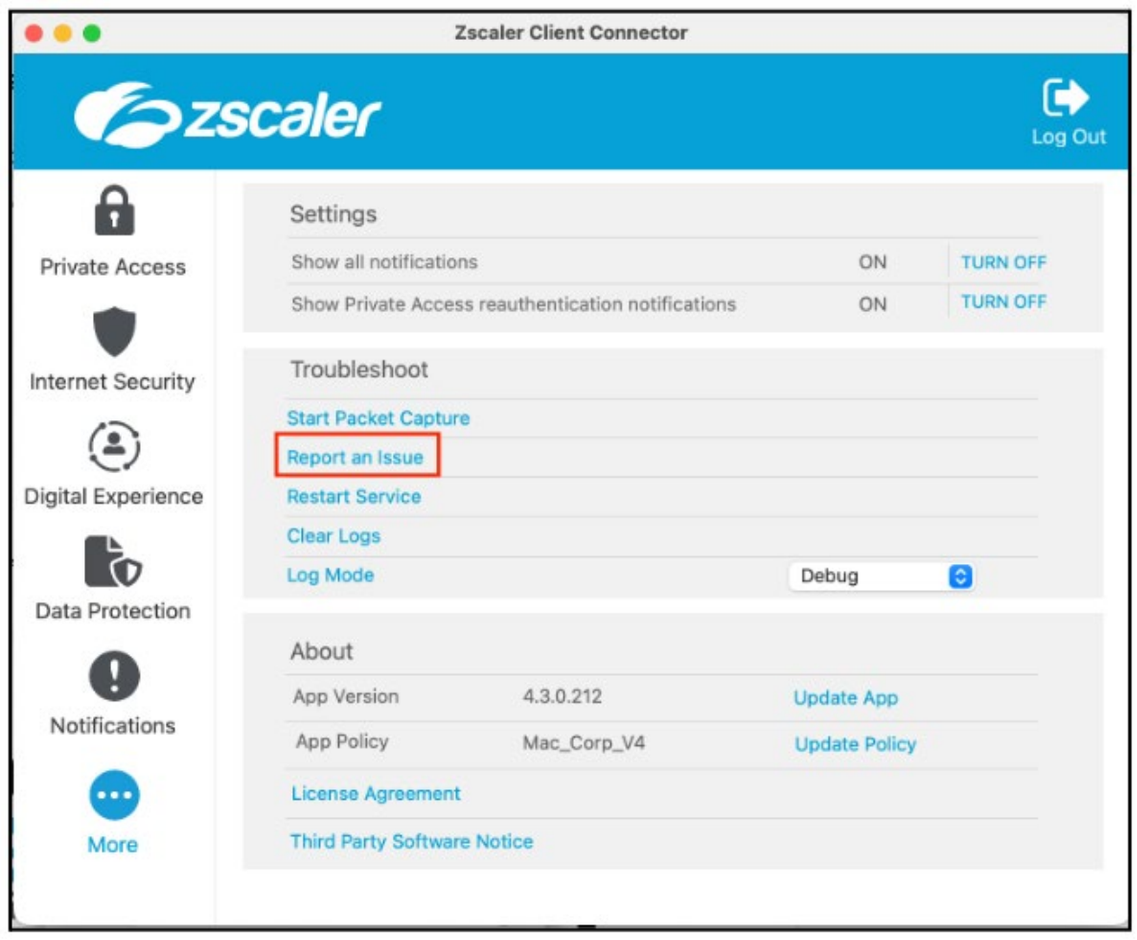
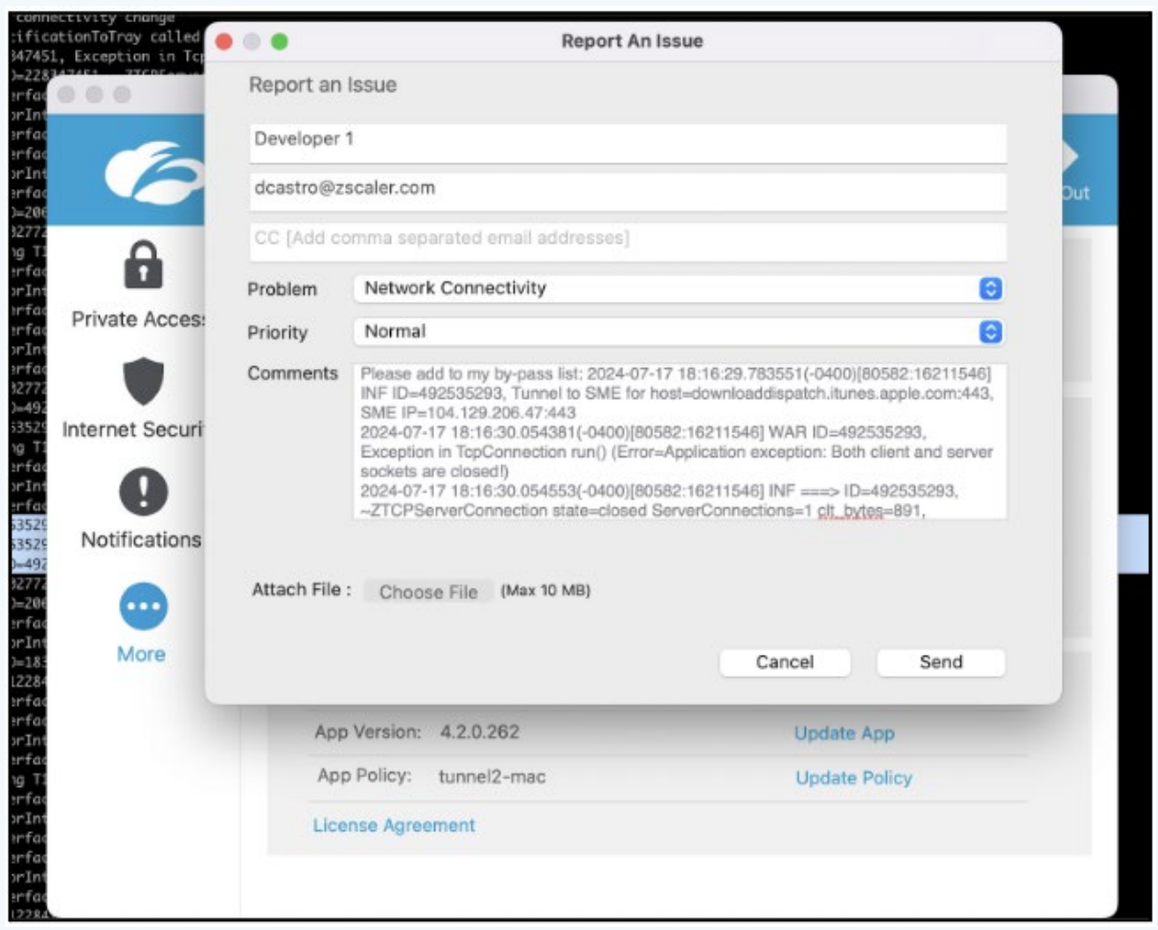


Figure 12: Troubleshooting Apple App Store

The easiest way to troubleshoot an application in Mac is to use several built-in tools and the ability of the Zscaler Client Connector to do a Packet Capture. Follow the steps outlined below.

Step	Screenshot Reference
1. If the application opens but does not work, use “Application Monitor” to get the PID of the application.	
2. With the PID in hand we can now use nettop to track the connection done by the application. Use the following command: **nettop -L 0 -p <PID>** Add a “-n” if you want to try to see IP addresses in-stead of names. This will open a log file on screen that shows any TCP connections that the ap-plication tries. Most often, you will be able to see the names or destination IP address. In this case, the IP address is 23.55.204.23. A reverse lookup for the IP points to akamaitechnol-ogies.com, a known CDN provider.	

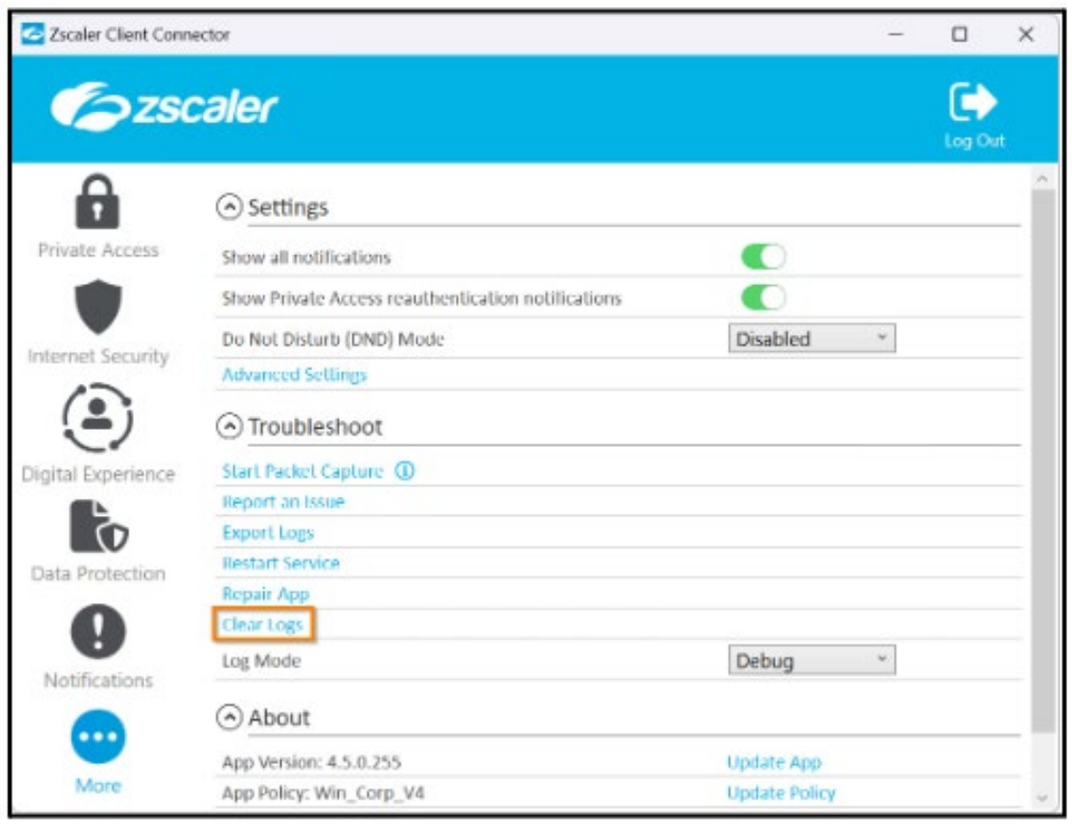
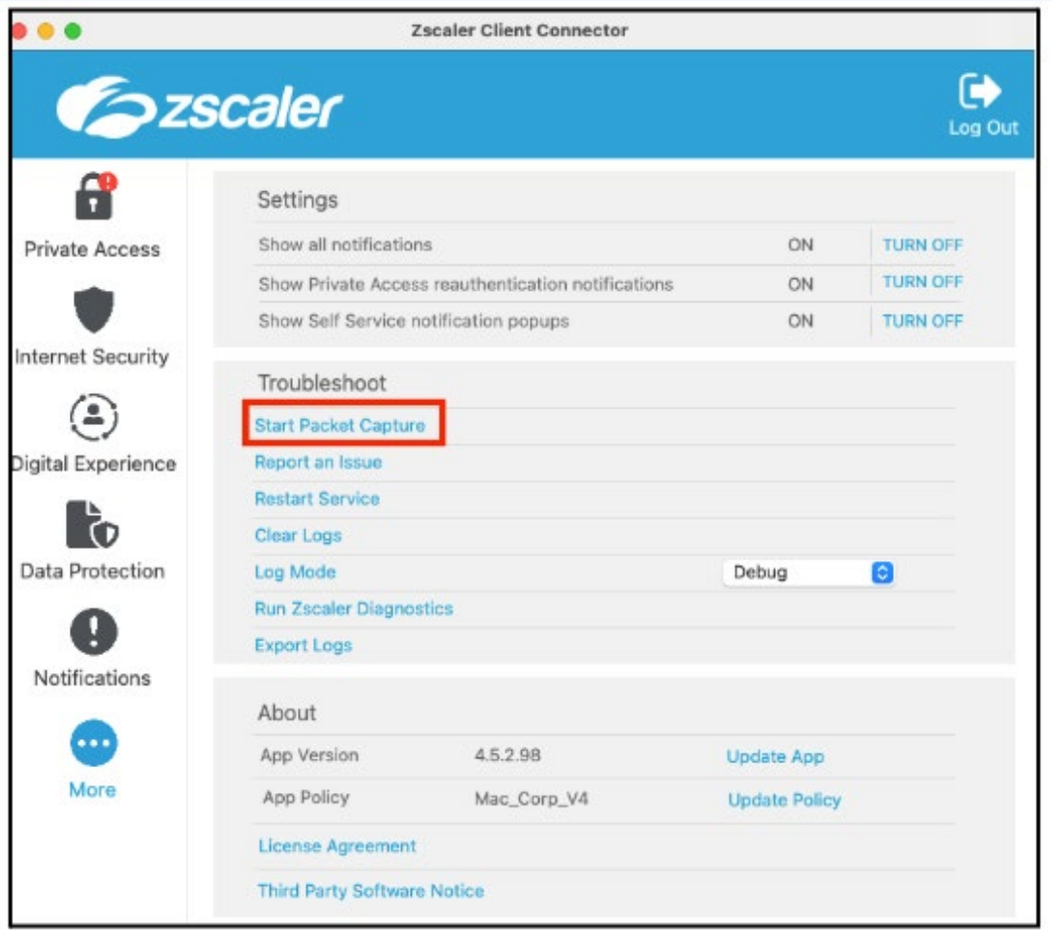
Step	Screenshot Reference
3. Refer to the Zscaler Client Connector logs. In this case, it corroborates that indeed there is a connection problem with the application.	`2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to SME for host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443` `2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception in TcpConnection run() (Error=Application exception: Both client and server sockets are closed!`
4. Communicate with the ZIA Administrator or follow the procedure described in the section Feedback from User to Administrator to send a message to the administrators using the Zscaler Client Connector. Report the issue using the Zscaler Client Connector More option.	
4a. In the form, specify that there was a Network Connectivity issue and include a copy of the Zscaler Client Connector log entry that shows the URL for the connection. With that information, the Administrator can investigate the issue on the ZIA logs.	

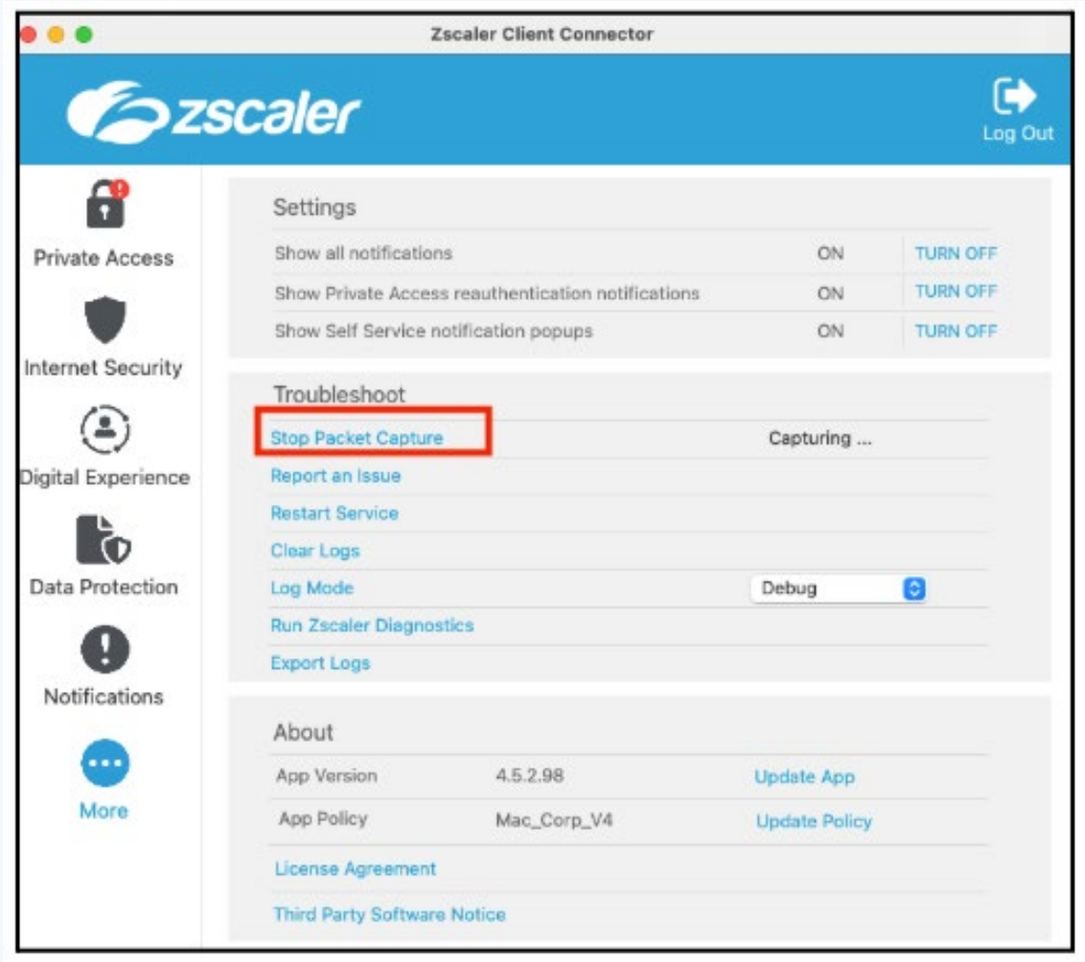
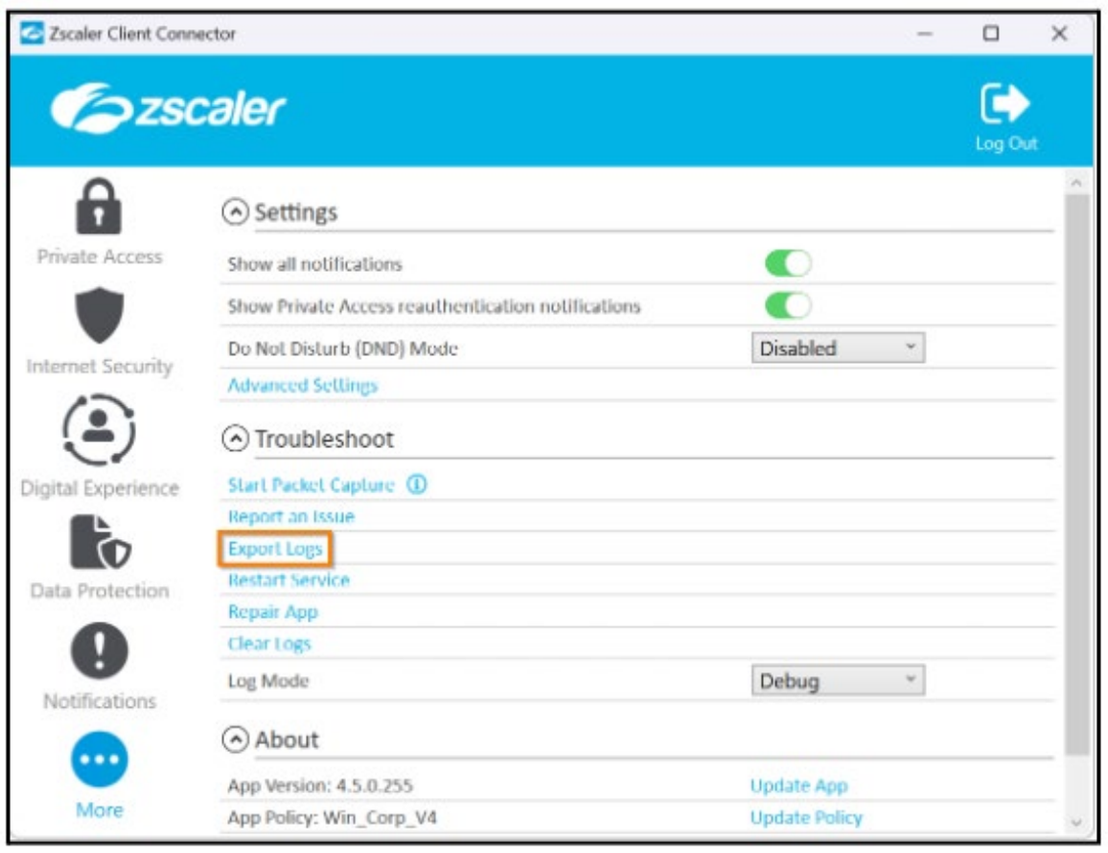
The developer has now provided enough information for the Administrator to act on this and possibly resolve the case.

Note: Appendix: Wireshark Troubleshooting Example contains examples of how the developer can use Wireshark and other utilities to capture the traffic and see the DNS queries and possibly the TCP reset on the connection. In Appendix: Wireshark Troubleshooting TLS Example we present an alternate troubleshooting method that focuses on the TLS handshake.

4.2 Windows

The Zscaler Client Connector is well suited for diagnostics on Windows systems, as it automatically creates the necessary logs and can also record data traffic for analysis in Wireshark if required. Follow the steps below to troubleshoot on Windows devices.

Step	Screenshot Reference
1. On the Client Connector, click the Clear Logs option. This is optional and recommended to keep the size of the logs to be analyzed small before reproducing the problem.	
2. Select the Start Packet Capture option to obtain the Wireshark logs as well as the logs.	

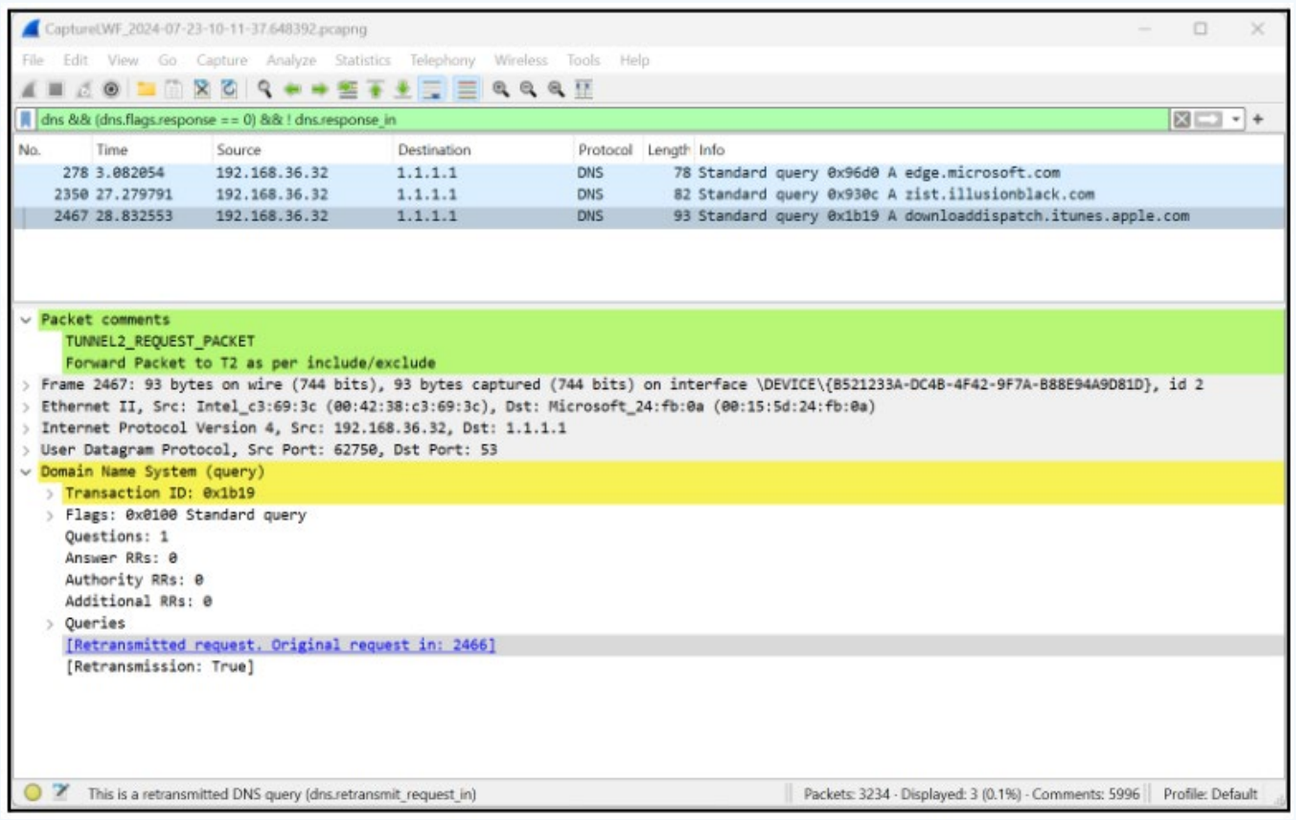
Step	Screenshot Reference
3. Reproduce the problem. Click **Stop Packet Capture** when the recording of the network traffic is com-plete.	
4. Export the logs in the form of a ZIP file via the Export Logs option in the Zscaler Client Connector.	

5. Analyze the logs.

   5a. The ZIP file contains many logs. For the analysis of non-permitted host names and IP addresses, the logs beginning with the designation **ZSATunnel*** and the explicitly created Wireshark logs with the designation **CaptureLWF*** are of interest.

   5b. To select possible problem candidates, open the network record with Wireshark and search for unanswered DNS queries. If the Zscaler Zero Trust Exchange blocks a connection at DNS level, this is shown in the logs by the fact that a DNS query is sent but no response is returned. The following Wireshark filter detects such requests:

```
dns && (dns.flags.response == 0) && ! dns.response_in
```

Step	Screenshot Reference
The following queries are displayed as a result:	

There are other reasons why a DNS request may not be answered, but the three hostnames listed – edge.microsoft.com, zist.illusionblack.com and downloaddispatch.itunes.apple.com – are candidates for further investigation.

Appendix

5.1 Install CA bundle Bash Script

Shell

```
#!/bin/zsh
set -x
echo $0
echo $1
echo $2
echo $3
echo $4

CERT_DIR="/Users/Shared"
WORK_DIR="/Users/Shared"
PLIST_LABEL="com.local.zscalerBundleUpdate"
PLIST_PATH="/Library/LaunchDaemons/${PLIST_LABEL}.plist"
UPDATE_BUNDLE_SCRIPT_PATH="/Users/Shared/zscaler_bundle_update.sh"
LOG_FILE="/var/log/zscaler_bundle_update.log"
ZSCALER_CERT="${CERT_DIR}/Zscaler_root.crt"
FIREFOX_BUNDLE="${CERT_DIR}/ca-bundle.crt"
FIREFOX_BUNDLE_URL="https://curl.se/ca/cacert.pem"

zscaler_ca(){
        ############################## Section 1
#################################################################

#################################################################################################
##################
    ##### This section creates the Zscaler Root Certificate file in the system

#################################################################################################
##################
    echo "-----BEGIN CERTIFICATE-----" > ${ZSCALER_CERT}
    echo "MIIE0zCCA7ugAwIBAgIJANu+mC2Jt3uTMA0GCSqGSIb3DQEBCwUAMIGhMQswCQYD" >>
${ZSCALER_CERT}
    echo "VQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2Ux" >>
${ZSCALER_CERT}
    echo "FTATBgNVBAoTDFpzY2FsZXIgSW5jLjEVMBMGA1UECxMMWnNjYWxlciBJbmMuMRgw" >>
${ZSCALER_CERT}
```

```
    echo "FgYDVQQDEw9ac2NhbGVyIFJvb3QgQ0ExIjAgBgkqhkiG9w0BCQEWE3N1cHBvcnRA" >>
${ZSCALER_CERT}
    echo "enNjYWxlci5jb20wHhcNMTQxMjE5MDAyNzU1WhcNNDIwNTA2MDAyNzU1WjCBoTEL" >>
${ZSCALER_CERT}
    echo "MAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBK" >>
${ZSCALER_CERT}
    echo "b3NlMRUwEwYDVQQKEwxac2NhbGVyIEluYy4xFTATBgNVBAsTDFpzY2FsZXIgSW5j" >>
${ZSCALER_CERT}
    echo "LjEYMBYGA1UEAxMPWnNjYWxlciBSb290IENBMSIwIAYJKoZIhvcNAQkBFhNzdXBw" >>
${ZSCALER_CERT}
    echo "b3J0QHpzY2FsZXIuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA" >>
${ZSCALER_CERT}
    echo "qT7STSxZRTgEFFf6doHajSc1vk5jmzmM6BWuOo044EsaTc9eVEV/HjH/1DWzZtcr" >>
${ZSCALER_CERT}
    echo "fTj+ni205apMTlKBW3UYR+lyLHQ9FoZiDXYXK8poKSV5+Tm0Vls/5Kb8mkhVVqv7" >>
${ZSCALER_CERT}
    echo "LgYEmvEY7HPY+i1nEGZCa46ZXCOohJ0mBEtB9JVlpDIO+nN0hUMAYYdZ1KZWCMNf" >>
${ZSCALER_CERT}
    echo "5J/aTZiShsorN2A38iSOhdd+mcRM4iNL3gsLu99XhKnRqKoHeH83lVdfu1XBeoQz" >>
${ZSCALER_CERT}
    echo "z5V6gA3kbRvhDwoIlTBeMa5l4yRdJAfdpkbFzqiwSgNdhbxTHnYYorDzKfr2rEFM" >>
${ZSCALER_CERT}
    echo "dsMU0DHdeAZf711+1CunuQIDAQABo4IBCjCCAQYwHQYDVR0OBBYEFLm33UrNww4M" >>
${ZSCALER_CERT}
    echo "hp1d3+wcBGnFTpjfMIHWBgNVHSMEgc4wgcuAFLm33UrNww4Mhp1d3+wcBGnFTpjf" >>
${ZSCALER_CERT}
    echo "oYGnpIGkMIGhMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8G" >>
${ZSCALER_CERT}
    echo "A1UEBxMIU2FuIEpvc2UxFTATBgNVBAoTDFpzY2FsZXIgSW5jLjEVMBMGA1UECxMM" >>
${ZSCALER_CERT}
    echo "WnNjYWxlciBJbmMuMRgwFgYDVQQDEw9ac2NhbGVyIFJvb3QgQ0ExIjAgBgkqhkiG" >>
${ZSCALER_CERT}
    echo "9w0BCQEWE3N1cHBvcnRAenNjYWxlci5jb22CCQDbvpgtibd7kzAMBgNVHRMEBTAD" >>
${ZSCALER_CERT}
    echo "AQH/MA0GCSqGSIb3DQEBCwUAA4IBAQAw0NdJh8w3NsJu4KHuVZUrmZgIohnTm0j+" >>
${ZSCALER_CERT}
    echo "RTmYQ9IKA/pvxAcA6K1i/LO+Bt+tCX+C0yxqB8qzuo+4vAzoY5JEBhyhBhf1uK+P" >>
${ZSCALER_CERT}
    echo "/WVWFZN/+hTgpSbZgzUEnWQG2gOVd24msex+0Sr7hyr9vn6OueH+jj+vCMiAm5+u" >>
${ZSCALER_CERT}
```

```
    echo "kd7lLvJsBu3AO3jGWVLyPkS3i6Gf+rwAp1OsRrv3WnbkYcFf9xjuaf4z0hRCrLN2" >>
${ZSCALER_CERT}
    echo "xFNjavxrHmsH8jPHVvgc1VD0Opja0l/BRVauTrUaoW6tE+wFG5rEcPGS80jjHK4S" >>
${ZSCALER_CERT}
    echo "pB5iDj2mUZH1T8lzYtuZy0ZPirxmtsk3135+CKNa2OCAhhFjE0xd" >> ${ZSCALER_CERT}
    echo "-----END CERTIFICATE-----" >> ${ZSCALER_CERT}
}

run()
{
    # Check if running as root or sudoer
    if [ "$EUID" -ne 0 ]; then
        echo "Please run as root (use sudo)"
        exit 1
    fi


    ############################## Section 2
##################################################################

##################################################################################
##################
    ##### Make CA bundle file

##################################################################################
##################

    touch ${FIREFOX_BUNDLE}
    echo "" > ${FIREFOX_BUNDLE}
    curl --insecure --retry 3 --retry-delay 10 -o ${FIREFOX_BUNDLE}
${FIREFOX_BUNDLE_URL}

    if [ ! -f ${ZSCALER_CERT} ]; then
        zscaler_ca;
    fi
    echo " " >> ${FIREFOX_BUNDLE}
    echo "Zscaler Root Certificate" >> ${FIREFOX_BUNDLE}
    echo "========================" >> ${FIREFOX_BUNDLE}
    cat ${ZSCALER_CERT} >> ${FIREFOX_BUNDLE}
```

```
    # Add env variables using launchctl so they are persitant in Mac OSX even after
reboots, i.e.:
    # launchctl setenv NODE_EXTRA_CA_CERTS ${ZSCALER_CERT}
    # check if the tool exists in the system so not to create many env variables that
would be useless
    touch ${CERT_DIR}/env_vars.conf
    echo "" > ${CERT_DIR}/env_vars.conf

    touch ${CERT_DIR}/cmds.conf
    echo "" > ${CERT_DIR}/cmds.conf

    ############################## Section 3
############################################################

################################################################################
##################
    ##### Exports the generic system env variables
    ##### based on the existence of pre-determined software it stores an appropriate
env variable in a file

################################################################################
##################

    # Generic Env
    export SSL_CERT_FILE="${CERT_DIR}/ca-bundle.crt"
    export REQUESTS_CA_BUNDLE="${CERT_DIR}/ca-bundle.crt"
    launchctl setenv SSL_CERT_FILE ${CERT_DIR}/ca-bundle.crt
    echo "SSL_CERT_FILE = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf

    if [ -x "$(command -v gcloud)" ]; then
        gcloud config set core/custom_ca_certs_file ${CERT_DIR}/ca-bundle.crt
        echo "gcloud config set core/custom_ca_certs_file ${CERT_DIR}/ca-bundle.crt"
>> ${WORK_DIR}/cmds.conf
    fi
    if [ -x "$(command -v git)" ]; then
        launchctl setenv GIT_SSL_CAINFO ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv GIT_SSL_CAINFO ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "GIT_SSL_CAINFO = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v curl)" ]; then
```

```
        launchctl setenv CURL_CA_BUNDLE ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv CURL_CA_BUNDLE ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "CURL_CA_BUNDLE = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v aws)" ]; then
        launchctl setenv AWS_CA_BUNDLE ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv AWS_CA_BUNDLE ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "AWS_CA_BUNDLE = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v npm)" ]; then
        launchctl setenv NODE_EXTRA_CA_CERTS ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv NODE_EXTRA_CA_CERTS ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "NODE_EXTRA_CA_CERTS = ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v python)" ]; then
        launchctl setenv SSL_CERT_FILE ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv SSL_CERT_FILE ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "SSL_CERT_FILE = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v python3)" ]; then
        launchctl setenv SSL_CERT_FILE ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv SSL_CERT_FILE ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "SSL_CERT_FILE = ${CERT_DIR}/ca-bundle.crt" >> ${WORK_DIR}/env_vars.conf
    fi
    if [ -x "$(command -v xcrun)" ]; then
        xcrun simctl keychain booted add-root-cert ${CERT_DIR}/Zscaler_root.crt
        echo "xcrun simctl keychain booted add-root-cert ${CERT_DIR}/Zscaler_root.crt"
>> ${WORK_DIR}/cmds.conf
    fi
    if [ -x "$(command -v pip)" ]; then
        pip config set global.cert ${CERT_DIR}/ca-bundle.crt
        echo "pip config set global.cert ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
    fi
```

```
    if [ -x "$(command -v claude)" ]; then
        launchctl setenv NODE_EXTRA_CA_CERTS ${CERT_DIR}/ca-bundle.crt
        echo "launchctl setenv NODE_EXTRA_CA_CERTS ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/cmds.conf
        echo "NODE_EXTRA_CA_CERTS = ${CERT_DIR}/ca-bundle.crt" >>
${WORK_DIR}/env_vars.conf
    fi

    ############################## Section 4
##############################################################

##############################################################################################
###################
    ##### Mac specific configurations #
    ## As per
https://developer.apple.com/documentation/macos-release-notes/macos-big-sur-11_0_1-rel
ease-notes
    ## macOS Big Sur 11 beta improves system security by requiring an administrator
password when a certificate trust settings
    # change is made in the admin trust domain. Running as the root user alone is no
longer sufficient to modify certificate trust.
    # User trust domain settings continue to require confirmation by entering the
password for the user's account.
    # This change may affect you if one of the following is true:
    # You have written scripts which call /usr/bin/security add-trusted-cert -d ... as
root.
    # Your process runs as root and calls the SecTrustSettingsSetTrustSettings
function to trust a certificate.
    # Finally, If done via a MDM file this will work

##############################################################################################
####################

    os_ver=$(sw_vers -productVersion| awk -F. '{ print $1; }')
    if [[ $os_ver -ge 11 ]]; then
        echo "sudo security add-trusted-cert -d -r trustRoot -k
"/Library/Keychains/System.keychain" ${ZSCALER_CERT}" >> ${WORK_DIR}/cmds.conf
        security add-trusted-cert -d -r trustRoot -k
"/Library/Keychains/System.keychain" ${ZSCALER_CERT}
    fi
    if [[ $os_ver -lt 11 ]]; then
```

```
        echo "sudo security add-trusted-cert -d -r trustAsRoot -k
"/Library/Keychains/System.keychain" ${ZSCALER_CERT}" >> ${WORK_DIR}/cmds.conf
        security add-trusted-cert -d -r trustAsRoot -k
"/Library/Keychains/System.keychain" ${ZSCALER_CERT}
    fi


    # From here, is intended to be run from an MDM

    # Check if config file exists
    if [ ! -f "${WORK_DIR}/env_vars.conf" ]; then
        echo "Error: env_vars.conf not found"
        exit 1
    fi

    # Check if config file exists
    if [ ! -f "${WORK_DIR}/cmds.conf" ]; then
        echo "Error: cmds.conf not found"
        exit 1
    fi

    ############################## Section 5
############################################################

################################################################################
###################
    ##### creates the environment variable list with path value that were detected
    ##### Iterates over all users and installs the environment variable list

################################################################################
###################

    # Read and set each variable from the config file
    while IFS='=' read -r VAR_NAME VAR_VALUE || [ -n "$VAR_NAME" ]; do
        # Skip comments and empty lines
        [[ $VAR_NAME =~ ^#.*$ ]] && continue
        [[ -z $VAR_NAME ]] && continue

        # Remove any leading/trailing whitespace
        VAR_NAME=$(echo "$VAR_NAME" | xargs)
        VAR_VALUE=$(echo "$VAR_VALUE" | xargs)
```

```
# Add to system-wide profile
echo "$VAR_NAME=$VAR_VALUE" >> /etc/profile

# Add to system-wide bash profile
echo "$VAR_NAME=$VAR_VALUE" >> /etc/bashrc

# Add to system-wide zsh profile
echo "$VAR_NAME=$VAR_VALUE" >> /etc/zshrc

# Add to user profiles
for user_home in /Users/*; do
    if [ -d "$user_home" ]; then
        # Add to .bash_profile
        echo "export $VAR_NAME=$VAR_VALUE" >> "$user_home/.bash_profile"

        # Add to .zshrc
        echo "export $VAR_NAME=$VAR_VALUE" >> "$user_home/.zshrc"

        # Set proper permissions
        chown $(stat -f "%Su:%Sg" "$user_home") "$user_home/.bash_profile"
"$user_home/.zshrc"

        if [ -f "$user_home/Library/Application
Support/Cursor/User/settings.json" ]; then
            python3 -c "
import json
path = '$user_home/Library/Application Support/Cursor/User/settings.json'
with open(path) as f:
    data = json.load(f)
data['http.systemCertificates'] = True
data['http.experimental.systemCertificatesv2'] = True
data['cursor.general.disableHttp2'] = True
with open(path, 'w') as f:
    json.dump(data, f, indent=4)
"
        fi
    fi
done

echo "Set $VAR_NAME=$VAR_VALUE"
```

```
    done < "${WORK_DIR}/env_vars.conf"
    # Clean up of env_vars.conf so it does not run again
    rm "${WORK_DIR}/env_vars.conf"

    ############################## Section 6
##############################################################

##############################################################################################
##################
    ##### Iterate over all users in the system and modify the bash_profile and zshrc
files
    ##### with application specific instructions.

##############################################################################################
##################
    # Insert set commands in user profiles
    for user_home in /Users/*; do
        if [ -d "$user_home" ]; then
        while read p; do
                # Add to .bash_profile
                echo "$p" >> "$user_home/.bash_profile"

                # Add to .zshrc
                echo "$p" >> "$user_home/.zshrc"

                # Set proper permissions
                chown $(stat -f "%Su:%Sg" "$user_home") "$user_home/.bash_profile"
"$user_home/.zshrc"
            done < "${WORK_DIR}/cmds.conf"
        fi
    done
    rm "${WORK_DIR}/cmds.conf"

    # echo "All environment variables have been set system-wide"
    # echo "Please restart your terminal or run 'source ~/.zshrc' (or ~/.bash_profile)
to apply changes"

    exit 0
}

# ============================================================================
```

```
# MODE: --setup  (run once as root to install the launchd daemon)
# ================================================================================
run_setup() {
    echo "[*] Installing system launchd daemon..."
    echo "    Script path         : $UPDATE_BUNDLE_SCRIPT_PATH"
    echo "    Plist path          : $PLIST_PATH"
    echo "    Zscaler Cert path   : $ZSCALER_CERT"
    echo "    Bundle Path.        : $FIREFOX_BUNDLE"

    if [ ! -f ${ZSCALER_CERT} ]; then
        zscaler_ca;
    fi

    touch ${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "#!/bin/zsh" > ${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "set -x" >> ${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "rm -f ${FIREFOX_BUNDLE}" >> ${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "curl --insecure -o ${FIREFOX_BUNDLE} '${FIREFOX_BUNDLE_URL}'" >>
${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "echo "Zscaler Root CA" >> ${FIREFOX_BUNDLE_URL}" >>
${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "echo '======================' >> ${FIREFOX_BUNDLE_URL}" >>
${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "cat ${ZSCALER_CERT} >> ${FIREFOX_BUNDLE}" >> ${UPDATE_BUNDLE_SCRIPT_PATH}

    echo "Installing plist as cron but in launchd"

    cat > "$PLIST_PATH" << PLIST
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>

    <key>Label</key>
    <string>${PLIST_LABEL}</string>

    <!-- Calls this script with --run. No helper files needed. -->
    <key>ProgramArguments</key>
    <array>
        <string>/bin/bash</string>
```

```
    <string>${UPDATE_BUNDLE_SCRIPT_PATH}</string>
    <string>--only_cron</string>
</array>

<!-- Run on 1st of Jan, Apr, Jul, Oct at 08:00 (every ~3 months) -->
<key>StartCalendarInterval</key>
<array>
    <dict>
        <key>Month</key><integer>1</integer>
        <key>Day</key><integer>1</integer>
        <key>Hour</key><integer>8</integer>
        <key>Minute</key><integer>0</integer>
    </dict>
    <dict>
        <key>Month</key><integer>4</integer>
        <key>Day</key><integer>1</integer>
        <key>Hour</key><integer>8</integer>
        <key>Minute</key><integer>0</integer>
    </dict>
    <dict>
        <key>Month</key><integer>7</integer>
        <key>Day</key><integer>1</integer>
        <key>Hour</key><integer>8</integer>
        <key>Minute</key><integer>0</integer>
    </dict>
    <dict>
        <key>Month</key><integer>10</integer>
        <key>Day</key><integer>1</integer>
        <key>Hour</key><integer>8</integer>
        <key>Minute</key><integer>0</integer>
    </dict>
</array>

<key>EnvironmentVariables</key>
<dict>
    <key>PATH</key>
    <string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin</string>
</dict>

<key>StandardOutPath</key>
<string>${LOG_FILE}</string>
```

```
    <key>StandardErrorPath</key>
    <string>${LOG_FILE}</string>

    <key>RunAtLoad</key>
    <false/>

</dict>
</plist>
PLIST
    echo "[*] Plist written."
    chown root:wheel "$PLIST_PATH"
    chmod 644 "$PLIST_PATH"
    echo "[*] Plist written and permissions set."
    echo "    Script path : $SCRIPT_PATH"
    echo "    Plist path  : $PLIST_PATH"

    # Load the daemon — use bootstrap/bootout on macOS Ventura (13)+
    # and fall back to the legacy load/unload on older versions
    OS_MAJOR=$(sw_vers -productVersion | cut -d. -f1)
    if [[ "$OS_MAJOR" -ge 13 ]]; then
        launchctl bootout system "$PLIST_PATH" 2>/dev/null || true
        launchctl bootstrap system "$PLIST_PATH"
    else
        launchctl unload "$PLIST_PATH" 2>/dev/null || true
        launchctl load "$PLIST_PATH"
    fi
    echo "[*] Daemon loaded into launchd."

    echo ""
    echo "============================================================"
    echo " Setup complete!"
    echo "============================================================"
    echo " Script        : $SCRIPT_PATH"
    echo " Plist         : $PLIST_PATH"
    echo " Zscaler PEM   : $ZSCALER_CERT"
    echo " Output bundle : $FIREFOX_BUNDLE"
    echo " Log file      : $LOG_FILE"
    echo " Runs as       : root (LaunchDaemon, no login required)"
    echo " Schedule      : 1st of Jan, Apr, Jul, Oct at 08:00 AM"
    echo ""
    echo " Run the update manually now:"
```

```
    echo "   sudo launchctl kickstart system/$PLIST_LABEL"
    echo ""
    echo " Check daemon status:"
    echo "   sudo launchctl list | grep $PLIST_LABEL"
    echo "============================================================"
}

run_uninstall() {
    echo "[*] Uninstalling launchd daemon..."
    OS_MAJOR=$(sw_vers -productVersion | cut -d. -f1)
    if [[ "$OS_MAJOR" -ge 13 ]]; then
        launchctl bootout system "$PLIST_PATH" 2>/dev/null \
            && echo "    Daemon unloaded." || echo "    Daemon was not loaded."
    else
        launchctl unload "$PLIST_PATH" 2>/dev/null \
            && echo "    Daemon unloaded." || echo "    Daemon was not loaded."
    fi
    rm -f "$PLIST_PATH" && echo "    Plist removed: $PLIST_PATH"
    rm -f ${UPDATE_BUNDLE_SCRIPT_PATH}
    echo "[*] Done."
}

# ============================================================================
# Argument dispatcher
# ============================================================================
case "${4:-}" in
    --run_with_cron)
        run
        run_setup
        ;;
    --run)
        run
        ;;
    --only_cron)
        run_setup
        ;;
    --remove_cron)
        run_uninstall
        ;;
    *)
```

```
    ## If not running in JAMF change the case to $0, as $0, $1, $2 and $3 are used by
JAMF
        echo "Usage: $(basename "$4") [--run_with_cron | --run | --only_cron  |
--remove_cron]"
        echo ""
        echo "  --run_with_cron      Install launchd daemon and env vars for users"
        echo "  --run          Only install environment variables for all users    "
        echo "  --only_cron          Install the launchd daemon (run once as root) "
        echo "  --remove_cron           Remove the launchd daemon, script and plist"
        exit 1
        ;;
esac
```

5.2 Terraform Code Examples

Please check the order of any policies and any references in the code based on your deployment.

5.2.1 URL Category

None

```
resource "zia_url_categories" "AppleBypass" {
  super_category  = "USER_DEFINED"
  configured_name = "Bypass Key-pinning Apple"
  description     = "List of domains that apple devices do Certificate Pinning
preventing inspection"
  keywords        = ["developers", "apple"]
  custom_category = true
  type            = "URL_CATEGORY"
  urls = [
    "amp-api-edge.apps.apple.com", "amp-api.apps.apple.com", "api.apple-cloudkit.com",
    "app-site-association.cdn-apple.com", "appldnld.apple.com",
    "apps.mzstatic.com", "bag-cdn-lb.itunes-apple.com.akadns.net",
    "bag.itunes.apple.com", "xp.apple.com", "xp-cdn.apple.com",
    "configuration.apple.com", "dit.whatsapp.net",
    "downloaddispatch.itunes.apple.com", "updates.cdn-apple.com",
    "entitlements.itunes.apple.com", "gateway.icloud.com", "gdmf.apple.com",
    "gg.apple.com", "gs-loc.apple.com", "gs.apple.com", "gsa.apple.com",
    "h3.media.apple.map.fastly.net", "humb.apple.com", "identity.ess.apple.com",
    "ig.apple.com", "init.itunes.apple.com", "itunes.apple.com",
    "lcdn-locator.apple.com", "lcdn-registration.apple.com", "mesu.apple.com",
```

```
    "metrics.icloud.com", "mmg.whatsapp.net", "ns.itunes.apple.com",
    "oscdn.apple.com", "p63-acsegateway.icloud.com", "p63-fmip.icloud.com",
    "playgrounds-assets-cdn.apple.com", "ppq.apple.com", "profile.ess.apple.com",
    "securemetrics.apple.com", "serverstatus.apple.com", "setup.icloud.com",
    "skl.apple.com", "suconfig.apple.com", "swcdn.apple.com", "swdist.apple.com",
    "swscan.apple.com", "token.safebrowsing.apple", "updates-http.cdn-apple.com",
  ]
}

resource "zia_url_categories" "developerSafeSites" {
  super_category  = "USER_DEFINED"
  configured_name = "Developer Safe Sites"
  description     = "EXAMPLE List of domains developers are known to use and are
considered safe"
  keywords        = ["developers"]
  custom_category = true
  type            = "URL_CATEGORY"
  urls = [
    "github.com/microsoft/vcpkg.git", "github.com/composer",
    "codeload.github.com", "api.github.com/repos/Behat",
    "cursor.com", "api2.cursor.sh",
  ]
}

resource "zia_url_categories" "blockingCopilot" {
  super_category  = "USER_DEFINED"
  configured_name = "Developer are not allowed to use github copilot"
  description     = "EXAMPLE URLs to block github copilot"
  keywords        = ["developers"]
  custom_category = true
  type            = "URL_CATEGORY"
  urls = [
    "github.com/github-copilot",
    "github.com/features/copilot/"
  ]
}
```

5.2.2 DNS Control

None

```
resource "zia_firewall_dns_rule" "block_dns_req_https" {
  name        = "Block H/3 QUIC & ECH"
  description = "Block H/3 QUIC & ECH"
  action      = "BLOCK_WITH_RESPONSE"
  block_response_code = "SERVFAIL"
  state       = "ENABLED"
  order       = 7
  rank        = 7
  dns_rule_request_types = ["HTTPS"]
  protocols   = ["DOHTTPS_RULE"]
}
```

```
resource "zia_firewall_dns_rule" "block_dns_on_http"  {
  name        = "Block DoH"
  description = "Block DoH"
  action      = "BLOCK_WITH_RESPONSE"
  block_response_code = "SERVFAIL"
  state       = "ENABLED"
  order       = 8
  rank        = 7
  protocols   = ["ANY_RULE"]
}
```

5.2.3 SSL Interception

Pay close attention to the list of Cloud Applications assigned to Developers and change the action by uncommenting the relevant DECRYTP section of the resource.

None

```
resource "zia_ssl_inspection_rules" "SSLBypass" {
  name          = "SSL Bypass"
  description   = "SSL Bypass"
  state         = "ENABLED"
  order         = 1
  rank          = 7
  road_warrior_for_kerberos = false
  platforms     = ["SCAN_IOS", "SCAN_ANDROID", "SCAN_MACOS", "SCAN_WINDOWS",
"NO_CLIENT_CONNECTOR", "SCAN_LINUX"]
```

```
url_categories= [zia_url_categories.AppleBypass.id]

action {
  type = "DO_NOT_DECRYPT"
  # show_eun       = false
  # show_eunatp    = false
  do_not_decrypt_sub_actions {
    server_certificates = "ALLOW"
    bypass_other_policies     = false
    block_ssl_traffic_with_no_sni_enabled = false
    min_tls_version           = "SERVER_TLS_1_0"
  }
}


resource "zia_ssl_inspection_rules" "DevelopersCases" {
  name          = "Developers Special Cases"
  description   = "Developers Special Cases"
  state         = "ENABLED"
  order         = 2
  rank          = 7
  road_warrior_for_kerberos = false
  platforms     = ["SCAN_IOS", "SCAN_ANDROID", "SCAN_MACOS", "SCAN_WINDOWS",
"NO_CLIENT_CONNECTOR", "SCAN_LINUX"]
  url_categories= [zia_url_categories.developerSafeSites.id, ]

  action {
    type = "DO_NOT_DECRYPT"
    # show_eun       = false
    # show_eunatp    = false
    do_not_decrypt_sub_actions {
      server_certificates = "ALLOW"
      bypass_other_policies     = false
      block_ssl_traffic_with_no_sni_enabled = false
      min_tls_version           = "SERVER_TLS_1_0"
    }
  }
}

data zia_url_categories developer_tools {
```

```
  configured_name = "Developer Tools"
}
data zia_url_categories system_tools {
  configured_name = "Information Technology"
}

resource "zia_ssl_inspection_rules" "DevelopersChatchAll" {
  name           = "Developers Catch-All"
  description    = "Developers Catch-All"
  state          = "ENABLED"
  order          = 3
  rank           = 7
  road_warrior_for_kerberos = false
  platforms      = ["SCAN_IOS", "SCAN_ANDROID", "SCAN_MACOS", "SCAN_WINDOWS",
"NO_CLIENT_CONNECTOR", "SCAN_LINUX"]
  url_categories= ["DEVELOPER_TOOLS", "OSS_UPDATES"]
  cloud_applications         = [
"APACHE_JENA", "APACHE_PIVOT", "AUTOFLOW", "CONYEDIT", "ZEROQODE"
"APACHE_COCOON", "SLIM_FRAMEWORK", "APACHE_TEZ", "MACARON", "DWGSEE"
"STREETLAYER_API", "DOODLE3D", "SHAPE_WAYS", "QUESTIONSCOUT", "SEAMLESS3D"
"SENSEYE_PDM", "CODETOGETHER", "PANOTICA_HYDRA", "PASTEHTML", "MODELUR"
"PAPERCUPS", "GOODLOOKING", "TRADLY_PLATFORM", "SYN_APPS_MOBILE", "WEBHOOKS_IO"
"BUDDY", "AUTOVUE", "CARQUERY", "HCL_ACCELERATE", "ARIZE_AI"
"RUBY_DATUM", "THE_WELKIN_SUITE_IDE", "WIREFY", "SIMPLIFY3D", "METRIC_INSIGHTS"
"BUILDBOT", "KONCEPTAPP", "REMA", "GRAPHICRIVER", "FLICKITY"
"OBJECTGEARS", "CHAINER", "APACHE_BIGTOP", "TWEENY", "APPSMOMENT"
"BIMDATA_IO", "CAMALEON_CMS", "ECLIPSE_SYSDEV", "THREE_POINT_ZERO_SIDEDCUBE",
"UBERTESTERS"
"POLYMER", "APACHE_STRUTS", "APPDRAG", "ELEGANT_THEMES", "CONTENTA_CMS"
"MOCHA_PRO", "WAREWOLF", "PINEGROW", "SCHOOL_SCRIPT", "APACHE_DELTASPIKE"
"BOXCAST", "SHORTCODES_ULTIMATE", "COMMANDO_IO", "ART_TEXT", "RAGAPA_CAPTIVEXS"
"THYMELEAF", "MYECHO", "DATAWIRE", "SEMATEXT", "FLIPBUILDER"
"KABBEE", "APACHE_CAYENNE", "CONTENTSTACK", "EXCEPTIONLESS", "PHUSION_PASSENGER"
"APACHE_PHOENIX", "CLEVERTAP", "PLAYBOOKUX", "FEATUREMAP", "CPPCHECK"
"VISAGE", "LOGODESIGN_NET", "CINNAMON", "APACHE_SYNCOPE", "THINKGEO"
"UCALC_BUILDER_OF_CALCULATORS_AND_FORMS", "CRYSTALLIZE", "IGNITE_REALTIME",
"HANGFIRE", "GRAPHITE_GTC"
"CODEALIKE", "INPLAYER_PAYWALL_PLATFORM", "ROBOLOGIX", "APACHE_CALCITE",
"CUBBIT_FOR_TEAMS"
"FORMTAB", "UXCAM", "PUREBASIC", "GARAGEPLUG", "RARCHY"
"CORDIAL_SAS", "MOTO_CMS", "APPMAKER_XYZ", "PUB_HTML5", "DEALERS_UNITED"
```

```
"REVIEWSTUDIO", "AUDIO_WEAVER", "PRISMJS", "SEMANTIC_UI_REACT", "THE_LIBRA_TOOLKIT"
"JFROG_PIPELINES", "ICECREAM_PDF_SPLIT_AND_MERGE_FOR_MAC", "OPENGTS", "WEBX", "JIGGY"
"BROWSHOT", "CROWDTESTERS", "EMC", "APACHE_SPARK", "HASHICORP_CONSUL"
"MIRANTIS_OPENSTACK", "MAVEN_CENTRAL", "OPENWGA", "ISIMA", "STATION_FIVE"
"NEAR_ME", "SEARCHMAN", "WAMPSERVER", "WEAVEWORKS", "BID_SAW"
"PARKLOGIC", "PIXELIXE_STUDIO", "SK1", "APPLESEED", "NOMADESK"
"MOBILOITTE", "APACHE_AIRFLOW", "MAPZEN", "IONIC_DOCUMENTS", "NLPP"
"ACTIVE_QUERY_BUILDER", "QUATTOR", "SERVICEPRO_NET", "CC3", "MYCROWD_QA"
"ELFSIGHT", "UPCODES", "CARS_INTERNET", "SHOTCUT", "RATE_MONITOR"
"SOCIAL_CREATOR", "NEW_RELIC_CODESTREAM", "IBUILDAPP_ENTERPRISE_APP_STORE",
"TEXIFTER", "CERTHAT"
"PHALCON", "GIGAFOX", "VSTUDIO", "CODIQA", "SOURCEGRAPH"
"IMAGINESOFTWARE", "SHOOTER_SUITE", "AIDAWEB", "ICESL", "QUILTDATA"
"POLYMAPS", "SURGE_AI", "XWIKI", "HELP_GENERATOR", "MAPSTED"
"REQSTUDIO", "ZILVERLINE", "CRAB", "PDFJET", "DIOLKOS"
"OMNIGRAFFLE", "APACHE_TOMCAT", "GLYPHS", "IQUALIF", "ODIN_AUTOMATION"
"APACHE_AMBARI", "MAGICBELL", "RAPIDAPI", "DESIGNRR", "RAZOROPS"
"URHO3D", "APACHE_JCLOUDS", "RENDERNETWORK", "OCTOPUS", "APACHE_SENSSOFT"
"SEEKSTORM", "BLAZEGRAPH", "SCRIVITO", "WEBIX", "VIOLATION_RADAR"
"MANTRA", "PROKON", "OPENMAKE", "FIBERY", "TEXTASTIC"
"QUIXXI_SECURITY", "BUGTRACK", "SIMPLION", "NATIVEBASE", "RORVSWILD"
"JSREPORTS", "DATKO", "PHASER", "APTANA", "HANDBRAKE"
"CFWHEELS", "BROWSERSTACK", "HANDLEBARS", "XIMMIO", "NOTERRIFIC"
"WEB_CONTENT_EXTRACTOR", "CORONA_SDK", "CAMAZEE", "FYIPE", "CAMA_SOFTWARE"
"SELENIUM_HQ", "EMBRACE_IO", "FMOD", "AMAZEE_IO", "APACHE_NUTCH"
"REQUIREJS", "SDLCFORMS", "LORDICON", "OCTOPUS_DEPLOY", "VOXIMPLANT"
"APACHE_SHIRO", "RENTAL_SOFTWARE", "AXURE_RP", "AUTOMATIC_DUCK", "BULLET_TRAIN"
"JSDELIVR", "SECURELY_SEND", "CHIEF_ARCHITECT_PREMIER", "WEBTRANSLATEIT_COM", "PDF_CO"
"APACHE_HELIX", "LANSA", "UNITY", "NUMBER_ANALYTICS", "ECAMM"
"Q_FI_SURVEYS", "ASPOSE_TOTAL", "TURNINGCLOUD", "CHERRYPY", "ANDROMO"
"APACHE_SQOOP", "FORMIDABLE_FORMS", "WEKINATOR", "ONLYMEGA", "PAPERWISE"
"TIKI_TOKI", "READ_THE_DOCS", "USERFLOW", "TOAD_WORLD", "WEBPROTEGE"
"WING_PYTHON_IDE", "ROOTCAUSE", "WILDFLY", "SOCKET_INTEGRATIONS", "NEBULA_GRAPH"
"SCALATRA", "MOCK_IT", "MYFXBOOK", "HTML_TIDY", "QUICKWORK"
"SCIRRA", "SUBSCREASY", "FLOWBOX", "STACKSTORM", "PETROJS"
"CODILITY", "CANVASJS_CHARTS", "OUTSYSTEMS", "UXRISK", "ALLSYSTEMSMAX"
"INSTANT_ESTIMATOR", "NIKKTTO", "REACT_STARTER_KIT", "APACHE_ROCKETMQ", "CLOUD_LSI"
"QUIXXI_LICENSING", "MARATIS", "MILIZE", "PHRONTEX", "VOIDTOOLS_EVERYTHING"
"CONTINO", "TEXTMATE", "OPS_MX", "WONDERFLOW", "FLASHDEVELOP"
"MODSY", "INSTAFEED_JS", "PLANNING_PLUS", "SPRING_ENGINE", "DAOCLOUD"
"PURE_CSS", "DEPLOYBOT", "PHOTOSWIPE", "NXTRENDER", "FILE_REQUEST_PRO"
```

```
"SITESELL", "VELOCITY_JS", "ENABLEX", "JITPACK", "WEBSITEBAKER"
"APPONBOARD", "NOCLOUD", "IMAGESLOADED", "SNEWS", "PUSHMON"
"LI3", "ABLECOMMERCE", "RAML", "BLUEJ", "GRABZIT"
"RENTEON", "VENNGAGE", "MONIMBUS", "SITEEDIT", "CONFIGHUB"
"LOCATE32", "MOMENT_JS", "PYUP", "APACHE_FLUO", "ASTROPRINT"
"LABISITE", "ENGAGESPOT", "APACHE_HAWQ", "ONSEN_UI", "TESTOBJECT"
"DANCER", "CLARA_IO", "I_MONIT_", "JEFFREYAI", "PASTED_CO"
"PROTEGE", "PRERENDER", "STREAMLINE_MEDIA_GROUP", "GRAVITY_FORMS", "B2EVOLUTION"
"BIDTRACER", "NIOS_4", "WORLDFORGE", "OPTIMOLE", "PYTHON_CELERY"
"CLOUDREPO", "TERMIUS", "VERTEEGO", "ABLY", "APACHE_ANY23"
"ONESOFT", "SIP3_IO_TAPIR", "SUNEDITOR", "XORM", "ZEROMQ"
"CRITICAL_CSS", "CHARTSBIN", "ARGU", "DAML", "THANKSTER"
"BCFG2", "COMETCHAT", "LAUNCHDECK", "PIXELSQUID", "DSP_LAZARUS"
"BUILDKITE", "OMEGA_CMS", "AIRFOCUS", "APACHE_SERVICEMIX", "CODEFRESH"
"LANTEK_EXPERT", "LIVEGAP_CHARTS", "KODI", "CAPTURE_BY_TECHULUS", "TAGSPACES"
"TESTRIGOR", "BABELFORCE", "WWISE", "UPTALE", "KEYMARK"
"STRAPUI", "FLOWPLAYER", "STRIDE", "PIONEER_SOFT", "LIGHTTPD"
"APACHE_NIFI", "ECAMM_LIVE", "WIDGY", "FLEET", "WONDERSHARE_MOBILEGO"
"APPTOOLS", "AERISWEATHER", "KAPWING", "EPSAGON", "PYXLL"
"URGENT", "VALUEKEEP", "NO_MAGIC", "SCROLLREVEAL_JS", "TESORIO"
"DO_O_MATIC", "STYLECI", "SCREENIUM", "YARNPKG", "BOARDROOM_IO"
"POEDITOR", "SOFTEC_APS", "EASEL_LY", "BACKBONE_JS", "APACHE_HTRACE"
"OCF", "STORIE", "MEETINGWORDS", "EXPONENT", "SCULPTRIS"
"LITTLSTAR", "CLOUDANT", "SPLINT", "DAUX_IO", "PASTE_ORG_RU"
"THUNKABLE", "CODETREE", "A_PARSER", "TEXTSEEK", "HELPNDOC"
"MOBINCUBE_DIY_APP_BUILDER", "LOOKAT", "ELEMENTS", "CENTRALPOINT", "BUGAWARE"
"OXYGINE", "SLIMFAQ", "EMBER_JS", "TOTAL_RENTAL", "SIMPLYAUGMENTED"
"APPARMOR", "GOGS", "UCLUSION", "CM4ALL_SITES", "VERSIONPRESS"
"QUIKWEB", "BULK_RESIZE_PHOTOS", "SANDVOX", "ZEPLIN", "LEASEPOINT"
"TOPCODER", "CLOUD_ELEMENTS", "SAAS_MAKER", "SIMPLIFIER", "ENHANCE"
"PROBETURION", "PBS", "STENCYL", "CAFFE", "VAADIN"
"COLLABORO", "PROTT", "MINDBREEZE_INSPIRE", "MOBIRISE", "APACHE_BAHIR"
"APACHE_GUMP", "VECTOR_MAGIC", "PLOTTABLE_JS", "TOTALCLOUD", "REWOO_SCOPE"
"CRMSUITE", "ZENATON", "EMAZE", "TERMII", "WESCHEME"
"CHART_JS", "IMBACHAT", "SPRINT_BOARDS", "LINDENLAB", "LISTASO_SALES"
"WAG", "SPRITEBUILDER", "CODE_BLOCKS", "HUBAPI", "ICA_ATOM"
"USERTESTING", "AMAZON_DYNAMODB", "CHURNZERO", "STORE_LOCATOR_WIDGETS", "KUBEFLOW"
"QUESTION2ANSWER", "DREAMCATCHER", "VALISPACE", "GOOGLEAPPMAKER", "SOCIALSHAKER"
"SQUADCAST", "MLPACK", "AUTODESK_TINKERCAD_SYSDEV", "PANINTELLIGENCE", "TEXT_CAPTURE"
"SAVANNAH_GNU", "MAXAR", "ARTISGL_3D_PUBLISHER", "JFROG_CONTAINER_REGISTRY",
"MOOSOCIAL"
```

```
"MITHRIL", "SIBERIAN_CMS", "INSPECTOR", "MOCKFLOW", "ANALYTICS_CANVAS"
"QOVERY", "DUFFEL", "POPMOTION", "INFOGAMY", "DATAON"
"UNDO", "REQBIN", "ULTRAMAIL24", "ICECREAM_EBOOK_READER", "BETTY_BLOCKS"
"WIDEO", "ANYCHART", "FROSTBITE", "LAGURU", "APACHE_PREDICTIONIO"
"APACHE_EAGLE", "STORYBLOK", "TOPOL_IO", "SIMPLE_GRID", "AXONATOR_INC_"
"MEDIAELEMENT_JS", "VOOG_PREVIOUSLY_EDICY", "MORGUEFILE", "TORO_CLOUD", "CSD_SOFTWARE"
"TRANSLATION_IO", "SPRING_EDGE", "GWT_GOOGLE_WEB_TOOLKIT",
"SCALEWAY_KUBERNETES_KAPSULE", "HORDE3D"
"STENCIL", "PIXELMATOR", "QUICKDBD", "COCONUT", "ALKANYX"
"DOT_NET_FIDDLE", "CODIFIED_SECURITY_INSTANT", "MEMCACHED", "NUMERAL_JS", "OMNIA"
"MAINSIM", "KOYEB", "PLUTORA_RELEASE_MANAGEMENT", "RAVENPACK", "APACHE_XERCES"
"VCHECK", "ZOOMSHARE", "DIM3", "WUFOO", "BUGCROWD"
"PATTERNRY", "APACHE_CLEREZZA", "BANSHEE", "BIDSWITCH", "RABBITSOFT"
"STREAMINGVIDEOPROVIDER", "NETWORK_NEXT", "SPEEDSHEETS", "KRPANO", "XML_DIRECTOR"
"KOHANA", "MYPAINT", "ZOOMDATA", "MONOLITH_AI", "ROSETTE"
"DRAW", "SUPERSCRIPT", "TESTCHAMELEON", "ALPHA_ANYWHERE", "DHIWISE"
"QUALZZ", "CMAPS_ANALYTICS", "MODERATECONTENT", "PACKAGECLOUD", "APPREEF"
"SHEETSU", "LMN", "ENDAVO", "LINEARB", "STRATUS5"
"RUBYONGEMS", "BUBBLE", "PADRE", "MECHANIC_S_MATE", "APACHE_THRIFT"
"GHOSTBSD", "PLACEIT", "APACHE_BUILDR", "HELPSCRIBBLE", "SKITTERPHOTO"
"MYDATAMODELS", "WEBSITES_FOR_ACTORS", "CHECKLY", "VALO_INTRANET", "ROADMAP_PLANNER"
"SMTPBOXES_COM", "APACHE_HAMA", "CMS2CMS", "ROLLAPP", "KEYSHOT"
"NEOVIM", "CONVRRT", "FIELDGOAL", "FUSION_SYSDEV", "ARGOS_RPA_"
"LOCALAZY", "EQMARS", "ARTIFICIAL_SOLUTIONS_TENEO", "FLUIX", "SIDENGO"
"WEBSITE_DESIGN", "MAPGAGE", "WEB_OPTIC", "EASY_BUILDER", "ZEPTO_JS"
"REPLAY_IO", "CLOUDBOOST", "WARP_10", "SITEDISTRICT", "SCALEDRONE"
"PRESSPLUS", "HOMESTEAD", "FORMSTACK", "BLAPPSTA", "IDANGERO_US"
"ALICE_APP", "BITRISE", "KAJONA", "SOUNDJAM_CO", "ACME_TEXT_EDITOR"
"XIRSYS", "SCOPELAND", "TAGPLAY", "NATIVEFORMS", "CSP"
"DELTA_DRONE", "IPIFY", "SITER", "HADOOP_HDFS", "DEVIZE"
"INVIDEO", "VIDEOBOLT", "APACHE_TRAFFIC_SERVER", "TSURU", "GRAVITY_SKETCH"
"GINGERSAUCE", "MESHLAB", "SITEFOTOS", "VIDEOPROC", "EXPRESSPCB"
"LOCIZE", "APACHE_FELIX", "GAMEPLAY3D", "MAGNETO", "OPENSMPTD"
"QUERYCLIPS", "APPS_PANEL", "ELDARION", "KEEL", "NUCLOUD"
"VIDICT", "AUGMENT", "UNIVERSAL_DATA_TOOL", "ETHION", "IDMR_INFORM"
"ALERTIFY_JS", "NEOCODE", "MICROSTUDIO", "SNAZZYDOCS", "ZEIT"
"COTONTI", "UPVERTER", "BOTSCREW", "FORTY_SIX_POINT_ZERO_ELKS", "CAKEPHP"
"PACKAGIST", "SERVOY", "MANTISBT", "WP_SITE_IMPORTER", "FLOWIZE"
"ACUITYCRM", "MAKEVT", "ALTIUM", "CHAOS_GROUP", "FORETELLIX"
"WYN_ENTERPRISE", "LOCATIFY", "BRL_CAD", "KWES_FORMS", "PIXELGRADE"
"CLICKINSIGHTSIO", "ADVANTECH_WEBACCESS", "POEDIT", "APACHE_GIRAPH", "PACE_SUITE"
```

```
"STORMFORGER", "LEPTON_CMS", "OWIS", "PAYARA_SERVER", "WEBAUTOMATION_IO"
"MASSMAILER_DOCS", "AXTERISKO", "SYNOVIA_SOLUTIONS", "WAVOSAUR",
"DESIGNMODO_POSTCARDS"
"TEST_IO", "AETOPIA", "DESIGNBOLD", "DOC_DIGITIZER", "OCSIGEN"
"THINGX_CONNECT", "CLIPPING_MAGIC", "VDOWNLOADER", "GREENSOCK_ANIMATION_PLATFORM",
"PRSHOTS"
"SOURCEPOINT", "SEVEN_HUNDRED_AND_TWENTY_NINE_POINT_ZERO_SOLUTIONS", "ELGG",
"DESIGNMODO_SLIDES", "GROSVENOR"
"PHPIXIE", "POLLJOY", "ICSHARPCODE", "D_AMIES_TECHNOLOGIES", "NETFLIX_CONDUCTOR"
"SILVRBACK", "BEAVER_BUILDER", "SPEEDMENT", "CONCOURSE_CI", "MAPZANIA"
"HEPTAWARD", "SWEET_HOME_3D", "APPELIS_MOBILE_APP_DEVELOPMENT_PLATFORM", "BEYOND",
"AFFECTIVA_AFFDEX"
"SYNX", "XORICON", "VIDDYOZE", "METAMUG", "GRAPHICSPRINGS"
"YOUPIC", "CSS3PIE", "SOLIDCAM", "UB_FRAMEWORK", "PAPERBITS"
"SCRUMPY", "PAYHIP", "SIMPLESITE", "ECUBE_APPS", "APPTECH"
"KUULA", "AM", "MEDIASHAREIQ", "ADOBE_BUSINESSCATALYST", "MERLOT_AERO"
"BACKBONE_PLM", "SUPERANNOTATE", "UNOAPP", "ESTIMOTE", "WORDPRESS_VIP"
"JUSTLIKEAPI", "SPINE_ANIMATION_SOFTWARE", "WEM", "ATILEKT_NET", "CODEPLY"
"VANNEVAR_LABS", "FLOT", "SORTSPOKE", "SKULIBRARY", "AUTORAPTOR_CRM"
"APACHE_ROLLER", "SUNEIDO", "QUARTZ_JOB_SCHEDULER", "DEAD_SIMPLE_CHAT", "CHARTS_MAKER"
"BXSLIDER", "JFROG_ARTIFACTORY", "CODEKIT", "ZGAMEEDITOR", "OPENCFU"
"SEO_TESTER_ONLINE", "OPENXAVA", "SVGATOR", "ADEACA", "APACHE_USERGRID"
"HIREFIRE", "READYMAG", "VOICEXP", "EYES", "TIMETONIC"
"BLUESKY_ANALYTICS", "BUGSNAG", "DOPPINS", "THEVIEWER", "PATHENGINE"
"S3BUBBLE", "LUMINAR", "HITS", "SEEKBEAK", "BIGVU"
"ORCHARD_CMS", "TRINKET", "DEMATIC_SPROCKET", "GO_SITEBUILDER", "WEB2PY"
"ONE_MILE_UP", "MARIONETTE", "ROCKSET", "APPSMAKERSTORE", "ALTIS_DXP"
"PIXELLIGHT", "PIXELZ", "WP_ALL_IMPORT", "JOTFORM", "ORCAD"
"QUICKCHART", "PROXEM_SOFTWARE", "XSOLLA", "COFFEESCRIPT",
"SCALEWAY_CONTAINER_REGISTRY"
"TLK_IO", "ZAPTEST", "FORMWORKS", "ONE_POINT_ZERO_LINK_IO", "ALMXPERT"
"BROWSER_UPDATETOOL", "COREBOOK", "MODELDBA", "PERCY", "MONODEVELOP"
"APIARY", "BLUE_PRISM", "REDLIST", "ZAPWORKS", "MEDIAFLY"
"APACHE_UIMA", "MEZZANINE", "APACHE_AIRAVATA", "PDF_TO_DOC", "ONLINE_LOGO_MAKER"
"EVOPOS", "CODE_CLIMATE", "HAMMER_JS", "YAKUAKE", "TANZLE"
"ARCHI", "STACKSWARE", "RANKTRACKR", "FITNESSE", "PEAR_NOTE"
"SEARCHBERG", "CHROME_MOBILE_DEVTOOLS",
"MICROSOFT_DEFENDER_ADVANCED_THREAT_PROTECTION", "GRUNT", "PANOROO"
"SYSTEMAX", "N1ED", "OBJECTBOX", "ONLIM", "PIDGIN"
"HALO", "NUSPHERE", "MAHARA", "BIGID", "HAZELCAST_IMDG"
"BLISK", "AMPRY", "APACHE_KNOX", "CHARITABLE", "DIVSHOW"
```

```
"FACTORIE", "JVSG", "CODE", "SHAREPOINT_IMAGE_MAPS", "SIMPERIUM"
"AIRSLATE_BUSINESS_AUTOMATION", "DEVEO", "PUSHPAD", "QUIZBREAKER", "GOCANVAS"
"JIRA", "WEIGHTS_BIASES", "CLOUDCRAFT", "TORQUE3D", "EXTRALI"
"MATERIAL_GALLERY", "APACHE_REEF", "APIFLASH", "SLICKEDIT", "THE_NOUN_PROJECT"
"AUTODESK_ARNOLD", "VIDYO_IO", "SALVAGENOW", "IBUILDAPP", "MAKEPRINTABLE"
"APACHE_ARROW", "FORMSORT", "KERNELCARE", "PIKWY", "MATHWORKS_MATLAB"
"POLYGON_CRUNCHER", "PUSHSAFER", "WASTETRAKR", "TESTCENTER", "CLEAROUTPHONE"
"LEANPUB", "APPIAN", "SOLOFOLIO", "RAIDRIVE", "SUHOSIN"
"APACHE_BLOODHOUND", "OPERATR", "SHAPELETS", "SPACELIFT", "DIGIOH_LIGHTBOX"
"GEOPLUGIN", "THREE_POINT_ZERO_D_COAT", "MANICARE", "WHERE_S_WHAT", "JAILER"
"WAVEMAKER", "APIQ", "ALTIUM_CIRCUITMAKER", "APACHE_ZEPPELIN", "BARCODEWIZ_ONLABEL"
"MAKR", "ZESTY", "AUTOSHOP_CONTROLLER", "WAITRONBOT", "NEW_OXATIS"
"JDOODLE", "SPEQIT", "VSDC_FREE_VIDEO_EDITOR", "PROGRESS_KINVEY", "WPHUBSITE"
"STACKPILE", "CLOUDSPONGE", "MEDIANEST", "GOOGLE_DEEPMIND", "RSSHEAP"
"SIFTER", "PLATFORM_SH", "TETRA_INSIGHTS", "WIREEDIT", "IOQUAKE3"
"SQL_SERVER_BACKUP", "STACKER", "MOMENTA_CMMS", "FORTY_TWO_POINT_ZERO_WINDMILLS",
"ICONSCOUT"
"HUMANITEC", "ECLIPSE_IOT", "NLREG", "LIVECODE", "LINOTYPE"
"OMNIQUE", "THINK_CELL", "APMONITOR", "FELGO", "VECTA"
"PROUDSYRUP", "NEAR_SPACE_LABS", "PEPPERMINT", "PDF_COMPLETE", "INBOLD_SOLUTIONS"
"PADDEE", "FLUTTER", "USABITEST", "PARTSBOX", "MAGIC_BULLET_SUITE"
"JOURNEYAPPS", "SPLIT", "APPINSTITUTE", "TARI", "LOOPBACK"
"ZETAPUSH", "MANTA_FLOW", "JMONKEYENGINE", "EEMS", "MARVEL"
"PINPOINT", "QUOTEBREEZE", "ANYLOGIC", "FORMCARRY", "MIT_APPINVENTOR"
"GAVAGAI_EXPLORER", "NLTK", "WEEBLY", "SPLITFORCE", "HASHTAGIO"
"COILED", "SPRINGML", "MONKOP", "PORTEUS_KIOSK", "RANOREX"
"NOESIS", "DOWN_STREAM_TECHNOLOGIES", "BLUEWORLD", "CASECOMPLETE", "STITEL_NETWORKS"
"TWISTED", "ALITU", "ENTERMEDIA", "COBOT", "VEV"
"STICKTAIL", "DYNOMAPPER", "BRAND_IQ", "GEMFURY", "SELFCAD_SOFTWARE"
"BASIC4ANDROID_B4A_", "DUBIDOT", "GAMEMAKER", "VIM_SYSDEV", "RC_LOCALIZE"
"REQTEST", "LEICA_GEOSYSTEMS", "PRAGMAMX", "DATAFINITI", "NETOBJECTS_FUSION"
"CDATA_API_SERVER", "CARD_IO", "TRANSPORT_API", "ATIR_STRAP", "EXTRACT_SYSTEMS"
"LOGOFURY", "MILKSHAPE_3D", "REFERLY", "SEATABLE", "LINKSTRATEGY"
"SOUNDWISE", "CALEMEAM", "MYDRAW", "NIXOS", "VR2020"
"BACKHUB", "JUMPCHART", "LAUNCHDARKLY", "BRAND_ENSEMBLE", "MOQUPS"
"MOBILIZER", "COMPOSITE", "APACHE_LENS", "LARAVEL_SHIFT", "UNFORM"
"VEXOR", "BEAM_BRAND_CENTER", "TEXTADEPT", "ARTISENSE", "GOODGRIDS"
"AUPHONIC", "FUELPHP", "HONEYCODE", "SNAPPII", "SIMCENTER_STAR_CCM"
"APPZILO", "ONEDASH", "WORDPRESS_THEMES", "FRONTITY", "AUTOMATION_STUDIO"
"F2", "CODEPILE", "TESTPROJECT", "BOOSTLINGO", "FORMSWIFT"
"APIAXLE", "ICECREAM_SCREEN_RECORDER", "NOTESTACK", "POLIIGON", "OFFSPROUT"
```

```
"ETAP", "MYDATAPROVIDER", "TALLY", "PLATFORMA_FLOWCHARTS", "REVIEWTYCOON"
"LINKBRANDR", "SKYLAB_ANALYTICS", "SILO", "CLEARSKY", "SPHINX"
"FEATOOL_MULTIPHYSICS", "MASHSHARE", "SEQWA", "APACHE_ALLURA", "NITROPDF"
"CUBICWEB", "SKYNOX", "UTILITIES_ONLINE_INFO", "DR_EXPLAIN", "HTML_KIT"
"UFO_ENGINEERING", "XONGOLAB", "GFX_SYMPHONY", "NETPLAN", "REGEXPLANET"
"APACHE_STORM", "WEBBY_GROUP", "APACHE_FLEX", "DATA_IMPACT", "APACHE_ISIS"
"APACHE_MINA", "HELPSTUDIO", "DIGITILE", "BOWER", "POSTHAVEN"
"TRAPCODE_SUITE", "CONFIGURE_IT", "STRIDETRACK", "INSTANTAUTOSITE", "LIME"
"SQL_DELTA", "QLONE", "APACHE_BOOKKEEPER", "PSF_LAB", "MY_VISUAL_DATABASE"
"ORBISGIS", "NEXUSFONT", "QUEPID", "KATE", "CODEGIANT"
"CODEBOTS", "ONLINE_JSON_VIEWER", "APACHE_MARMOTTA", "NUGET", "APPCIRCLE"
"BEEZER", "PSPAD", "BRAINCLOUD", "MUX", "LIFT"
"UPFLOW", "BOX2D", "STRATAGUS", "AGILE_STACKS", "ANALYSISPLACE"
"VAGRANT", "ZYYX", "OPENCAGE", "SITE3D", "WEBGEN"
"VBVOICE", "CALCAPP", "SIVUVIIDAKKO", "SCOMPONENTS", "ULTIMATE_UNWRAP_3D"
"INVIVOSTAT", "APACHE_ARCHIVA", "SIMULATIONX", "SQLGATE", "VERACITY_SCM"
"FATHOM_DYNAMIC_DATA", "WEBSCRIPT_IO", "PURAN_SOFTWARE", "SAASLER", "APACHE_SYSTEMML"
"NUTSHELLURL", "FAT_FINGER", "THEA_RENDER", "STORYBOARD", "NHOST"
"LOGOTYPEMAKER", "YOAST", "ADDY", "KOTLIN", "RED_FOUNDRY"
"FONTSPRING", "HPAGE", "HTML_ANGELS", "INTROBRAND", "TURBOGEARS"
"STACKBLITZ", "WEBSITEBUILD_NET", "VOILA_CABS", "APACHE_METRON", "FCS_DIRECT"
"ROCKETSHIPIT", "ARENGU", "IOPIPE", "QUANTA_WEBHIMS", "HIPCAST"
"PASTE_SX", "FOHLIO", "AUTOWORX", "K6", "AUDIOKIT"
"FONTASTIC", "PHOTOADKING", "NAVPOWER", "DADABIK", "KUDZU"
"BBBOOTSTRAP_COM", "KOBITON", "HUULA", "DRCEDIRECT", "QUIXEL_SUITE"
"SENSAT", "LEANCLOUD", "MATERIALIZE", "APPERY_IO",
"THREE_POINT_ZERO_SCALE_API_MANAGEMENT"
"ROLLOUT_IO", "AIRFUSION", "DOXTER", "SIX_FEET_UP", "TEXT_EDIT"
"CARTWEAVER", "VERIPHONE", "SWAGGER_UI", "MAJEEKO", "BAREBONES"
"DHTMLX_SPEADSHEET", "GDEVELOP", "APACHE_SLING", "BETA_FAMILY", "PSCAD"
"STEPFORM", "PANAYA", "SNIPAWAY", "PDF_GENERATOR_API", "PUNCH_SOFTWARE"
"AEROSDB_SMS", "APACHE_FLUME", "APPCELERATOR", "BRIGHTXPRESS", "TILT_BRUSH"
"GLITCH", "SNAPLITICS", "TACHYONS", "SMTPJS_COM", "WINDOWS_MOVIE_MAKER"
"POOLCAR", "SHOPACCINO", "TULIX_SYSTEMS", "MMHMM", "TEMPLATETOASTER"
"ENTALE", "NGROK", "LOADSTER", "NETTY", "BOKEH"
"CAPTISA_FORMS", "KALDI", "WEESWARES", "IDEAROOM", "DYNAMIC_WEB_TWAIN"
"LEAP_MOTION", "ENSCAPE", "APACHE_METAMODEL", "PLAYER_BEST", "SPHERE"
"SWAGGER", "MOO_DO", "MONGOTRON", "MORTGAGEHIPPO", "DLIB"
"APACHE_SENTRY", "GAMESALAD", "ASINHUNT", "APACHE_STANBOL", "THE_WILD"
"ANYFILE_NOTEPAD", "GENEXUS", "ICONTROLWP", "WORKSHOP_BUTLER", "PUBCODER"
"SIMLAB_COMPOSER", "SCALE_AI", "CODEKEEPER", "PASTEBIN_FR", "AOS"
```

```
"TWIXL", "BEHANCE", "SEQUENCE_JS", "SOFTOOLS", "SQLBAK"
"MOJITO_SITES", "WHAT_FONT_IS", "APACHE_CRUNCH", "DESKREE", "DROPSHIPME"
"WEBILE_TECHNOLOGIES", "WEBEDEN", "THREE_POINT_ZERO_D_SLASH", "VINYL", "CIRCUITLOGIX"
"COCOS2D_X", "QUICK_BASE", "MATALI_PHYSICS", "ZOTONIC", "ANYMOD"
"CONNECTEAM", "CORE_STACK", "MOCKUP_IO", "OPEN_WATCOM", "GAMEBRYO"
"IGNITEPOST", "MARZIPANO", "ZINGGRID", "APERTUSVR", "KORBYT"
"MODX", "EASY_WEB_EXTRACT", "SITEOP_MOBI", "OASES", "LORDUI"
"CAKE_BUILD", "DBASE", "RUDDERSTACK", "SHOUTEM", "DBCOMPARER"
"TIPTOPWEBSITE", "CODECOMET", "GOOGLE_FONTS", "WEBWAVE", "FLEXAGON"
"FRAMER", "SYMFONY", "HITFILM_EXPRESS", "ALLEGRO_CL", "OPENLINK_VIRTUOSO"
"BOSH", "KIVY", "BUILDBOX", "PRECOGNIZE", "CATALYST_BY_ZOHO"
"CONDITIONREPORTS", "HAYSTACK", "METADATA2GO_COM", "RESPONSE_O_MATIC", "NOOMAC"
"WINAIR", "APACHE_BVAL", "LINKTIGER", "ECOSIMPRO", "JQUERY_MOBILE"
"STREAMLIT", "TIZEN_STUDIO", "YOUZIGN", "OPENGEX", "CHARTURL"
"FINALCAD", "PIXELPOINT", "DIGITALBRIDGE", "SPARTA_SYSTEMS", "DOTCLEAR"
"RIZEPOINT", "MONITORO", "GULPJS", "LOCALIZE_IO", "AVOCODE"
"MONGOHQ", "SKEDLER", "ITRAK", "CHEKRITE", "FLOWSPACE"
"DREAMTEMPLATE", "NOMAD", "REPL_IT", "AG_GRID", "ESTIMASTER2"
"ULTRAEDIT", "SUSE", "WAPPLER", "EMBUNIT", "SEARCHTAP"
"VRAPTOR", "APPENATE", "HASURA", "KABOOMPICS", "UNDERSCORE_JS"
"DRIVY", "DWSIM", "TYPESCRIPT", "ICECREAM_PDF_CONVERTER", "ORX"
"RESONANCE", "SELENDROID", "INDIGO_RENDERER", "LEININGEN", "SCREENSPACE"
"BLUE_CEDAR", "CENTMIN_MOD", "CHATCAMP", "UPDRAFT", "FORMCRAFTS"
"JIGYASA_ANALYTICS", "YAPPES", "QUICKSLICKS", "APPGINI", "WHITESOURCE"
"OCURUS", "ANTIRYAD_GX", "COCOAPODS", "BOT_LIBRE", "CARANDAWAY"
"KLAROS_TESTMANAGEMENT", "ROMMANA_ALM", "BONSAI_SEARCH", "AMAZING_SLIDER", "BUGFEEDR"
"MAQETTA", "YODIZ", "JSONWHOIS", "XPRESSENGINE", "BUNIFU_FRAMEWORK"
"SAMSARE", "GROMMET", "AVONNI_CREATOR", "SHARE_YOUR_STORY", "QUTTERA"
"GRABYO", "ARCHEPLAY", "PAGECLIP", "POV_RAY", "AMPERSAND_JS"
"DOGTAG_PKI", "VARASSET", "EQUINOX_BOOKING", "GLUE_MEDIA_PUBLISHING_SYSTEM",
"RISKPULSE"
"STOCKSY", "MOCKUUUPS_STUDIO", "FINEUPLOADER", "JCPPEDIT", "WHIMSICAL"
"BLE_MOBILE_APPS", "BRIGHTSTARDB", "RINGCAPTCHA", "NATRON", "DMXREADY"
"KYTHERA", "FOTOWARE_DAM", "LIVEWEAVE", "UNIVERSAL_PASSWORD_MANAGER", "NATURALTEXT"
"SCSK_CELF", "HEEK", "PATHSENSE", "PANDA3D", "MIME4_NET"
"ENTERPRISE_DYNAMICS", "ESF_DATABASE_MIGRATION_TOOLKIT", "GURUJADA", "VECTARY",
"HYPERMILL"
"AWAREMANAGER", "DUPLICATOR_PRO", "HARVEY", "DIVERGENT", "MAXBLOX"
"DESIGNFILES", "FLOWMAPP", "VISUALCRON", "VEEPLAY", "SHAPESPARK"
"ESYNDICAT", "PASTEBIN_PT", "EIGHT_POINT_ZERO_BASE", "HIGLIGHT_JS", "PLAYCANVAS"
"SIGNALWIRE", "TENDERBOARD", "ATTENSEE", "PERCONA", "DISCOVERTEXT"
```

```
"DIAGRAMO", "OPENSHOT", "MONACA", "THINKGAMING", "HELP_MANUAL"
"VENUE_AN_ENABLEDWARE_PRODUCT", "KRITA", "JHIPSTER", "MANGO_ANIMATE", "PDF_BOB"
"LONGRANGE", "RCMS", "BOTMAKE_CHATBOT", "NUCLEO_ICONS", "METRIA"
"MONOTONE", "NPMJS", "REACTS", "EASYPOST", "ONLINE_PDF_CONVERTER"
"OPENUI5", "NEXAWEB", "ICECREAM_VIDEO_EDITOR", "POWERDMS", "PORTUS"
"VALENTINA", "FILEPURSUIT", "PLAY", "THE_SILVER_LOGIC", "ARCENTRY"
"SAILS_JS", "TURBULENZ", "TEST_ANYWHERE", "DROPSOURCE", "FREERANGE_STOCK"
"RBCOMMONS", "ANKR", "DOCMOSIS", "CARPROLIVE", "REALTIMEDESIGNER"
"SOFTINTERFACE", "GUIDESIGN", "LYCOS_MAIL", "HIBERNATE", "WINIT"
"JELLYFISH", "BOBCAD_CAM", "MORECUSTOMERSAPP", "UV_LAYOUT", "GIANT_SWARM"
"XERRIS", "BLUEFISH_EDITOR", "OUTSELL", "ROCKIT", "TESTBIRDS"
"JAM_PY", "FORMALIZE", "FORMWAREPRO", "SMS77_IO", "SNAGGY"
"NUCLIDE", "XONA_SOFTWARE", "PROTEUS420", "PERCEPTILABS", "EASYQA"
"NINETY_NINE_POINT_ZERO_INBOUND", "TRESHNA_ENTERPRISES", "PLANGRID", "COSEER",
"ERP_IN_CLOUD"
"GNUMIMS", "INTERMIX_IO", "REDOKUN", "SYNKRON_CMS", "MULLTTA"
"JQUERY", "BASE_SYSTEM", "MODE_ANALYTICS", "CLEARSTRING", "ASP_NET"
"BEECUT", "FUSE", "TRAVIS_CI", "INSTABUG", "CODE_TIME"
"KML_VIEWER_AND_CONVERTER", "VIRTUALPAPER", "PENCIL", "APACHE_TAPESTRY", "PHOTON"
"VANTAGE_CLOUD", "EACHSCAPE", "MAXSTAT", "SYNCTHING", "APACHE_UNOMI"
"QSANDBOX", "JEDIT", "ACUNOTE", "EDIFLO", "APPETIZE_IO"
"BRUIT_IO", "MIRRORFLY", "PACKETZOOM", "PHENIX", "VPLAYED"
"SCREENSHOTSCLOUD", "ALLGANIZE", "AZURE_DEVOPS", "DITTO_CLIPBOARD_MANAGER",
"RIVERSIDE_FM"
"TEAMDESIGN", "MYSOCIALAPP", "SKETCHBOX", "APACHEBOOSTER", "VUFORIA_ENGINE"
"REAPP", "WOODWORK_FOR_INVENTOR", "EBLEARN", "STRONY_Z_KLASA", "VISWIZ_IO"
"RAPIDDOX", "WAREHOUSEBLUEPRINT", "APACHE_JAMES", "AXISVM", "ACCELQ"
"VIDERIS", "Q_ACTION", "GUERILLA_RENDER", "ELEVATESOFT", "CLARISSE_IFX"
"SBUILD", "QALCWISE", "KATAPULTPRO", "STOCKIQ", "CALCFUSION"
"CIMS_ERP", "IRISVR", "TISTORY", "ICECREAM_PDF_EDITOR", "IPENV"
"RENDER", "PLAIDCLOUD_ANALYZE", "CDAP", "INTERACTIVE_GATES", "TELERIK"
"BCAST", "QUICKERSITE", "CARBONMADE", "LIME_TEXT", "SPINE"
"FORBINARY", "APACHE_CONTINUUM", "PUSHY", "SOURCEREPO", "WHOAPI"
"EZGENERATOR", "DRIVERSE", "OBERMIND", "TRIMBLE_FIELD_POINTS", "ANGULARJS"
"BUILT_IO_FLOW", "SCHEDUFLOW", "ZBRUSH", "MICROSOFT_POWERAPPS", "KNOCKOUT_JS"
"ENIGMA", "OPENAL", "USER_INSIGHT", "MATERIAL", "PITCHERO"
"PUNCHZEE", "ACCUMULO", "PATTERN_RECOGNITION_TOOLBOX", "NINE_POINT_ZERO_BITS",
"CAMPUS_AUTOMATION"
"READWHERE_MOBILE_CMS", "CREATELY", "AXIOM", "DAEMON_TOOLS", "FORMSITE"
"JET_PROFILER", "BEANSTALK", "XPODA", "AUTOCLOUD", "STOCKSNAP"
"PROCREATE", "APIGEE_EDGE", "OPENJUMP", "TREACYFACES", "SKYGLUE"
```

```
"VQSERVER", "SANDSTORM_FOR_WORK", "CUSJO", "FOXTOW", "JOGET"
"PIXTELLER", "STORE_MANAGER_FOR_MAGENTO", "VUNGLE", "LOGOMASTER_AI", "INFANYWHERE"
"EPIGRAPH", "WORLDVIZ", "EMBARCADERO", "MOBILEMMS", "SMARTMENU"
"ESPION", "BUZZTOUCH", "MENDIX", "APACHE_LUCENE", "SUNFLOW"
"POWERSOFT_AUTO", "EJABBERD", "PROOFQUICK", "RADAR", "LESS"
"BOOSTNOTE", "IKNODE", "SMARTYARD", "ROLLBAR", "SWIFTSPEED_APPCREATOR"
"ENVOYER", "INNOVATION_MANAGEMENT_DASHBOARD", "TUMULT_HYPE", "ACTRAN", "APACHE_OFBIZ"
"VIRTUEMART", "SIZZLEJS", "OGANRO", "CARVUE", "INTEL_DEVELOPER_ZONE"
"KEYZY", "FLATLOGIC", "SONETTO", "AIRLINQ", "SOCIAL_WIFI"
"PLATIO", "BARBERSTOCK_SYSTEMS", "MARKZWARE", "PROCEZIO", "UXPIN"
"CANJS", "FACSIMILE", "MEMBERDEV", "COSENTIAL", "BRICKCONTROL"
"FREELOGOSERVICES", "OCCIPITAL", "OCTOBER", "OPA", "SAFESHARE_TV"
"MAIL_DESIGNER", "SCOUT_APM", "APPHIVE", "FANCYBOX", "MY_BRAND_NEW_LOGO"
"INFRANODUS", "JQUERY_UI", "PAGESCREEN", "IDR_SOLUTIONS", "SCINTILLA"
"DIALOG_ANALYTICS", "WORKSHARE", "INTELFOLIO", "SPOTZI_MAPBUILDER", "BOARDINGBOT"
"JSSOR_SLIDER", "BITBUCKET", "THE_SMS_WORKS", "APACHE_SERVER", "DESKSPACE_CMS"
"SEPIA_SOLUTIONS", "KISSLICER", "CFENGINE", "WEBMECCANO", "FITVIDS_JS"
"SVRF", "TRUSTEDHEIR", "TWINMOTION", "PAGEPEEKER", "CREATEJS"
"APACHE_CHEMISTRY", "HERE_LOCATION_SERVICES", "VERGE3D", "TOCR", "DJANGO"
"VIDSTART", "OSHYN", "APACHE_CURATOR", "EXPERIMENTS_BY_GROWTHHACKERS", "USE_TOGETHER"
"DEDOOSE", "TIMEGRAPHICS", "GRATISOGRAPHY", "APPSGEYSER", "FITTEXT_JS"
"SCREENSHOT_GURU", "EXPLORERMAX", "PIPEONE_ME", "CONTENTDER", "RENTAL_CAR_MANAGER"
"REZZA", "DATATABLES", "OPENSIMULATOR", "PGMPY", "CYPRESS"
"MOCKLETS", "PANDASUITE", "PERAICHI", "APPWARP", "LOADVIEW"
"HELPSMITH", "CUBICASA", "CODOTA", "REN_PY", "GEOTRELLIS"
"OROPLATFORM", "VIZIAPPS", "PROJECTCODEMETER", "FORMSUBMIT", "SINTELIX"
"AURACHAIN", "VIDGRID", "FLUIDRAY_RT", "INCODOCS", "PARKAVE"
"REPLIT", "PILLIR", "THREE_POINT_ZERO_DSYSTEMS", "SUNFLOWER_ASSETS",
"GOOGLE_DEVELOPERS"
"PRORESPONSE", "AVO", "HABITAT", "MAVO", "ARES_COMMANDER"
"ARKIVUM", "YUJ_DESIGNS", "RAKUNEKO", "RAWTHERAPEE", "POSTICO"
"TXN", "PLATFORMIO", "PAPERLESS_FORMS", "MOBISCROLL", "UPDATABLE"
"BROSWERCMS", "ZENCAST", "IRISE", "PROOVL", "PADRINO"
"MUPDF", "NINJA_IDE", "URLBOX_IO", "ANDROID_STUDIO", "OPENEAAGLES"
"OPSGENIE", "PLANT", "TEXTTEMPERATURE_API", "HANDWRITE", "TYK_API"
"NEO_NET_ENERGY_OPTIMIZER", "RESCALE", "APACHE_FINERACT", "APACHE_GOBBLIN", "KROP"
"ATIUM", "REACTIVEX", "MICROBILT_DEVELOPER_PLATFORM", "BANNERSNACK", "KUBERNETES"
"SCRAPESTORM", "URIPORTS", "MKDOCS", "STRESSTIMULUS", "PRODUCTPLAN"
"CM_COM", "CODEANYWHERE", "SHEETLABS", "FEEDHENRY", "CLKER"
"COMSOL", "EASY_WEB_EDITOR", "HYLLO", "FROONT", "MATERIAL_UI"
"MACHINEWORKS", "ENDURO_JS", "TESTI_", "THEEMON", "AZORE_CFD"
```

```
"NOSQL_MANAGER", "TWILL", "FORNUX", "PUSHTABLE", "FLIPLET"
"WIZDEE", "CLEARSTORY_DATA", "HUGGING_FACE", "RUBY_ON_RAILS", "VOLT"
"ARCHONIC", "DEEPIN", "PHP_CRUD_GENERATOR", "CHILLIVAULT", "EAZYCODE"
"CODELITE", "REVOPS", "SKALE", "AGITAR", "EVERY8TH"
"QUTEMOL", "SIGMAPLOT", "ITEMPATH", "YESOD", "CLEARPREDICTIONS_COM"
"UBITRACK_UK", "SUPERNOVA", "LAND_F_X", "GITPOD", "ZIVRO"
"DISCO_PROJECT", "ZDOCS_PRO", "TWIG", "BUGHOST", "EFFORTIX"
"THOUGHTFARMER", "VELOCITYDB", "GAZELLE_PLATFORM", "TIMEHOP", "DRAFTBIT"
"LUMAVATE", "BLUESKY_INTERACTIVE", "SERPRIVER", "PAGE_REST", "SYNTHEYES"
"GRAVIT", "NG2_CHARTS", "FOUNDRY", "SCONS", "VISENZE"
"MADEDAILY", "FOUR_POINT_ZERO_A_ENGINE", "SQUARETEST", "EHCACHE", "WEB_DEVELOPMENT"
"THINK_UP_THEMES", "SPREADSIMPLE", "CLARITY_DESIGN", "N3TWORK", "POWWOW_MOBILE"
"MULESOFT", "PIIO", "D2IQ", "CREATIVE_MARKET", "YOJEE"
"CODELOBSTER", "APACHE_CTAKES", "BACKBEE", "NEURON_SOUNDWARE", "MAPNIK"
"QEMU", "NOTEVAULT", "PAW", "PROJECT_CHRONO", "SUBETHAEDIT"
"CODESHARE", "FLASK", "GEOPEEKER", "ULTRASERVE", "YAY_IMAGES"
"SKYCIV", "DITATOO", "EXISTDB", "TESTLING", "MOI3D"
"ONETWOTHREEFORMBUILDER", "VOTING_SERVER", "ROOKIEUP", "TOSMANA", "PAYCHECK"
"USERBRAIN", "ZEND", "LOGINWORKS", "PYLINT", "RAILS_ASSETS"
"BUGCLIPPER", "SHERIDAN_PUBFACTORY", "DARCS", "CHATSCRIPT",
"SYNCFUSION_ESSENTIAL_STUDIO_ENTERPRISE_EDITION"
"APIMAN", "AURELIA", "ANSIBLE_TOWER", "TELESIGN", "THINKERY"
"WALKIE", "FORTRABBIT", "JSHINT", "TRAVELTIME", "SQLFORMAT"
"APACHE_LIBCLOUD", "LOOKA", "FFMPEG", "GIDEROS_MOBILE", "EPIDATA_ENTRY"
"KENV", "J2STORE", "NEWSCI_PLATFORM", "APACHE_BROOKLYN", "WP_PIPELINE"
"LACROSSESHIFT", "SCHOOLOGY_DEVELOPERS", "APPSTRAND", "VTILITY", "IARIV_TOURS"
"METAFIZZY", "BLOCS", "MACSTADIUM", "KEYING_SUITE", "BLACKBERRY_ATHOC"
"ANYPOINT_API_PORTAL", "BOOTSTRAP_SHUFFLE", "PAGESUITE", "AMP_CMS", "SUBJECT7"
"TARSNAP", "WEPIK", "APACHE_TILES", "HBASE", "HAUTELOGIC"
"DIMENSIONEX", "FUNIFIER", "METEOR", "BREEZY_THEMES", "IKAN_ALM"
"EWWW_IMAGE", "KHRONOS_GROUP", "ICOMOON", "INFURA", "FOUNDEO"
"NEURAL_DESIGNER", "TIMELYVISIT", "CYCLES", "NEPTUNE_AI", "PRINTUI"
"MOCKINGBOT", "FOTOMOTO", "FOUNDATION_FOR_APPS", "SPLASHTOP", "IONIC"
"TISANE_API", "IMAGENAI", "LIVEEDIT", "DWKIT", "ASTROML"
"MOBILE_ANGULAR_UI", "UTEST", "RENDER_IN", "SIMPLYGON", "RAILS_LTS"
"TAGGUN", "CODE_SIGNING_CERTIFICATE", "TRUGRID", "LEADWERKS", "LINQPAD"
"TESTMODELLER", "FLAIRBUILDER", "CONCEPTDRAW", "APACHE_CXF", "NSOLID_PLATFORM"
"AREA28_TECHNOLOGIES", "FASTRAX", "JMOL", "WRAP_MEDIA", "WEJO"
"SINATRA", "DROOLS", "GNU_EMACS", "POLYFILL", "TURBOSQUID"
"SCROLLMAGIC", "DRIVEAXLE", "FLIXEL", "TICKCOUNTER", "JQPLOT"
"OBSERVIUM", "API_FORTRESS", "MOCKUP_EDITOR", "SMS_TO", "MOBITRACKER"
```

```
"CREATORIQ", "COFFEECUP_HTML_EDITOR", "SWING2APP", "ASCIIDOCTOR", "SIMPY"
"AWESOME_TABLE", "GENEBRA", "OCTANT", "MINDMELD", "NOTETAB"
"WEBSOLR", "REMOTEINTERVIEW_IO", "PUSHERAPP", "APIGILITY", "SECURE_GO"
"PHENOMIC", "BACKENDLESS", "ABCSUBMIT", "APACHE_BEAM", "SEASIDE"
"ELITE_FUNNELS", "HYPERWORKS", "BASEX", "ZENCASTR", "MUSE"
"APACHE_APEX", "PIVOTX", "GMSH", "KUBRIC", "ACMEAOM"
"TIZRA", "GOT_WWW_STORE_LOCATOR_SOFTWARE", "PLAYCO", "VOOG", "APEX_AI"
"BRIEFS", "NEUTRINO_API", "HIVO", "TWILIGHT_CMS", "APACHE_XALAN"
"UNISTORE", "URGENT_LY", "UXFOLIO", "INTERACTIVE_DMS", "LUMOA"
"DESYGNER", "ROCKY_DEM", "USERBIT", "PERL_GENIE", "ISCROLL"
"LOVE", "SCHEMATICS", "COOLORS", "NAS4FREE", "GUDANGADA"
"BETA_TESTING", "MOJOLICIOUS", "THINKSQL", "MINDJET", "ORACLE_APEX"
"UNSTACK", "RECAPTURE", "DWG_VIEWER_2_0", "PLAYFAB", "GONEVIS"
"NOOBAA", "DOCKER_HUB", "HTTPWATCH", "SHAREBASE", "DESKTOP_METAL"
"MORE4APPS", "SUMMIT_DIS", "GETADDRESS", "TOME_HOST", "SLACKWARE"
"SOFA", "CRASHLYTICS", "DOYPP", "INSTANTCMS", "HELIO"
"AURORA_2_1", "BOOMER", "CONTROLUP", "AUTOCODE", "ECOLEGO"
"GALATEA", "GREENHOUSE_CI", "MITSUBA", "BEECEPTOR", "PAYFORMIX"
"OPROMA", "SMOG_MASTER", "ALLEGRO_LIBRARY", "PEPPER_SQUARE", "SHIPBOB"
"DATA_TOOLBAR", "NEGATIVESPACE", "APACHE_KAFKA", "MOCKITO", "MONOGAME"
"MYMARKETPLACEBUILDER", "WEBGENZ_CMS", "SNOWFIRE", "INFOGRID", "SOLOLEARN"
"CODEPLEX", "OGRE3D", "WIX", "MUHIMBI", "TAPPLA"
"GRASS_VALLEY", "PREMIO", "MACHINIST_TOOLBOX", "OPENREAF", "SENSOR_TOWER"
"APPHARBOR", "SMOOLIS", "ENVISION", "USEREXPERIOR", "UDIG"
"KRAKEN_IO_IMAGE_OPTIMIZER", "FLEETNETICS", "ORBEON_FORMS", "IOTEX",
"PRO_SITEMAPS_COM"
"SUBLIME_TEXT", "RAYLECTRON", "ENGIE_CONNECT", "OMNIS_STUDIO", "ADEPTLY_AI"
"ASYNC", "VISUAL_PARADIGM", "CLOCKITIN", "TRANKYNAM", "PTERODACTYL_PANEL"
"CLUEMAKER", "GONITRO", "SILVERSTRIPE", "SWIG", "FDT"
"TALKJS", "OPENDDL", "MATHJAX", "SHORTCUT", "CLAYSYS_APPFORMS"
"LEGITEST", "POLYGON_CRUNCHER_SDK", "HTML_CART", "SPREADSHEETWEB", "BANJO"
"SINACLOUD", "STAN", "CURRENCYFREAKS_COM", "CALENDARIFIC", "NEVATECH"
"CMSMOVE", "PASTECRY_PT", "AGILITY_CMS", "ORGINIO", "DART"
"LETS_ENHANCE", "NETSAPIENS_SNAP", "PASTE_YT", "WAKATIME", "JAHIA"
"REALSOFT_3D", "KAJABI", "OVERFLOW_IO", "HSQLDB", "ADOBE_PHOTOSHOP"
"APACHE_JUNEAU", "WEB_CLIPPING", "GITENTIAL", "VARADA", "KICKPAGES"
"FORMIDABLE_VICTORY", "MSG91", "ADVANCED_CUSTOM_FIELDS", "FASTLY_LABS", "RESONANCE_AI"
"JSONBIX_COM", "APIMATIC", "GLIMPSE", "PIVOTALTRACKER", "APPFURNANCE"
"COMPONENTONE", "GENESISONE", "TESTLOG", "NEURON", "RAIJIN"
"ALTOVA", "ABBYY_CLOUD_OCR_SDK", "RENCATO", "CALCMASTER", "PUBNUB"
"PHPDOCUMENTOR", "VIRID", "IMAGIFY", "ATLANTIS_WORD_PROCESSOR", "BUGSEE"
```

```
"WINGS_3D", "WORKFLOW_ENGINE", "EZ_CAM", "LIVEPEER", "AMIO"
"VECTR", "TORNADO", "NCACHE", "HELPIE", "TIMELINEJS"
"GOESSENTIAL", "WOLFSSL", "HTTPMASTER", "ABOUT_ME", "ADEPTION"
"BERS", "MAPAL_TOOL", "REVIZTO", "ALTOSTRA", "BQURIOUS"
"XOOPS_CMS", "ECLIPSE_ENGINE", "KAGGLE", "ZED_BUILDS_AND_BUGS", "PUBLISHRR"
"SMART_ENERGY_WATER", "CODEIGNITER", "STREAM_FEEDS", "MYPHOTOAPP", "CHAPERONE"
"SEO4AJAX", "TRUVIDEO", "BLENDER", "DEVHAND", "PAGESTREAM"
"SLICKSTACK", "TRACKJS", "SUBTRA", "SUNVIEW", "XTRACTA"
"THREE_POINT_ZERO_BOX", "ARMORY", "EXOMATCH", "LASCOM", "SEOBILITY"
"MEERO", "LINKERD", "NEUROPH", "CRYSTAL_SPACE", "LUCID_SOFTWARE_LUCIDCHART"
"THE_JUPYTER_NOTEBOOK", "ANIMOTO", "ALIENBRAIN", "EPICA", "HVR"
"MGAGE", "PFORM", "FORMKEEP", "MOBSTAC", "CCV"
"LOGASTER", "NPM", "INBOXSDK", "FRAGMOTION", "DHTMLXPIVOT"
"IMPRESSCMS", "DESIGN_MASTER_ELECTRICAL", "REVIEWNINJA", "VIDDLY", "PRETE"
"PLAYERIO", "STOPLIGHT", "BANTERX", "GRAILS", "SEALPATH_IRM"
"THEAPPBUILDER", "UPLOADCARE", "EASYCIS", "HEX_PM", "PREVEDERE"
"PLAYTESTCLOUD", "STREAMLINEHQ", "TRNSYS", "SOFY_AI", "CODENINJA_AI"
"SQUARESPACE", "ADELPHIC", "WEBRTC", "SHOPIO", "NVIDIA_DIGITS"
"SPIDER_VO", "THREE_POINT_ZERO_DATA", "KITE", "AC3D", "ATOM"
"CODEMIRROR", "GERRIT", "AUTOSHOP_DESKTOP", "ASBRU", "MOBIROLLER"
"LOGOLOGY", "HIVECODE_IO", "DATAROOMS_COM", "DIA_DIAGRAM_EDITOR", "SUBRION"
"SIMEGO", "FULLPAGE", "CAMPUZ", "KALEIDO", "ACCORD_NET"
"MARGINFUEL", "IPCOP_FIREWALL", "TIZEN", "VPSSIM", "APISHUB"
"AUROS", "CESIUM", "META", "VANTEDGE_SOFTWARE", "TESTCRAFT_BY_PERFORCE"
"URLOOKER", "CENTILEO", "IONIC_APPFLOW", "RASTERIZER", "XP_DEV_COM"
"UNITY_ADS", "DASH_FOR_MACOS", "KICKAPPBUILDER", "TYPESQUARE", "WEECHAT"
"AVVIR", "PULSE_PLM", "CAPCUT", "VIDEOJS", "MICROTHEMER"
"S4A", "UNITY_MACHINE_LEARNING", "KNIME_ANALYTICS_PLATFORM", "GOOGLE_MAPS", "SEALD"
"CHEMDOODLE", "DOCKER", "PULLAPPROVE", "LOGIGEAR", "MYSQLDUMPER"
"PITCHER", "WIDGETIC", "FLOWPAPER", "LILABS_FILEDEPOT", "PYTHON"
"SYNDESIS", "AUTOACTION", "GUARDIAN_CMMS", "THREEDBROWSER", "ONEDAY"
"CACTI", "DHTMLXSUITE", "VOLUSION", "DUPLOCLOUD", "SPARK"
"LEAFLET_JS", "SMART_SLIDER_3", "CODEGRIP", "TEXTPAD", "VIRTUALBOSS"
"ANTIDEO", "MAAS_TECH", "INTEC", "LIMEWEB", "PINCELLO"
"KONSOLE", "BIOMETRIC_VISION", "FLYY_IO", "CERT_IN_SOFTWARE_SYSTEM", "APACHE_TOMEE"
"FONT_AWESOME", "INSTAPPY", "WEB_ASCENDER", "JSCRAMBLER", "YOUWORKFORTHEM"
"EASYCRON", "APACHE_ORC", "FAASTRUBY", "PALIGO", "APACHE_MYFACES"
"COGNITO_FORMS", "FIFE", "LUMENDATA", "FREESTYLE_PARTNERS", "SMARTVAULT"
"SCRUTINIZER_CI", "DATA_LOADER", "OPENFOUNDRY", "OCLINT", "POSTMAN"
"REALTIME_LANDSCAPING", "TYPOGRAPHY", "AZUL_SYSTEMS", "MAGEIA", "WSO2_API_MANAGEMENT"
"HAXEFLIXEL", "LIGHTNING_AI", "FLEEQ", "POOTLE", "CUSTOM_CODEX"
```

```
"NHP", "APACHE_FLINK", "TYLE", "ONE_PASSWORD_TEAMS", "PROCESSWIRE"
"WAYNE_REAVES_SOFTWARE", "PURECM", "NOX_SOLUTIONS", "SCHOOLSITEPRO", "SAVIDRAW"
"PRUVAN", "BACKTRACE", "THE_SCYLLA_GROUP", "BRIX_IO", "FASTHELP"
"NETIMAGER", "TEKNORIX", "QUANDORA", "FINAL_EFFECTS", "SWITCHER_STUDIO"
"SKETCH", "JIMDO", "KISSFLOW", "RESTREAM", "TAKES"
"FORMAPI", "STREAMLABS", "AUTOJINI", "FIREBASE", "STUNT_SOFTWARE"
"APPLE_DEVELOPER", "ARTISTEER", "SKETCHANET", "INCADEA_DMS", "SLIDE_BAZAAR"
"MYSHOPMANAGER", "PLESK", "APACHE_DIRECTORY", "CARDBOARDIT", "LIGHT_TABLE"
"MYELIN_IO", "FIGMA", "README", "STYLUS_STUDIO", "APACHE_ARIES"
"NUTSHELL", "SERVICESTACK", "ELYSIUM", "RAFAY_MANAGED_KUBERNETES_PLATFORM",
"APACHE_GERONIMO"
"DRIVEITNOW", "TEXEL", "GODOT", "APPSHEET", "DESIGNMODO_STARTUP_FRAMEWORK"
"CODEMAGIC", "NAPA_TRACS", "ANALANCE", "VIBE_D", "PIXOPA"
"CODEMRI", "GUIDETI", "RHINOCEROS", "EPICPXLS", "FIXD"
"SHEET2API", "APACHE_MESOS", "ICECREAM_IMAGE_RESIZER", "MEME_GENERATOR",
"HOTSCRIPTS_PRO_CHAT_ROOMS"
"EXPRESS_JS", "FILEMAKER", "MOOTOOLS_CORE", "QUNIT", "SPECIFI"
"ZUBTITLE", "DPASTE_COM", "SERVMASK", "SPARKLESHARE", "SIGNALMIND"
"APACHE_KUDU", "CONSTRUCTIVE_3D", "DESIGNER_TASK", "INFOCAPTOR_DASHBOARD", "PRINTBOX"
"KDEVELOP", "FOXIT_CONNECTEDPDF", "REACT_SHARE", "BELIVE_TV", "DESIGNER"
"FYBER", "GITLAB", "PDFTRON", "REACT_NATIVE", "PROBO_CI"
"SNIPPLR", "D3JS", "EZ_FORMS", "BUILDFIRE", "HTTP_TOOLKIT"
"HYPERGRAPHDB", "SAOLA_ANIMATE", "BOTSUPPLY", "CANVA", "APACHE_CHUKWA"
"SYNFIG_STUDIO", "PHRASEANET", "HVAC_CAD", "AVRO", "BASKETBALLSHIFT"
"K_3D", "LINEARICONS", "MACHINATIONS_IO", "APPSBUILDER", "BOTTLE"
"APPRENDA_MOBILE_APPLICATIONS", "OCI", "USEBERRY", "APACHE_CORDOVA", "CLIPSTREAM"
"ICONJAR", "SAUCELABS", "CODESTRIKER", "QCAMAP", "SNORKEL_AI"
"SEMANTIC_UI", "ROADMUNK", "SCIENCE_EXCHANGE", "APACHE_WICKET", "CASTR_LIVE_STREAMING"
"MODELICA", "WOCHIT", "BOOTBOX_JS", "CUSTOM_CURSOR", "PUBLIZ"
"DOTNETNUKE", "APPLICAN", "SITESUPRA", "WEBPAL", "DASHDASH"
"O_DEV_3D", "UNMINIFY", "SEMWARE_EDITOR", "UKIT_AI", "OPEN_VSWITCH"
"VLFEAT", "AUTOROX", "DATAIKEN", "DEPPBOT", "CONAN_IO"
"GARDEN_IO", "TENOR", "SUMMATTI", "INFINITE_SCROLL", "MASTERSTROKE"
"REDGIANT_EFFECTS_SUITE", "APP_DOTNET", "LIVEBOOKS", "CRAZYBUMP", "LINX"
"PROPOSALSMARTZ", "JMIX", "CHURCH111", "LIMOWIZ", "SPRINGTHROUGH"
"JARCHITECT", "PAS_MEDIA", "PECAN", "VALGRIND", "ZEN_FLOWCHART"
"BLUSOLUTIONS", "DOCFETCHER", "EPIC_CONJOINT", "AGENTY", "ANDPLUS"
"OTEEMO", "SQLSTREAM", "MULTIPARTNER_VDR", "IVISIT360", "HARNESS"
"CAFFEINE", "MECSOFT", "LYCIASTUDIO_IDE", "ROUNDME", "WIREFRAME_CC"
"SMSAPI", "SPIN_JS", "CANONICAL", "CANDU", "RAS"
"APACHE_YETUS", "STORIESONBOARD", "A_A_P", "ORYX", "PROTO_IO"
```

```
"STARUML", "QUICSOLV", "GRAND_PM", "PROXIMITY_KIT", "SHOHOZ"
"LOCALEDATA", "VEVS", "MEDIABANK", "BRANDMARK", "TESTUNITY"
"UNREAL_ENGINE", "CLANLIB_SDK", "GETPROFILES", "PDG_TOOL", "ZEND_TECHNOLOGIES"
"AUTOHOTKEY", "KUTAMO", "POPTIN", "LOGGLY", "SCALA_CENTER"
"FLATFILE_PORTAL", "BLAZEMETER", "RECONSTRUCT", "RIPS", "ANGULAR"
"APACHE_TURBINE", "CODERWALL", "FC2", "RED", "SOAPE_PLATINUM"
"TEXTSNIPER", "RELEASENOTES_IO", "CRATEDB", "BLACKPURL", "EMAGINE"
"APACHE_QPID", "LOADIUM", "PROPELLER_AERO", "PANBI", "GOLDSIM"
"PDFFORGE_PDFCREATOR", "AWARE_IM", "MOHIST_WEB_TECHNOLOGY", "FLOWROUTE", "SCRUMWISE"
"SITESUITE", "KALIPSO_STUDIO", "UNIDEV", "CONTINUUM", "OMBAQ"
"VISUALSITEMAPS", "REFLECTOR_3", "SOUNDATION", "YII", "NGINX_CONTROLLER"
"TREMOR_VIDEODSP", "KUBOTEK", "VHIZO", "ATLASSIAN_CONNECT", "MINKO"
"SOURCEFORGE", "SPLICER", "FREELOGODESIGN", "PHPRUNNER", "BASSETS_EDEPRECIATION"
"UMENG", "BIBLIONIX", "DEVELVE", "INSPIRE_WRITER", "DIPTRACE"
"APACHE_TINKERPOP", "COOKIEBOT", "SIMPHOLDERS", "MAXIMTRAK", "YOURTRADEBASE"
"XOJO", "CODESANDBOX", "IMOBILEAPP", "TCHOP", "GITHUB"
"SINGULAR_LIVE", "TENSORFLOW", "ZOOKEEPER", "FREEMAT", "SIMPALM"
"ICONARCHIVE", "WORKFOLIO", "KORTICAL", "AUTOMATION_ANYWHERE", "UNICORN_PLATFORM"
"BARKEEP", "DRAFTA", "QUICKPLUMB", "THREE_POINT_ZERO_DRESHAPER", "STORRITO"
"EXPOFUSE", "JS_BIN", "EARTH_POLIS", "ANGULAR_MATERIAL", "KNOWLEDGEWELL"
"FINDBUGS", "XSALTO", "RADZEN", "WEBASSETS", "REPLICATORG"
"NGX_BOOTSTRAP", "CROWDSPRINT", "URL_ENCODE_DECODE", "CODETASTY"
"APACHE_CELIX"
]
  action {
    type = "DO_NOT_DECRYPT"
    # show_eun       = false
    # show_eunatp    = false
    do_not_decrypt_sub_actions {
      server_certificates = "ALLOW"
      bypass_other_policies     = false
      block_ssl_traffic_with_no_sni_enabled = false
      min_tls_version           = "SERVER_TLS_1_0"
    }
  }

  # action {
  #   type = "DECRYPT"
  #   # show_eun       = false
  #   # show_eunatp    = false
  #   override_default_certificate = false
```

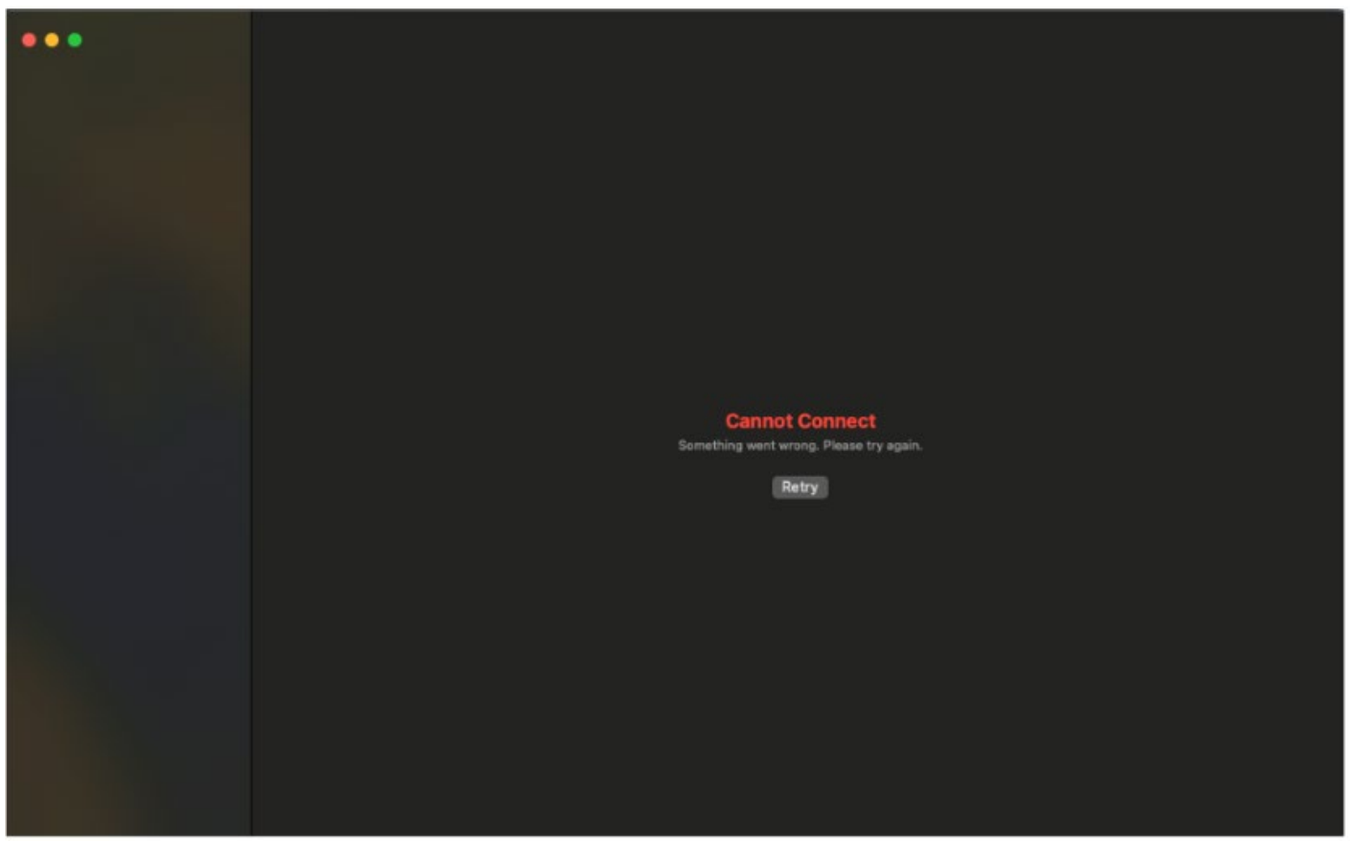
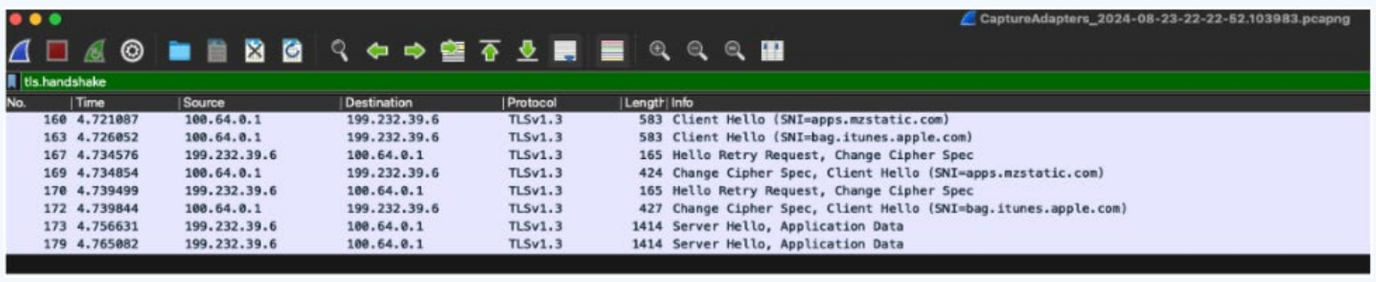
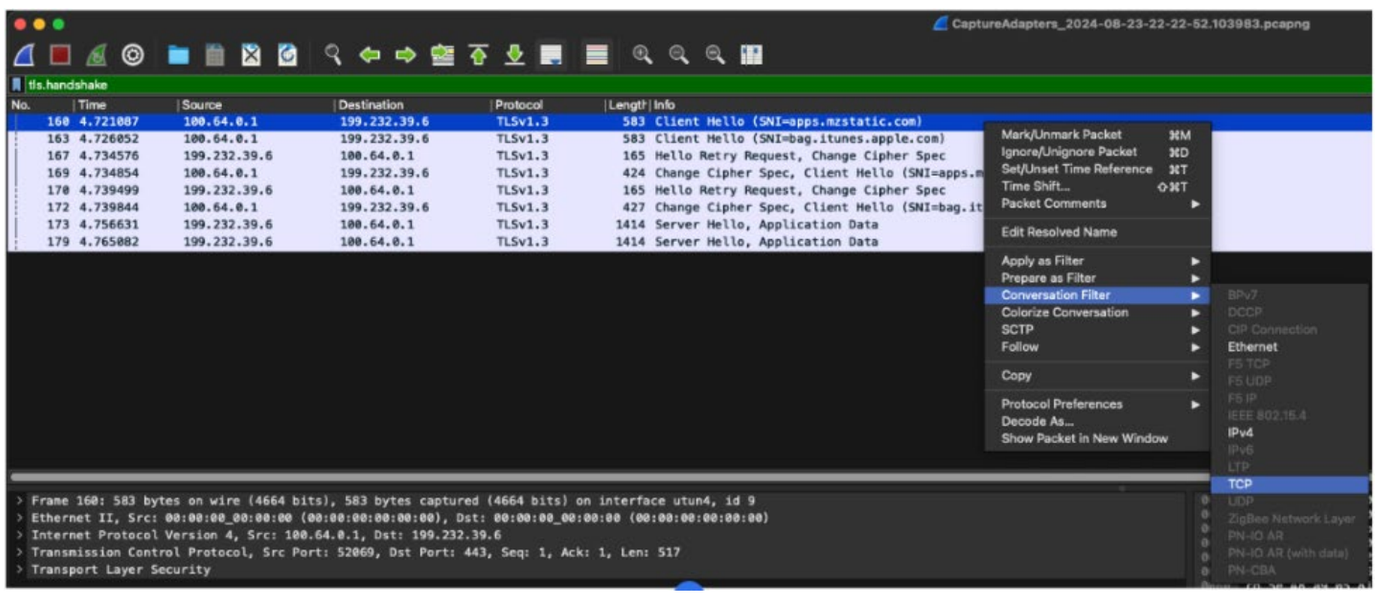
```
  #   decrypt_sub_actions {
  #     server_certificates       = "BLOCK"
  #     ocsp_check    = true
  #     block_ssl_traffic_with_no_sni_enabled = true
  #     min_client_tls_version    = "CLIENT_TLS_1_1"
  #     min_server_tls_version    = "SERVER_TLS_1_1"
  #     block_undecrypt           = true
  #     http2_enabled = false
  #   }
  # }
}


resource "zia_ssl_inspection_rules" "Github_Exemptions" {
  name           = "Github_Exemptions"
  description    = "Github_Exemptions"
  state          = "ENABLED"
  order          = 4
  rank           = 7
  road_warrior_for_kerberos = false
  platforms      = ["SCAN_IOS", "SCAN_ANDROID", "SCAN_MACOS", "SCAN_WINDOWS",
"NO_CLIENT_CONNECTOR", "SCAN_LINUX"]
  url_categories = [zia_url_categories.blockingCopilot.id]
  action {
    type = "DECRYPT"
    # show_eun       = false
    # show_eunatp    = false
    override_default_certificate = false

    decrypt_sub_actions {
      server_certificates       = "BLOCK"
      ocsp_check    = true
      block_ssl_traffic_with_no_sni_enabled = true
      min_client_tls_version    = "CLIENT_TLS_1_1"
      min_server_tls_version    = "SERVER_TLS_1_1"
      block_undecrypt           = true
      http2_enabled = false
    }
  }
}
```

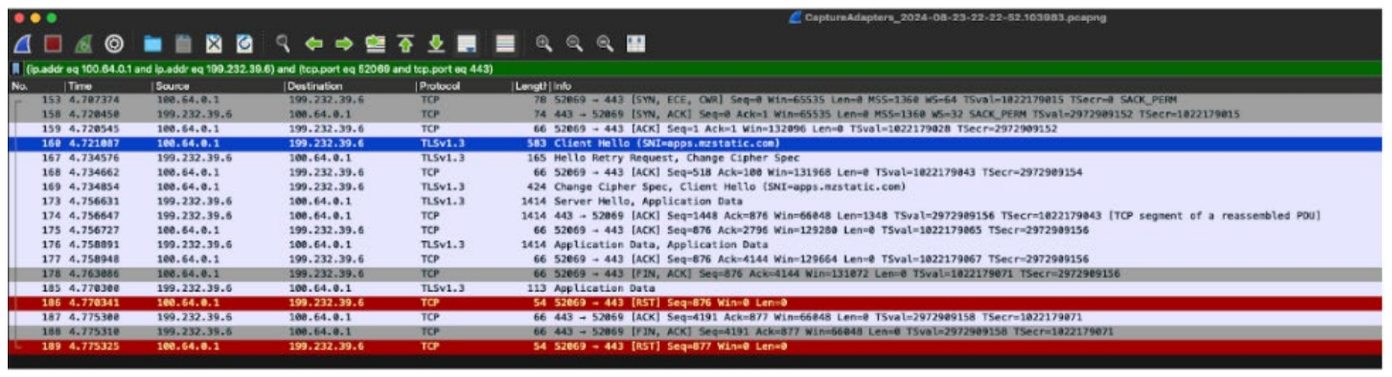
5.2.4 URL Filtering

This is the example rule to block Github Copilot only by URL.

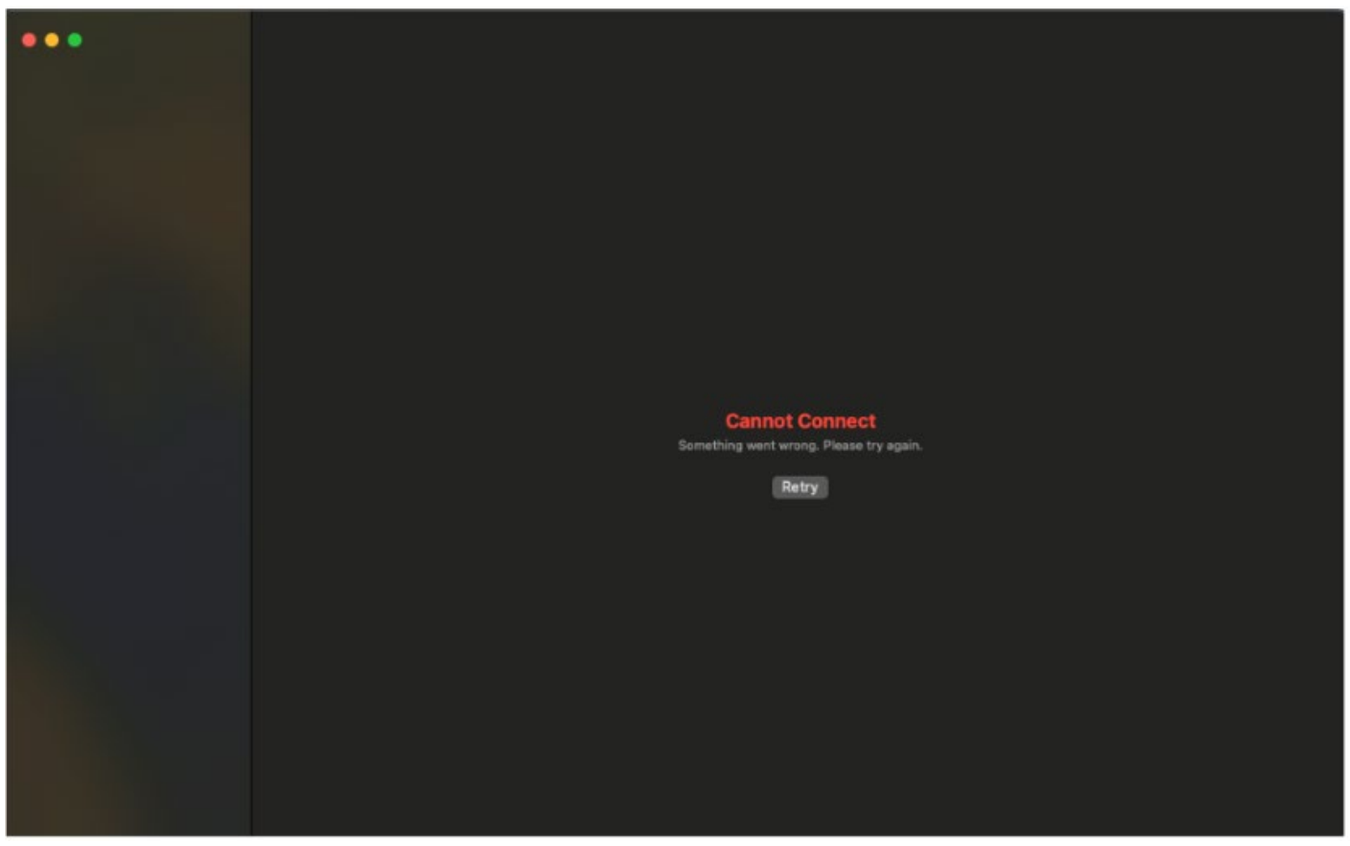
```
None
resource "zia_url_filtering_rules" "Block_Github_Copilot" {
  name        = "Block Github Copilot"
  description = "Block Github Copilot"
  state       = "ENABLED"
  action      = "BLOCK"
  order       = 3
  url_categories = [
    zia_url_categories.blockingCopilot.id
  ]
  device_trust_levels = ["UNKNOWN_DEVICETRUSTLEVEL", "LOW_TRUST", "MEDIUM_TRUST",
"HIGH_TRUST"]
  protocols           = ["ANY_RULE"]
  request_methods     = ["CONNECT", "DELETE", "GET", "HEAD", "OPTIONS", "OTHER",
"POST", "PUT", "TRACE"]
  block_override      = true
}
```

5.3 Example: Troubleshooting TLS with Wireshark

Step	Screenshot Reference
1. Start at the same spot as when the access to the App Store is broken.	
2. Open ZCC. Go to More > Troubleshoot. Clear Logs, then Start Packet Capture. Then hit the displayed “Retry” button from above. Stop Packet Capture and Export Logs.	
3. Unzip Log bundle, search for .pcapng file with today’s date & open in Wireshark.	
4. First look for TLS hand-shakes using the “tls.handshake” filter.	
5. Then use the Conversation Filter option to view a like-ly suspect as displayed.	

Step	Screenshot Reference
6. On the network trace, you can see after the server side transmits its hello information in packets 173, 174, & 176. Immediately following the ACK of packet 176 (packet 177), the client side (App Store) sends a RST in packet 178. This is what it looks like when the application terminates a session without finding an acceptable certificate (certificate pinning). In some cases an SSL Alert message packet is sent on the wire, but this is implementation dependent.	
7. In this case, both of these destinations (apps.mzstatic.com & bag.itunes.apple.com) display the same behavior shown using the mzstatic example above.	
8. Note, the ZSATunnel logs here show almost nothing from a web perspective as the web packets are routed directly to the tunnel.	

5.4 Example: Troubleshooting DNS with Wireshark

Step	Screenshot Reference
1. Start the Packet Capture in the Zscaler Client Connector.	

2. In another terminal window, and while the Packet Capture is already running, issue the following command to clear the DNS cache:

   `sudo dscacheutil -flushcache: sudo killall -HUP mDNSResponder`

3. Refresh the application, open the application, or do any similar action that forces the application to use the network. In our case we hit the Retry button.
4. Stop the Packet Capture and Export the logs to the local file system.
5. Unzip the file and look for a file with filename extension pcapng. Open this file in Wireshark. Once open, use the “dns” filter and look for the relevant entry that points to the application you are trying to use.

Note: The application developer could have used an acronym, a short name or some random cdn as the connection that is failing. In this case, the application provider is Apple and the IP is owned by Akamai Technologies. Therefore, look for a combination of Apple and Akamai in the domain name. The image to the right illustrates the DNS entry as a combined list of CNAME entries.

Step	Screenshot Reference

6. The DNS response is:

   ```
   Standard query response 0xc387 HTTPS downloaddispatch.itunes.apple.com CNAME
   downloaddispatch-cdn.itunes-apple.com.akadns.net CNAME downloaddispatch.
   itunes.apple.com.edgesuite.net CNAME a1988.dscapi6.akamai.net SOA n0dscapi6.
   akamai.net
   ```

7. Following the next DNS request in the trace will resolve a1988.dscap16.akamai.com to 23.55.204.23 This Akamai IP falls within the ASN 35994 (https://bgp.he.net/AS35994) so is 23.46.150.50. Name verification indicates that all other names are within this ASN.

   When exploring this DNS name on the exported logs from Zscaler Client Connector, you will find the Application Exception error for Client and Server connection close:

   ```
   2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to
   SME for host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443

   2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception
   in TcpConnection run() (Error=Application exception: Both client and server
   sockets are closed!

   2024-07-17 18:16:30.054553(-0400)[80582:16211546] INF ===> ID=492535293,
   ~ZTCPServerConnection state=closed ServerConnections=1 clt_bytes=891, srv_
   bytes=4230!

   2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to
   SME for host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443

   2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception
   in TcpConnection run() (Error=Application exception: Both client and server
   sockets are closed!)

   2024-07-17 18:16:30.054553(-0400)[80582:16211546] INF ===> ID=492535293,
   ~ZTCPServerConnection state=closed ServerConnections=1 clt_bytes=891, srv_
   bytes=4230!
   ```

8. In conclusion, the first entry shows the URL for the connection and the string `Error=Application exception: Both client and server sockets are closed!` was exactly what we needed to make sure this DNS name requires a corresponding bypass rule to be created in ZIA.

5.5 PKI and Certificate Authorities

In order for Zscaler to verify SSL traffic, the client operating system (from which an SSL connection is established to a target server FQDN on the Internet), must trust the root and intermediate CA certificate used by Zscaler.

It is possible to create and use your own root and intermediate CA certificate or to use the one provided by Zscaler.

If a Custom Root or Intermediate certificate is placed in the SSL Inspection section of the ZIA Admin Portal, and this certificate is trusted in the certificate store of the operating system, then all applications that access the certificate store of the operating system trust this certificate and any certificates issued by the Root or Intermediate certificate.

In managed environments, a certification authority usually already exists and a suitable certificate has already been stored in ZIA. If not, the following sections describe various options for creating a corresponding root and intermediate certification authority certificate that can be used.

When installing a custom certificate for SSL inspection on the ZIA portal, there are multiple methods to acquire the certificate. The following sections provide more information on each of these methods.

- Windows Domain with Certificate Authority
- Certificates without Windows Certificate Authority
- Create a new Root CA Certificate with PowerShell
- Create a new Intermediate CA Certificate with PowerShell
- Create a new Intermediate CA Certificate with PowerShell and OpenSSL
- PKI and Customer Provided Certificate Authority
- ZIA TLS Interception Certificate Request
- Create an Intermediate Certificate at Active Directory Certificate Services
- Create a Intermediate Certificate with OpenSSL

5.5.1 Windows Active Directory (AD) with Certificate Services (ADCS)

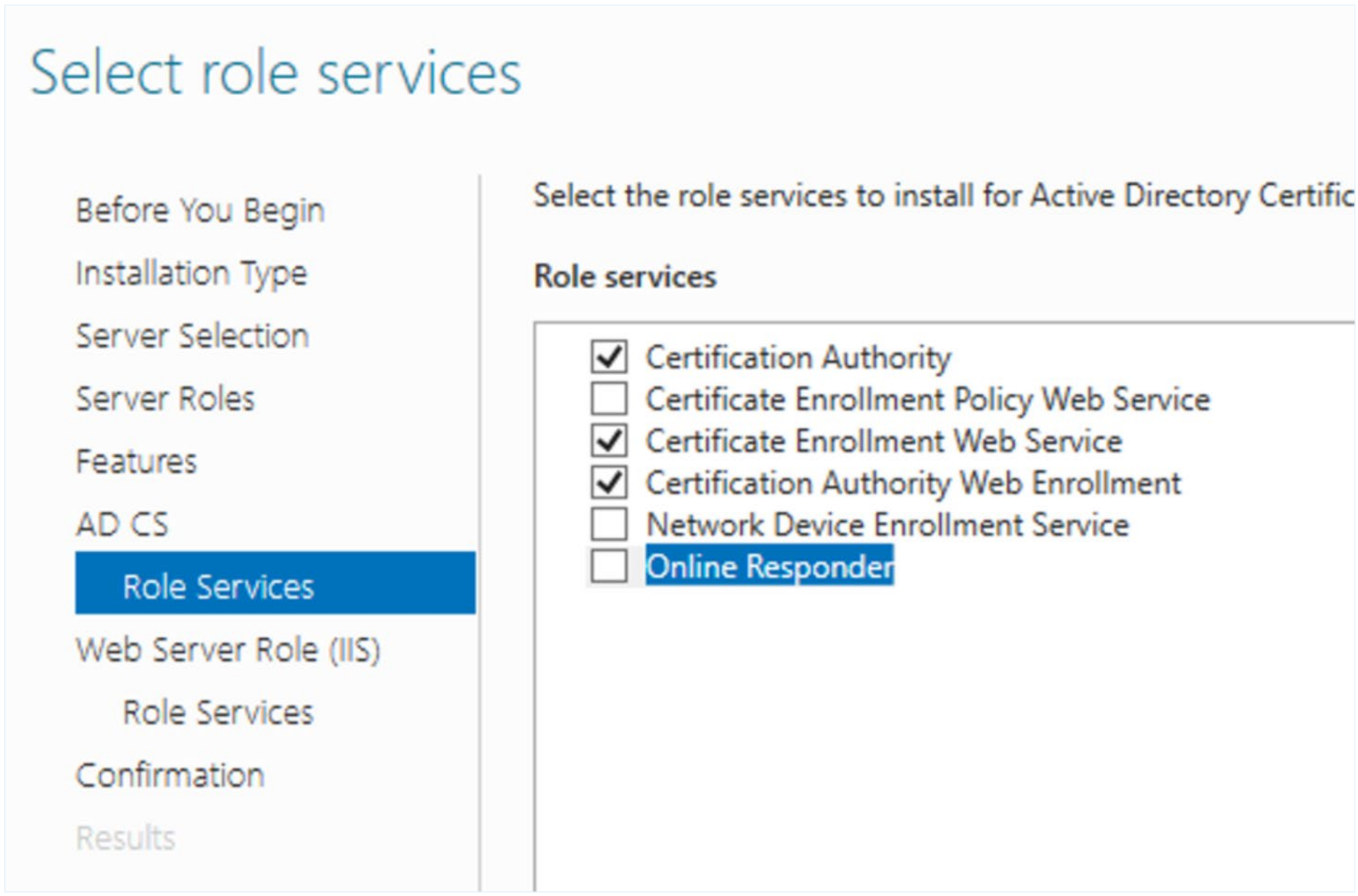
A Windows Certificate Authority can be deployed on a member server or domain controller of an Active Domain or on a standalone server. If the certificate authority is installed on a domain controller or member server of an Active Directory domain, the root certificate authority certificate is automatically distributed to all member servers and computers via group policy. If it is a standalone server, the root certification authority certificate must be published manually in the Active Directory domain.

The rest of this section shows how to configure AD and ADCS and then how to install an Enrollment Certificate in ZPA. Follow along with the screenshots below.

5.5.1 Windows Active Directory (AD) with Certificate Services (ADCS)

A Windows Certificate Authority can be deployed on a member server or domain controller of an Active Domain or on a standalone server. If the certificate authority is installed on a domain controller or member server of an Active Directory domain, the root certificate authority certificate is automatically distributed to all member servers and computers via group policy. If it is a standalone server, the root certification authority certificate must be published manually in the Active Directory domain.

The rest of this section shows how to configure AD and ADCS and then how to install an Enrollment Certificate in ZPA. Follow along with the screenshots below.



CA Name

DESTINATION SERVER
ad-test.domain.danielcastro.info

- Credentials
- Role Services
- Setup Type
- CA Type
- Private Key
  - Cryptography
  - CA Name
  - Validity Period
- Certificate Database
- Confirmation
- Progress
- Results

Specify the name of the CA

Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA:

domain-ad-test-CA

Distinguished name suffix:

DC=domain,DC=danielcastro,DC=info

Preview of distinguished name:

CN=domain-ad-test-CA,DC=domain,DC=danielcastro,DC=info

More about CA Name

Cryptography for CA

DESTINATION SERVER
ad-test.domain.danielcastro.info

- Credentials
- Role Services
- Setup Type
- CA Type
- Private Key
  - Cryptography
  - CA Name
  - Validity Period
- Certificate Database
- Confirmation
- Progress
- Results

Specify the cryptographic options

Select a cryptographic provider:

RSA#Microsoft Software Key Storage Provider

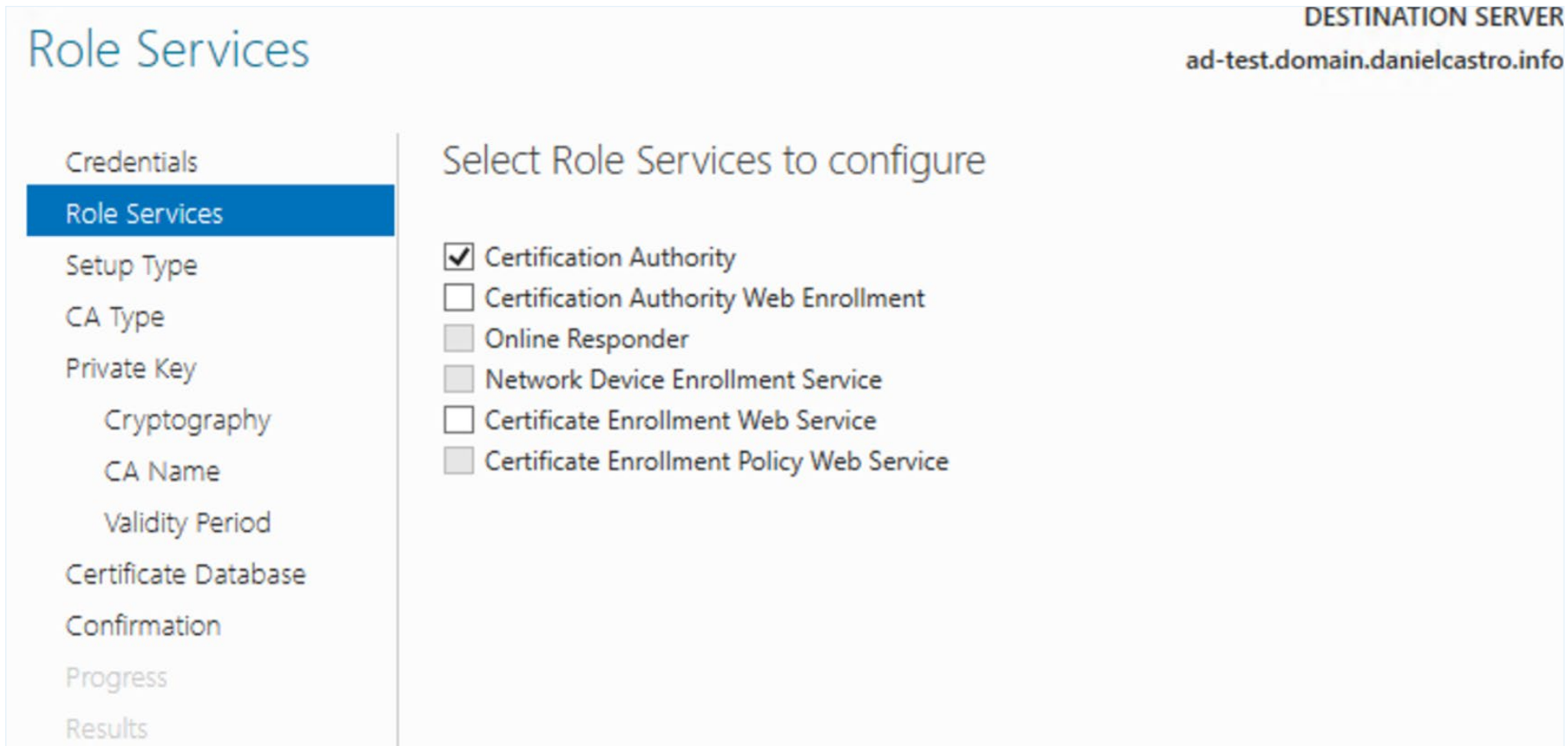
Key length:

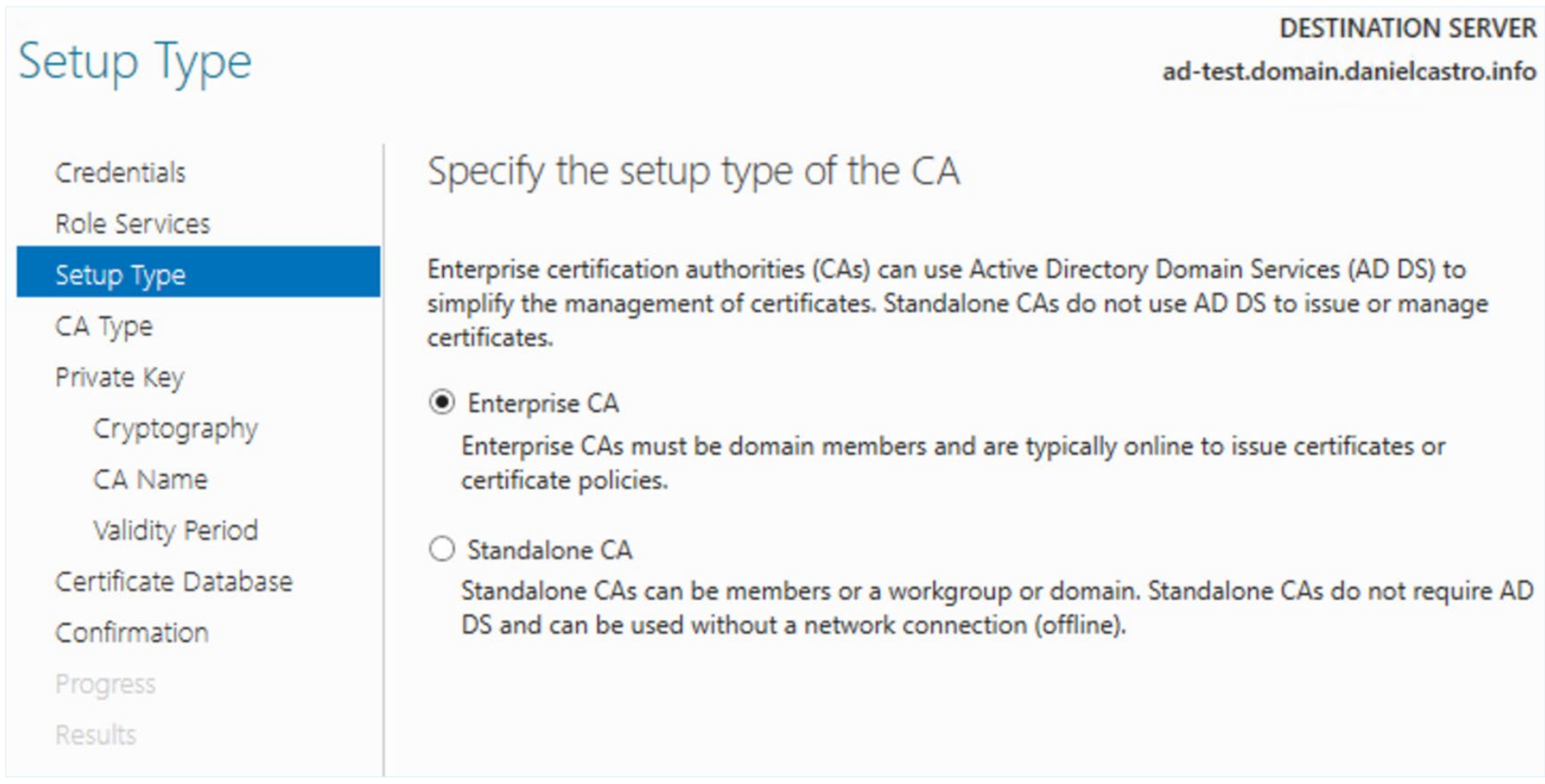
2048

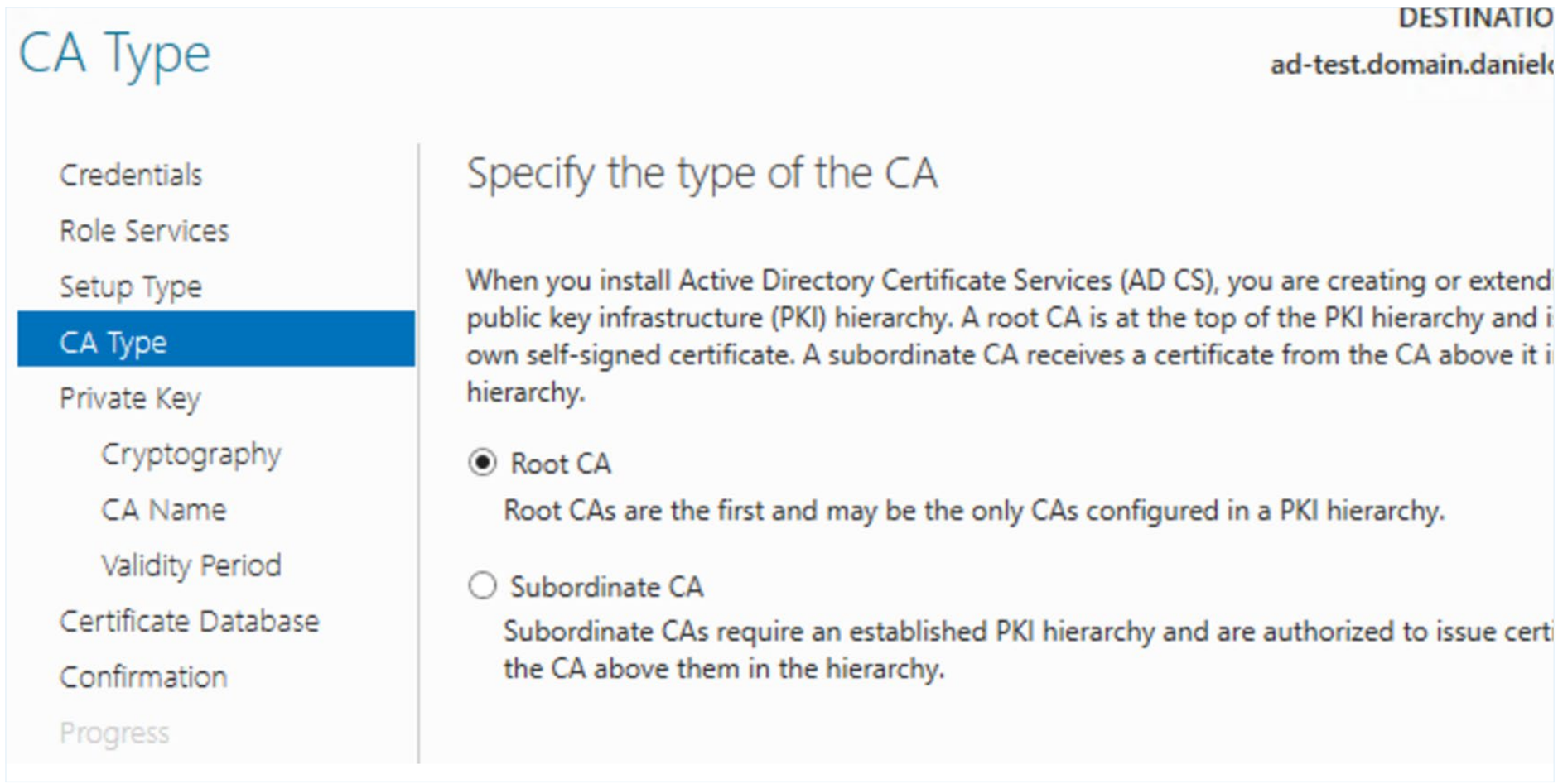
Select the hash algorithm for signing certificates issued by this CA:

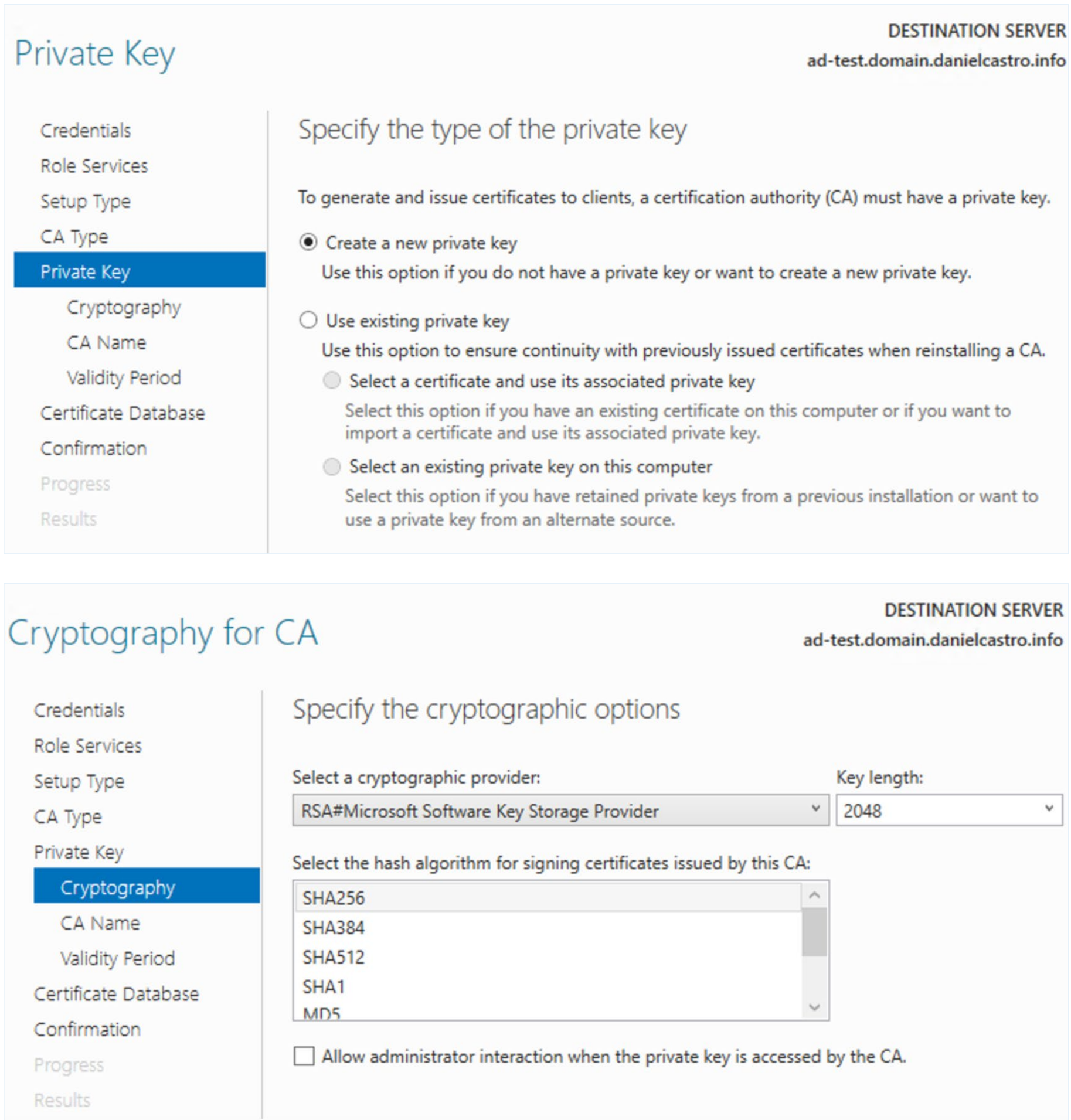
SHA256
SHA384
SHA512
SHA1
MD5

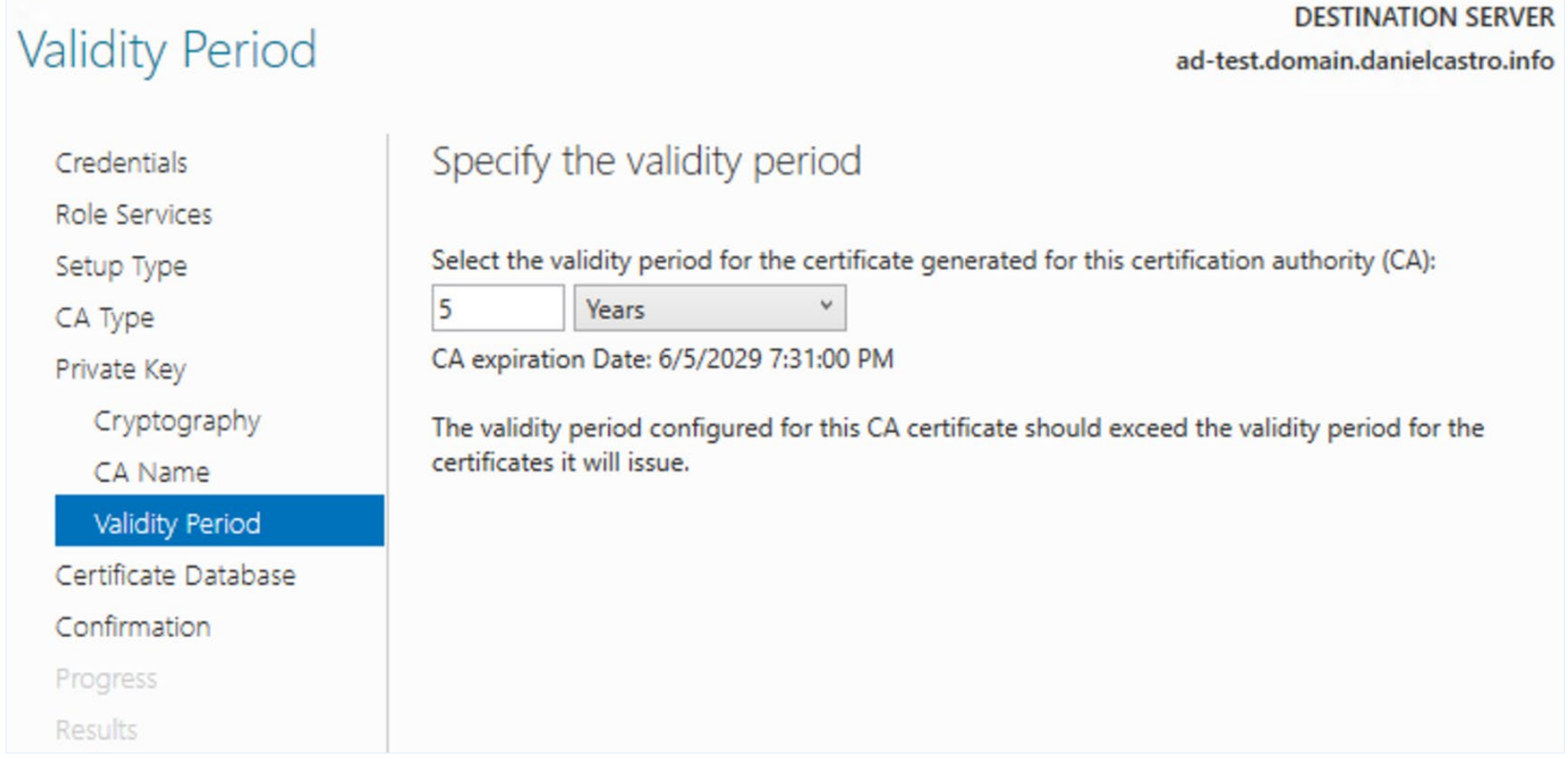
☐ Allow administrator interaction when the private key is accessed by the CA.











Specify the validity period

Select the validity period for the certificate generated for this certification authority (CA):

15 Years

CA expiration Date: 6/5/2039 7:33:00 PM

The validity period configured for this CA certificate should exceed the validity period for the certificates it will issue.

Results

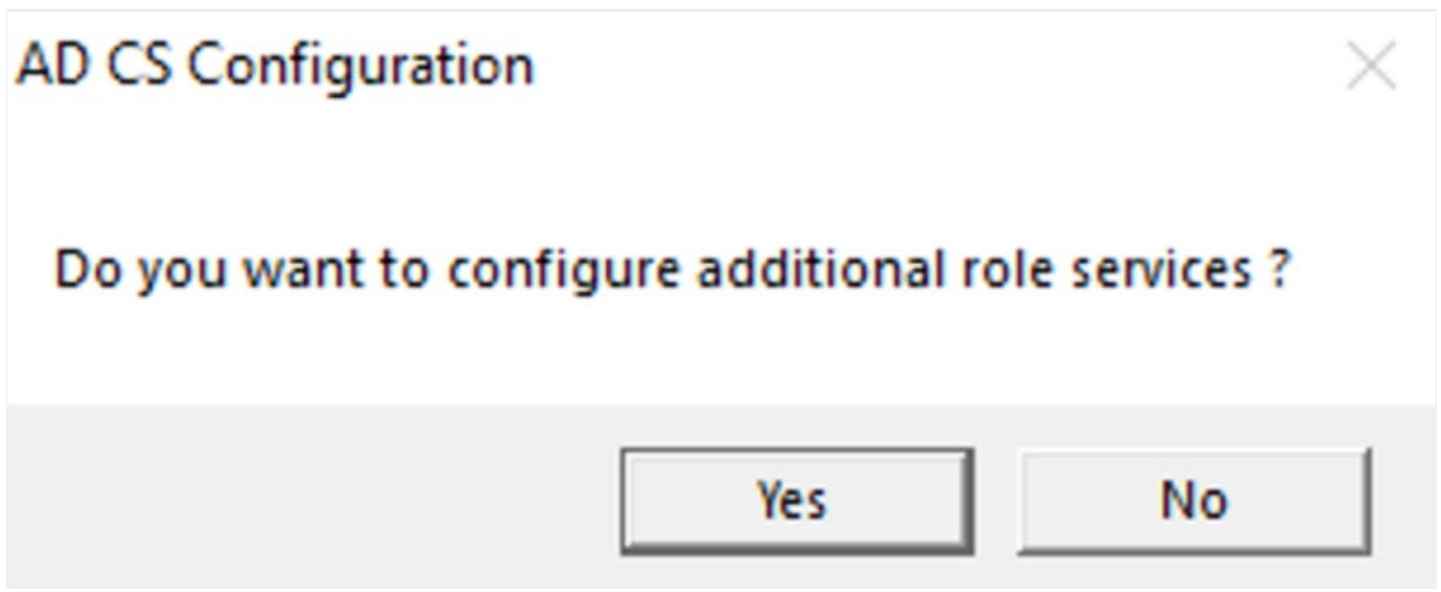
DESTINATION SERVER
ad-test.domain.danielcastro.info

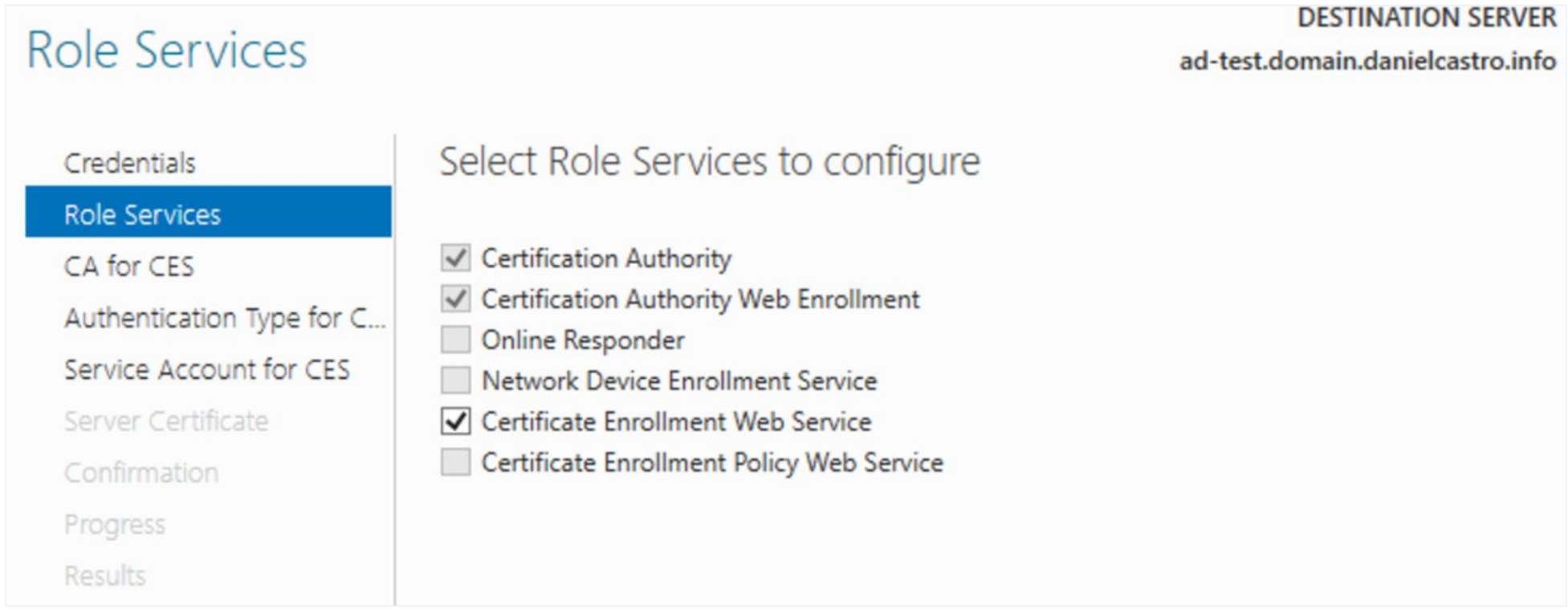
Credentials
Role Services
Confirmation
Progress
Results

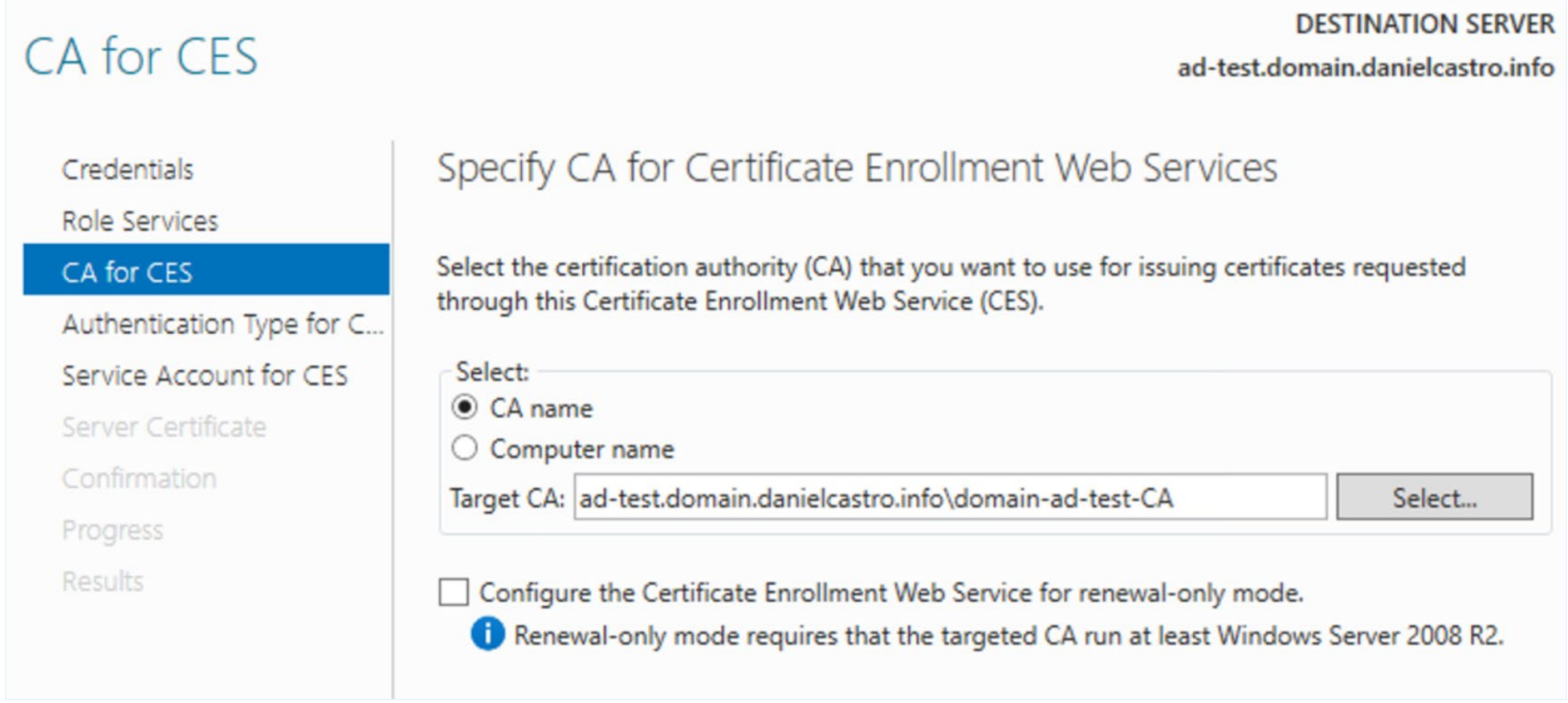
The following roles, role services, or features were configured:

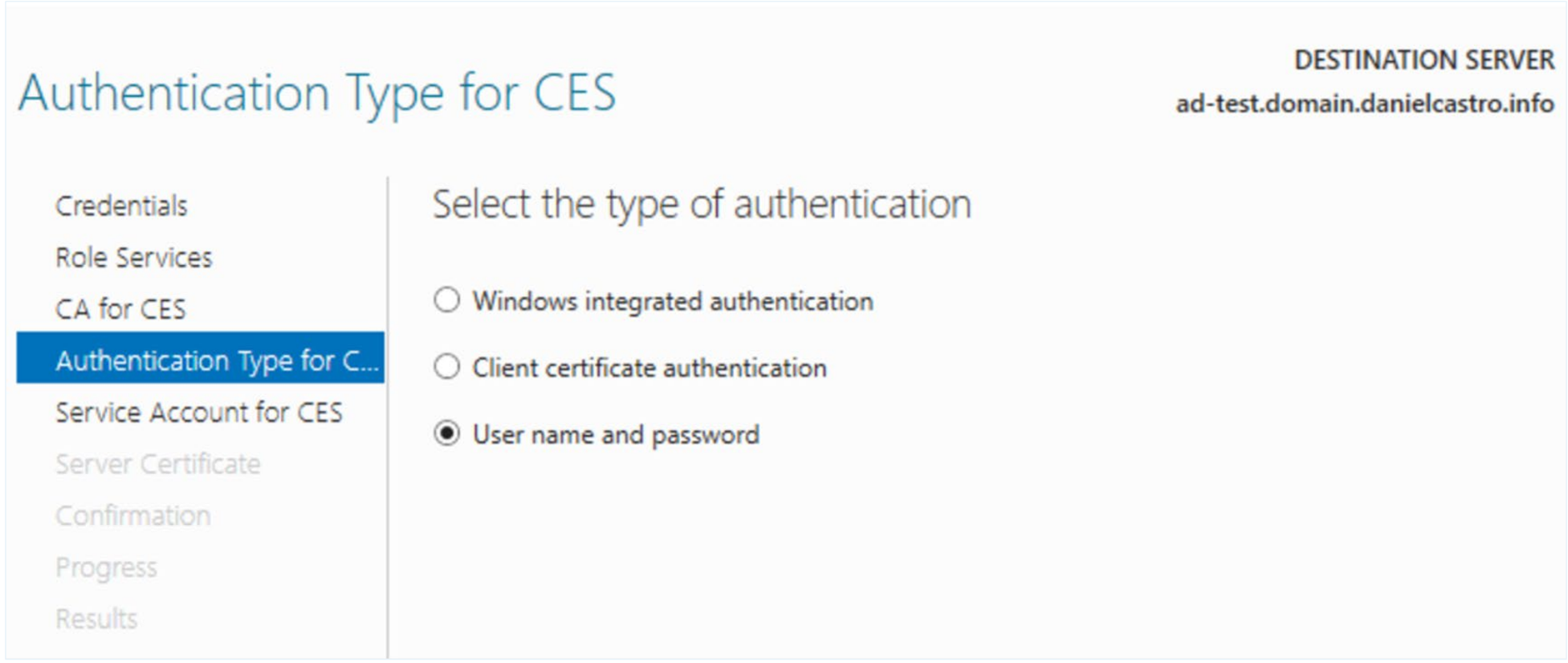
Active Directory Certificate Services

Certification Authority Web Enrollment — **Configuration succeeded**
More about Web Enrollment Configuration









Service Account for CES

DESTINATION SERVER
ad-test.domain.danielcastro.info

- Credentials
- Role Services
- CA for CES
- Authentication Type for C...
- Service Account for CES
- Server Certificate
- Confirmation
- Progress
- Results

Specify the service account

Select the identity that the Certificate Enrollment Web Service (CES) uses when communicating with the certification authority (CA) and other services on the network.

○ Specify service account (recommended)

The account selected must be a member of the IIS_IUSRS group. If Kerberos is selected as the authentication type, a service principal name is required for the service account.

[ ] [Select...]

◉ Use the built-in application pool identity

Server Certificate

DESTINATION SERVER
ad-test.domain.danielcastro.info

- Credentials
- Role Services
- CA for CES
- Authentication Type for C...
- Service Account for CES
- Server Certificate
- Confirmation
- Progress
- Results

Specify a Server Authentication Certificate

When communicating with clients, the web service(s) uses Secure Sockets Layer (SSL) protocol to encrypt network traffic.

◉ Choose an existing certificate for SSL encryption (recommended)

Issued To	Issued By	Expiration Date
domain-ad-test-CA	domain-ad-test-CA	6/5/2039

[Properties] [Refresh]

○ Choose and assign a certificate for SSL later

⚠ For this role service to function, you must configure this server with a valid certificate.

Confirmation

DESTINATION SERVER
ad-test.domain.danielcastro.info

- Credentials
- Role Services
- CA for CES
- Authentication Type for C...
- Service Account for CES
- Server Certificate
- Confirmation
- Progress
- Results

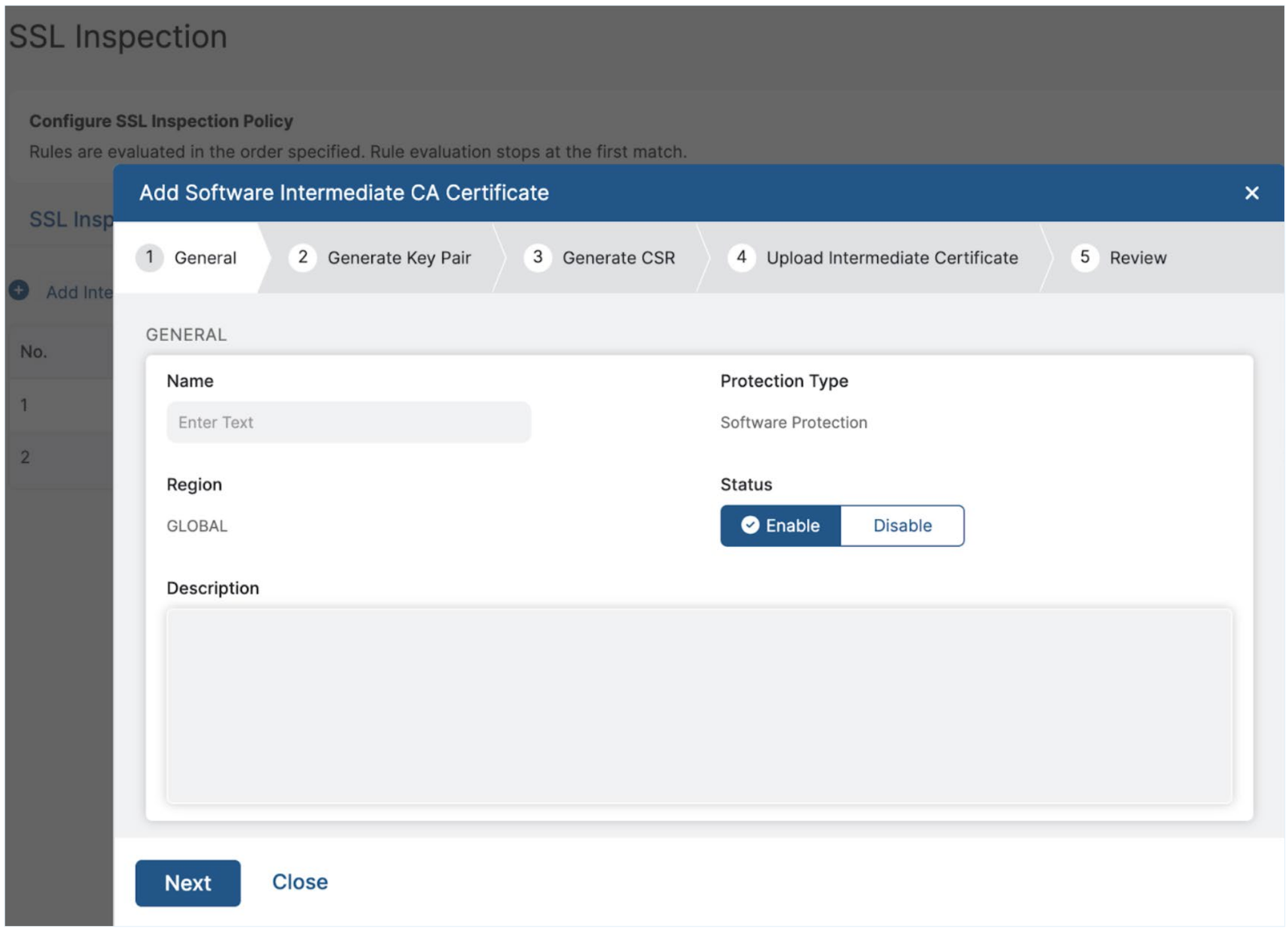
To configure the following roles, role services, or features, click Configure.

Active Directory Certificate Services

Certificate Enrollment Web Service

CA Name:	ad-test.domain.danielcastro.info\domain-ad-test-CA
Renewal Only Mode:	False
Authentication Type:	Username and password authentication
Allow Key-based Renewal:	False
Account:	Application Pool Identity
Server Authentication Certificate:	C665D67CB1551C05D49F7E1959C9B6173ACDE88A

At this point, ADCS has been successfully installed and we can proceed to open the browser and explore the CA services. But first head, over to either ZIA or ZPA and request a new CSR:



5.5.2 Certificates without Windows Certificate Authority

In addition to creating an intermediate certification authority certificate as described in the previous section, it is also possible, for testing purposes, to create corresponding certificates on a Windows client computer. It should be noted that certificates created in this way are only available on the local computer. For use on other computers, both the root and intermediate certification authority certificates must be exported and imported on the other computers.

5.5.3 Create a new Root CA Certificate with PowerShell

When ADCS is ready, create a new root CA certificate \ with the following PowerShell cmdlet:

```
PS C:\> $rootCert = New-SelfSignedCertificate -CertStoreLocation
Cert:\CurrentUser\My -DnsName "RootCA" -TextExtension
@("2.5.29.19={text}CA=true") -KeyUsage CertSign,CrlSign,DigitalSignature
```

The root CA certificate (without the private key) can then be exported as a file using the following command:

```
PS C:\> [String]$rootCertPath = Join-Path -Path 'cert:\CurrentUser\My\'
-ChildPath "$($rootCert.Thumbprint)"
PS C:\> Export-Certificate -Cert $rootCertPath -FilePath 'RootCA.crt'
```

5.5.4 Create a new Intermediate CA Certificate with PowerShell

A new intermediate certification authority certificate, which is signed with the previously created root CA certificate, can then be created with the following command:

```
PS C:\> $intermediateCACert = New-SelfSignedCertificate -CertStoreLocation
Cert:\LocalMachine\My -DnsName "IntermediateCA" -TextExtension
@("2.5.29.19={text}CA=true") -KeyLength 2048 -KeyUsage
CertSign,CrlSign,DigitalSignature -Signer $rootCert
```

The Intermediate CA certificate (without the private key) can then be exported as a file using the following commands:

```
PS C:\> [String]$intermediateCertPath = Join-Path -Path 'cert:\LocalMachine\My\'
-ChildPath "$($intermediateCACert.Thumbprint)"
PS C:\> Export-Certificate -Cert $intermediateCertPath -FilePath
'IntermediateCA.crt'
```

5.5.5 Create a new Intermediate CA Certificate with PowerShell and OpenSSL

In this section, we will show you how to create a root CA from which OpenSSL can create additional certificates or can be uploaded to ZIA for TLS interception.

Create a configuration file for the root CA. We will call this rootCA_OpenSSL.conf. Modify the folder locations to match your configuration.

```
[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
# The root key and root certificate.
private_key = root.key
certificate = root.pem
new_certs_dir = /Users/danielcastro/Downloads
```

```
database = index.txt
policy = policy_strict
dir = “~/CA”
certs = $dir/certs
crl_dir = $dir/crl
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/.rand

[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName = optional
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range
of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
distinguished_name = req_distinguished_name
extensions = v3_ca
req_extensions = v3_ca

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
```

```
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = “OpenSSL Generated Client Certificate”
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation,
digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = “OpenSSL Generated Server Certificate”
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always

[ req_distinguished_name ]
countryName = Country Name (2 letter code)countryName_default =
CA
countryName_min = 2
countryName_max = 2
organizationName = Zscaler Private CA
organizationName_default = Zscaler Inc.
0.organizationName = Zscaler Private
organizationalUnitName = Pro Serv
commonName = ca.danielcastro.info
emailAddress = dcastro@zscaler.com
```

Define the serial number of the generated certificates with this command:

```
> echo 1000 > serial
```

Proceed to create the ca private key, and certificate with:

```
> OpenSSL req -config rootCA_OpenSSL.conf -key root.key -new -x509
-days 7300 -sha256 -extensions v3_ca -out root.pem
```

Now, in the same folder, you will have root.pem, root.key and a configuration file. With these three files you can create server certificates, intermediate certificates, and many other types of cryptographic material.

5.6.6 PKI and Customer Provided Certificate Authority

To provide an Intermediate Certificate in ZIA when the organization uses their own enterprise Certificate Authority (CA), it is necessary to understand Public Key Infrastructure (PKI).

Public Key Infrastructure is a system of processes, technologies, and policies that allows you to encrypt and/or sign data. With PKI, you can issue digital certificates that authenticate the identity of users, devices, or services. These certificates work for both public web pages and private internal services. In the enterprise, this is usually a web service or software that is run as part of the enterprise infrastructure, and is typically called the Certificate Authority (CA).

In the context of this whitepaper, we will refer to the following objects and operations:

- Public Key is a number that is inside a certificate. When applied to data, it will encrypt the data but the data can only be decrypted with the Private Key.

- Private Key is a file (inside there is a string that represents a number) that was created and will allow it to decrypt any data that is encrypted using the Public Key of the certificate. This file must be stored in a safe place.

- Certificate Sign Request (CSR) is a file that describes the desired certificate that is being requested.

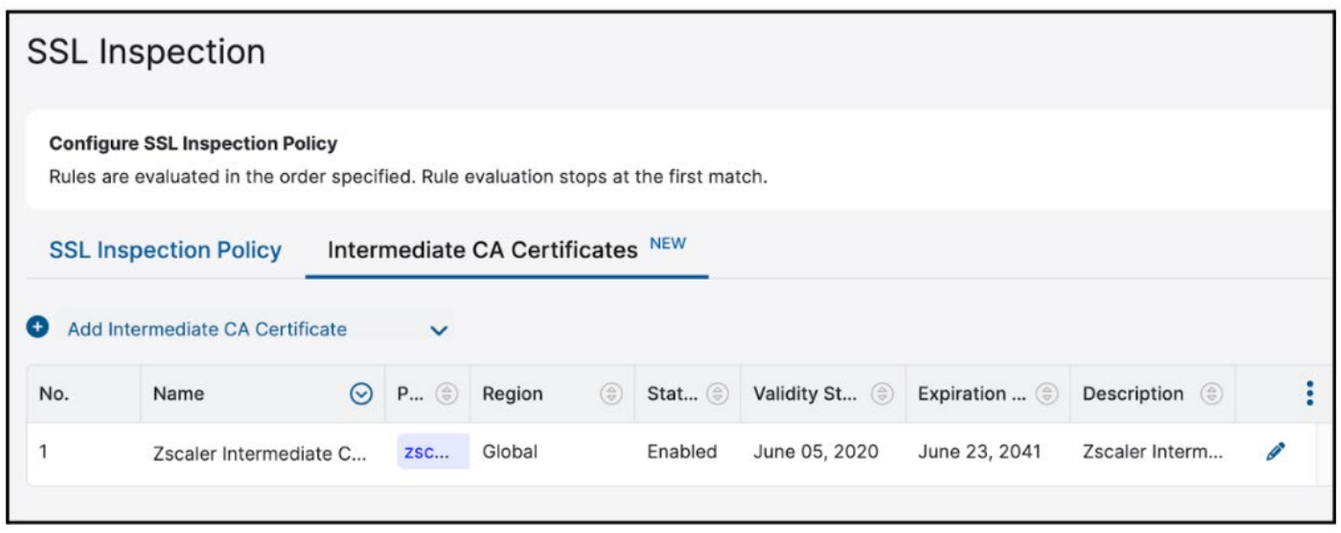
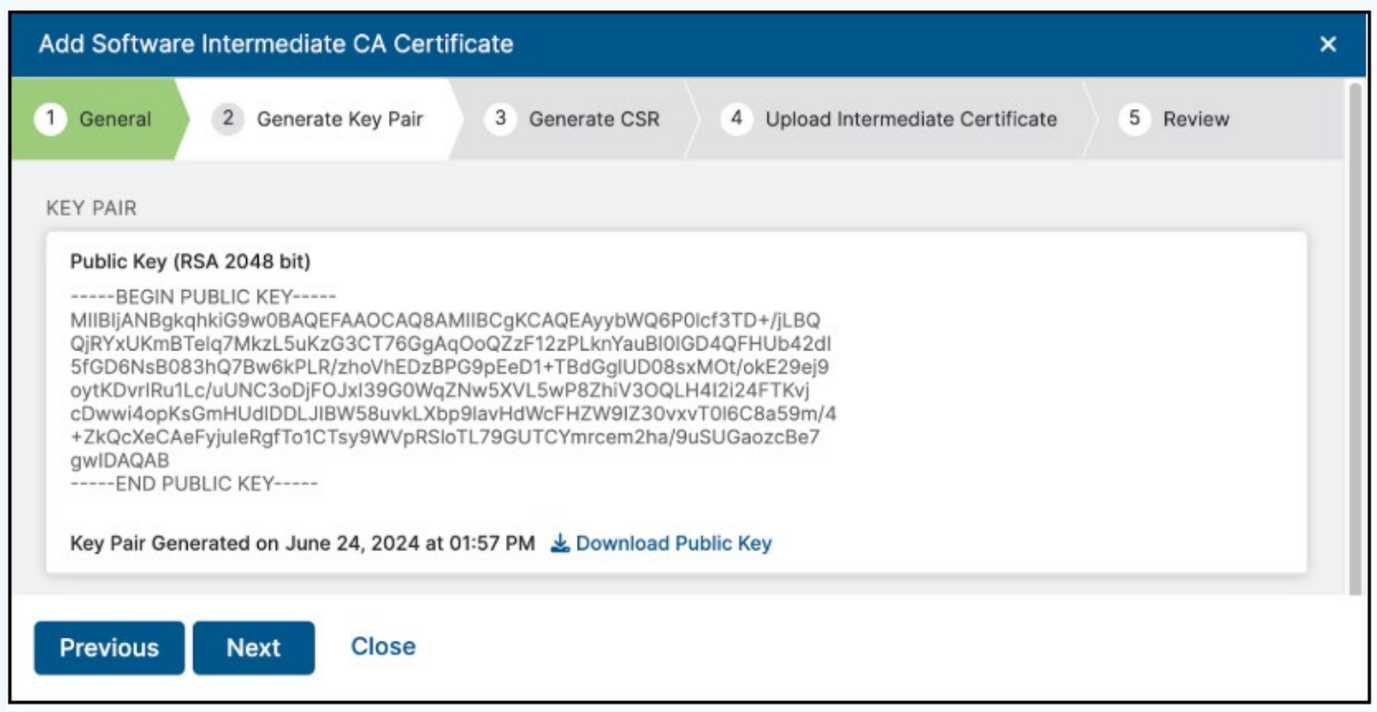
- Signing is the process of taking in a CSR to the CA and asking for a certificate.

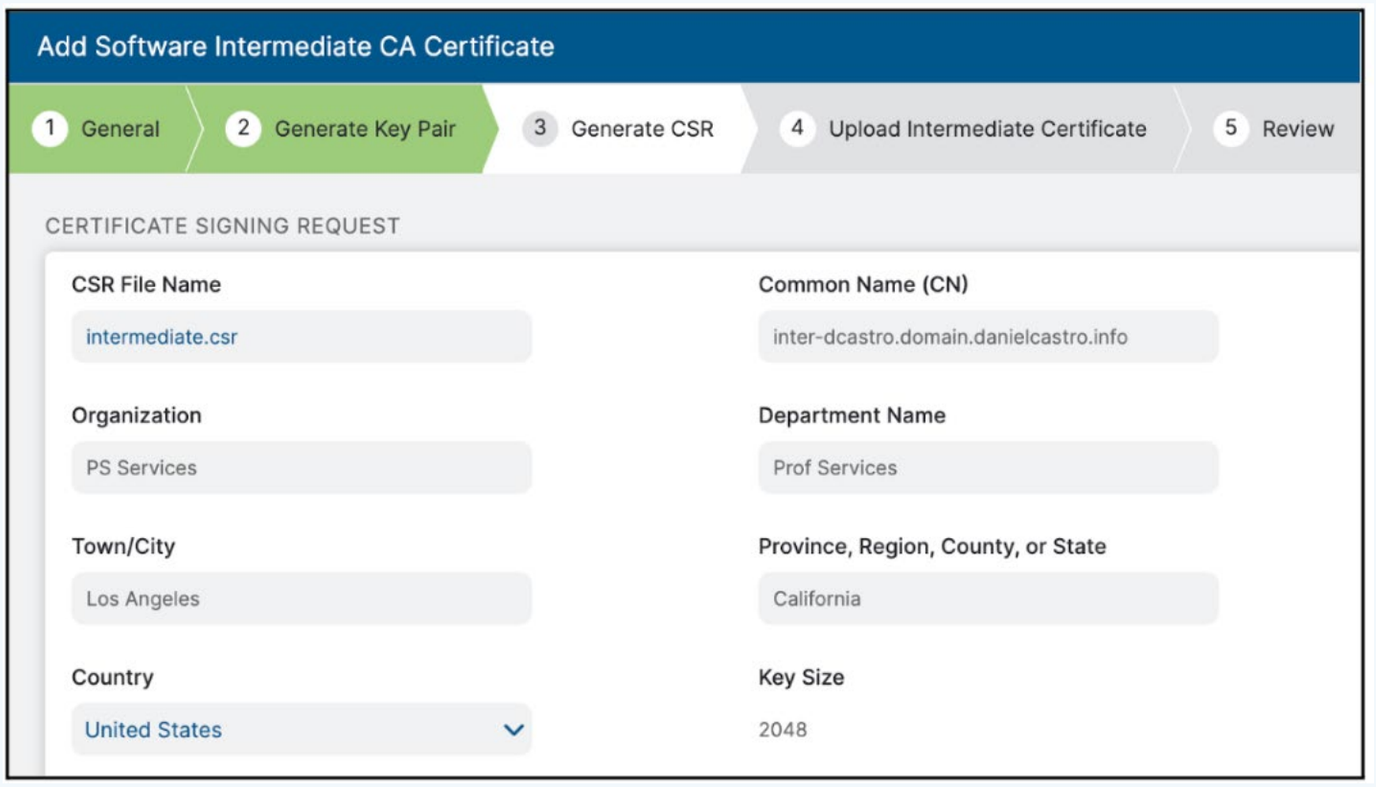
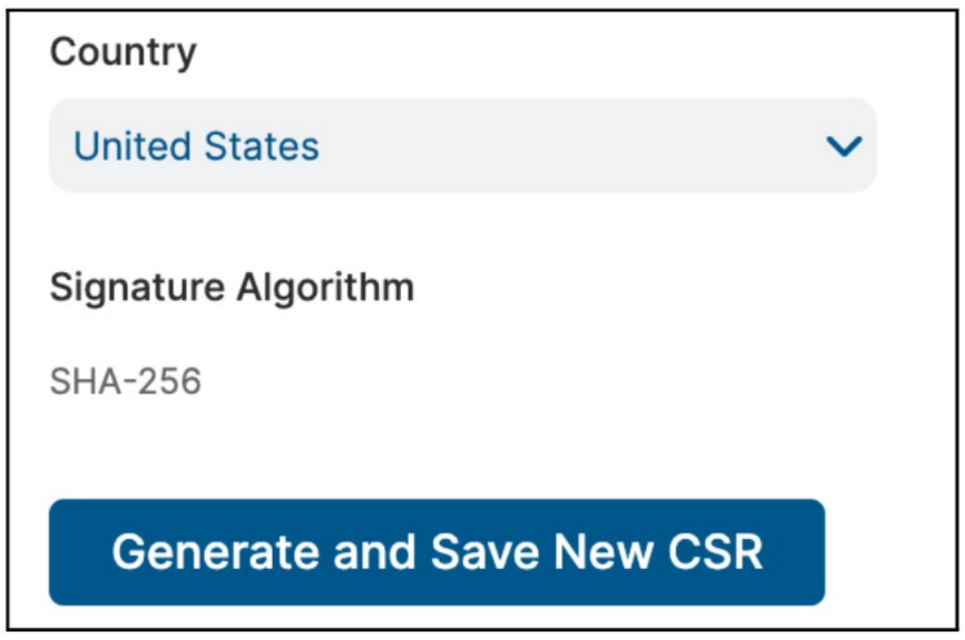
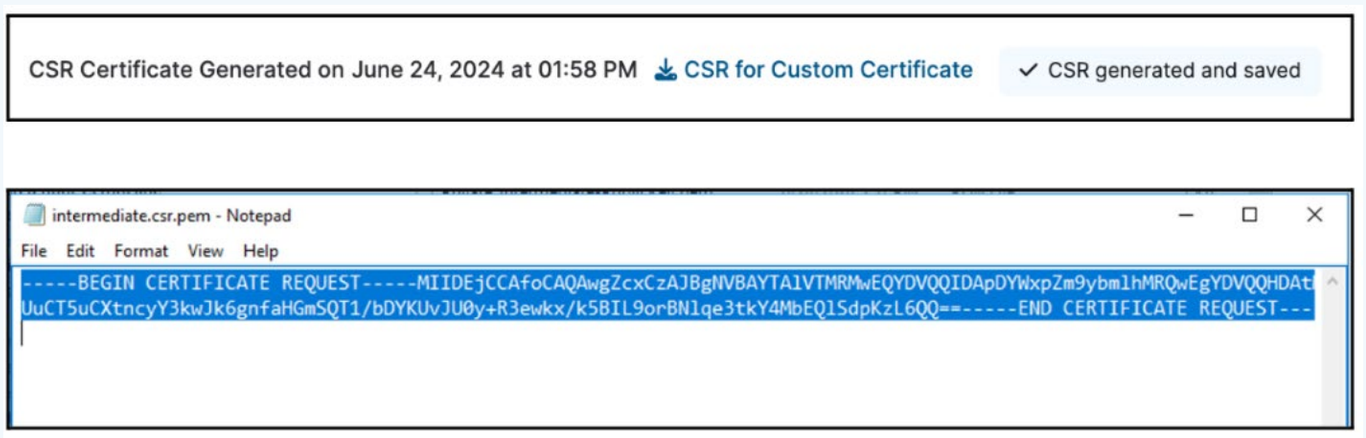
- Custom Intermediate Certificate is a certificate (used in both ZIA and ZPA but cannot be the same) that allows operations as if it were a Certificate Authority, becoming a delegate of a higher tier Certificate Authority. ZIA uses this for TLS Interception and ZPA uses this for certificate enrollment.

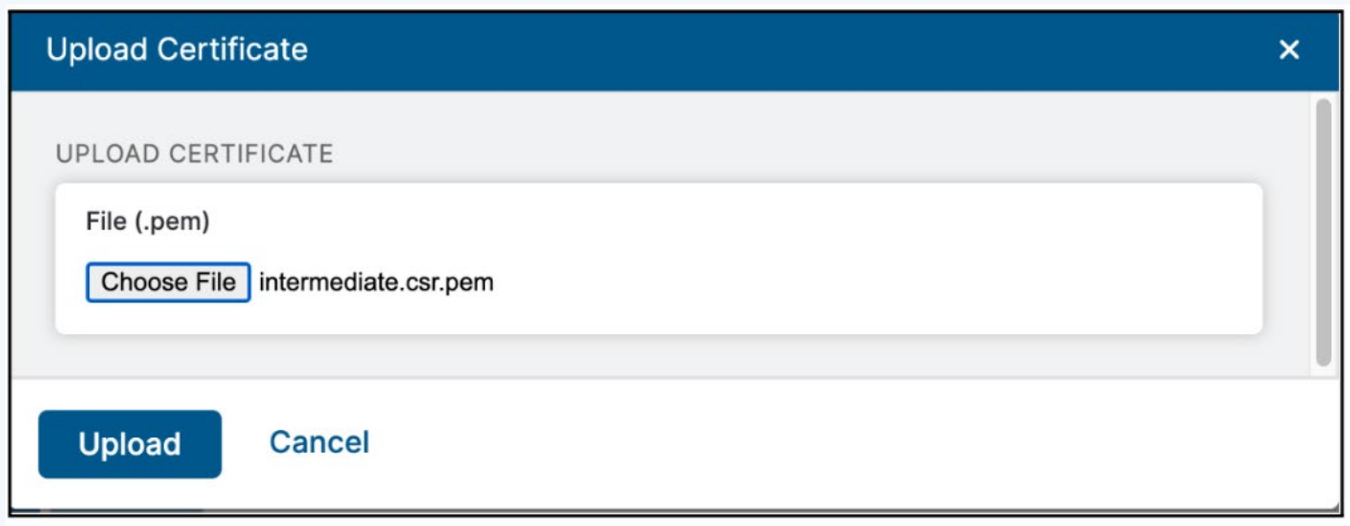
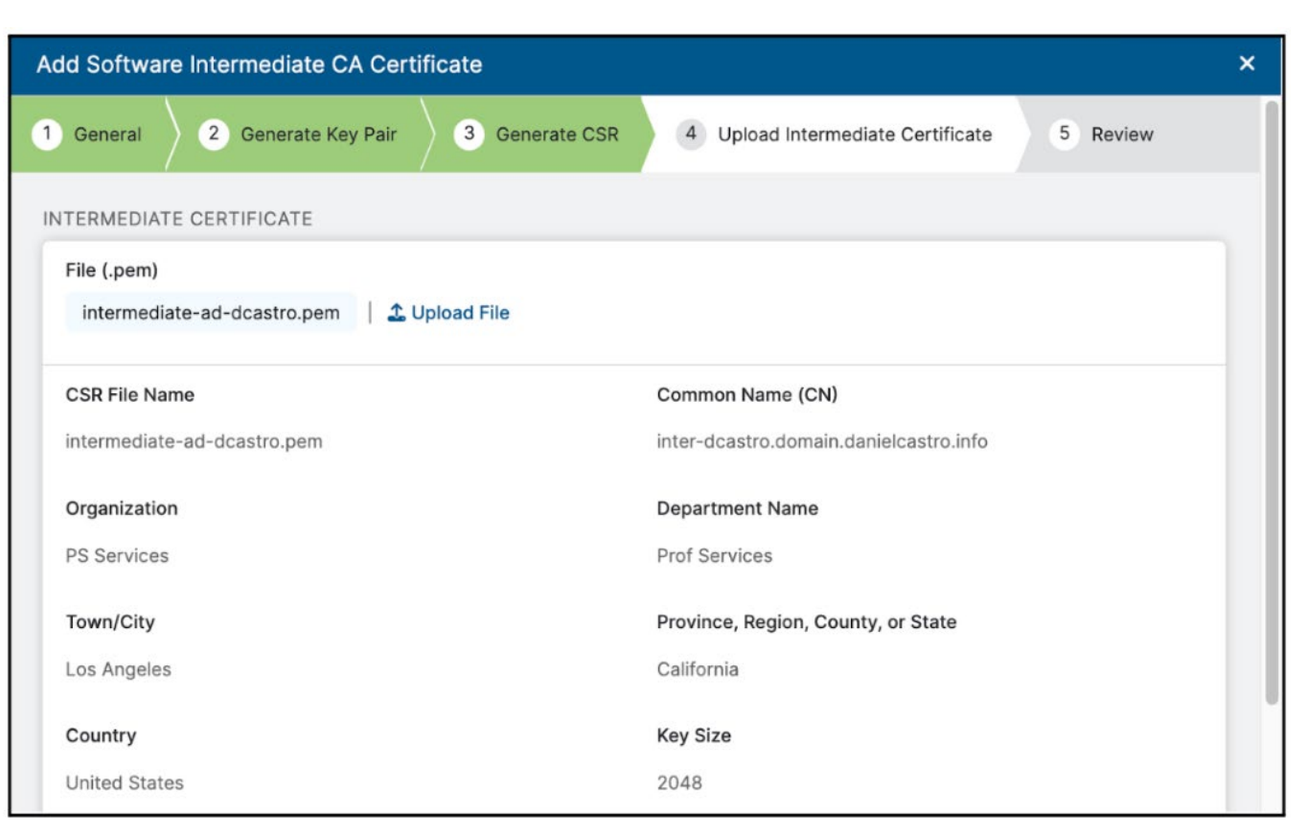
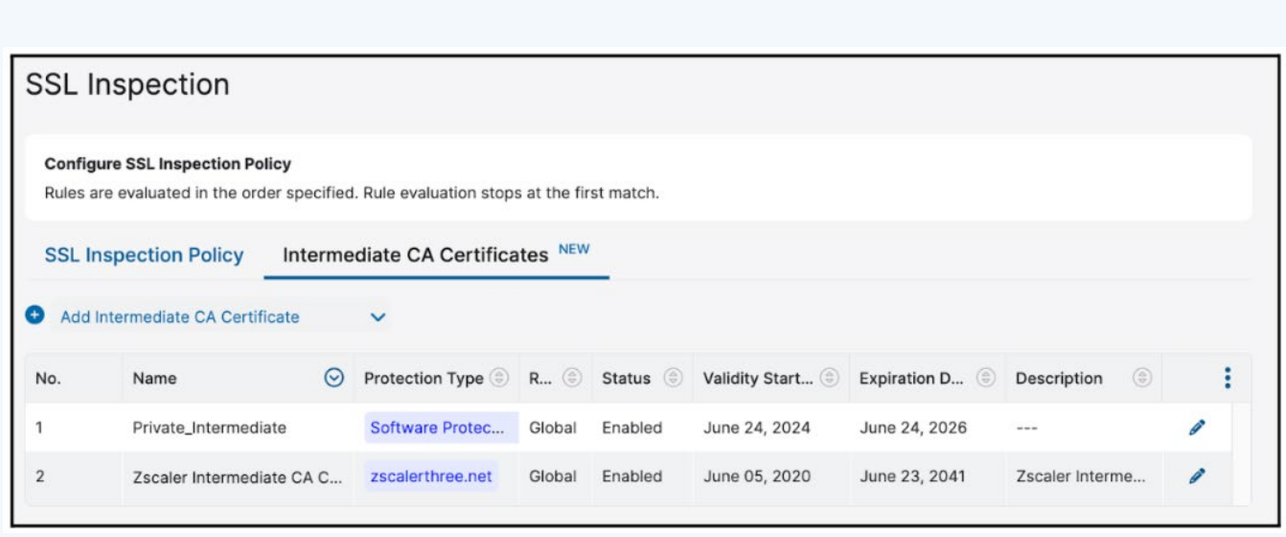
In order to make a secure connection, a client must have a trust relationship to the Certificate Authority that issued the certificate provided by the server. That inherited trust enables secure communications. Every device has a list of trusted certificate authorities (CA) that are provided by the manufacturer of the device. Installing an additional CA is usually documented by the manufacturer or software provider so that the device can trust any secure connections that make use of certificates under that CA. Furthermore, most individual software components provide a method by which insecure (untrusted) communication can be done. For example, in python's urllib3 a specific configuration needs to be done when sending HTTP requests to a server with an untrusted certificate, but this configuration does not enable trust between the client and server.

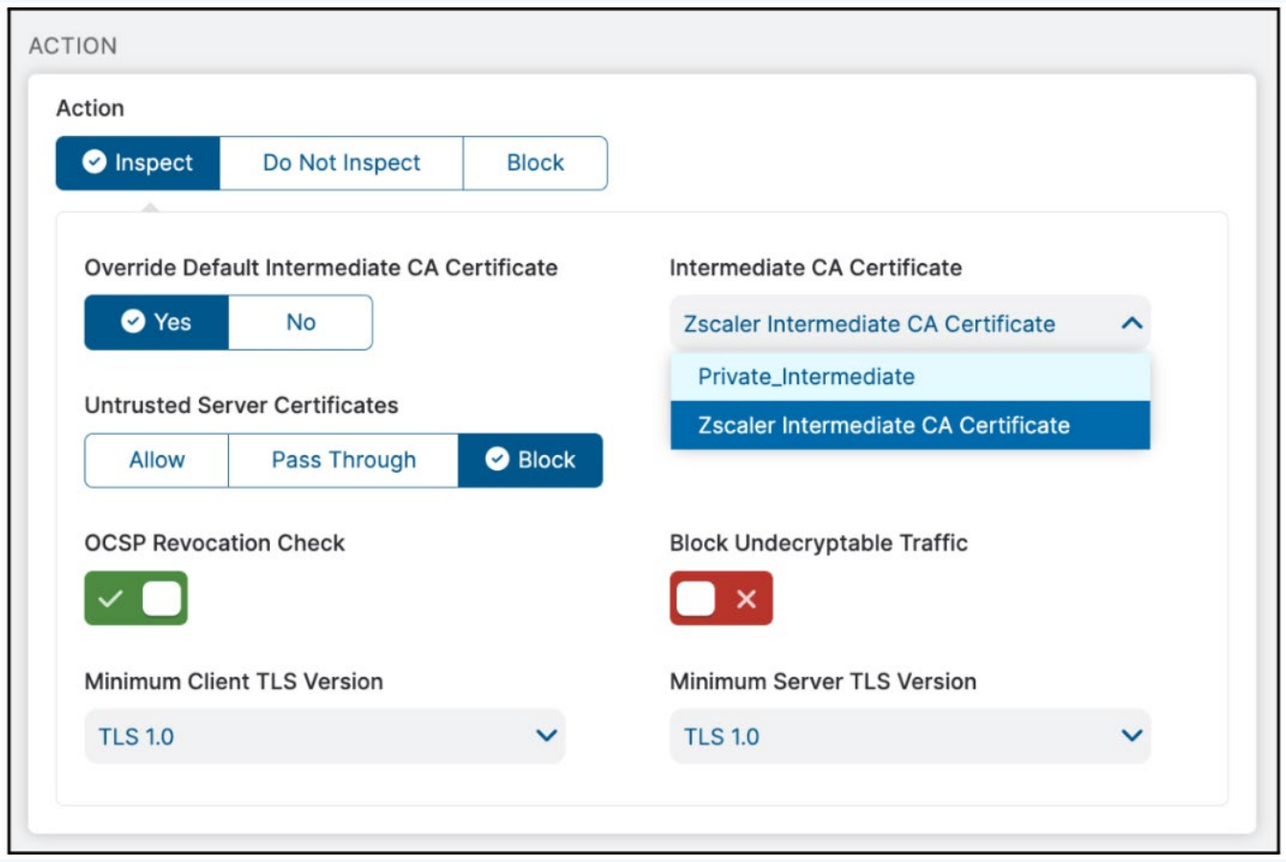
5.5.7 ZIA TLS Interception Certificate Request

When an organization wishes to bring their own certificate to ZIA, they first need to create a request. The table below outlines the steps to accomplish this.

Step	Screenshot Reference
1. Start on ZIA's SSL Inspection – Intermediate CA Certificates and use the Add option.	
2. Proceed with the Private Key. This key is stored inside Zscaler and can not be exported. Proceed with the key creation.	

Step	Screenshot Reference
3. Fill in the CSR details.	
4. The unique details about your certificate need to be adjusted based on your organization. Finish with Generate and Save New CSR.	
5. Proceed to download the file and open it in a text editor.	

Step	Screenshot Reference
6. The CSR's text will be needed in the CA to request the certificate. The next section explains how to request the certificate from Active Directory Certificate Services (ADCS). Once you have the certificate make sure it is in PEM format and the filename ends with .pem and then proceed.	
7. Upload and review.	
8. Complete the upload process. You will see the certificate in the administration panel.	

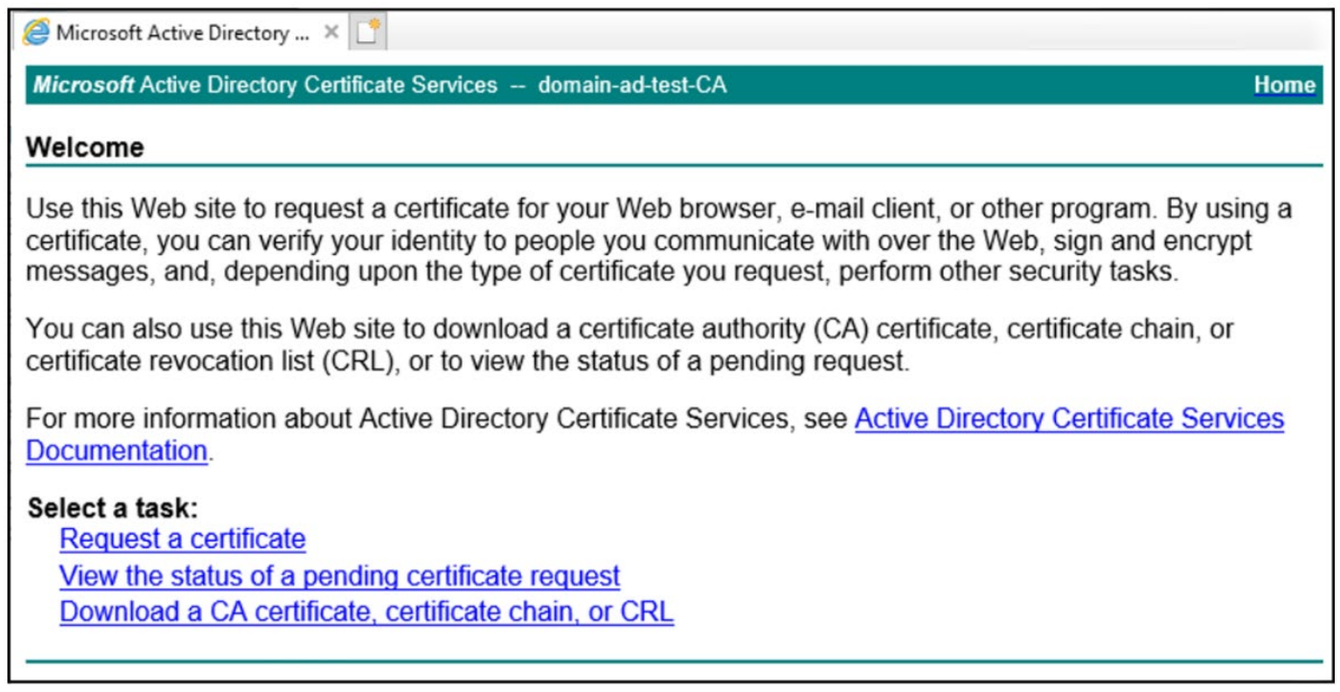
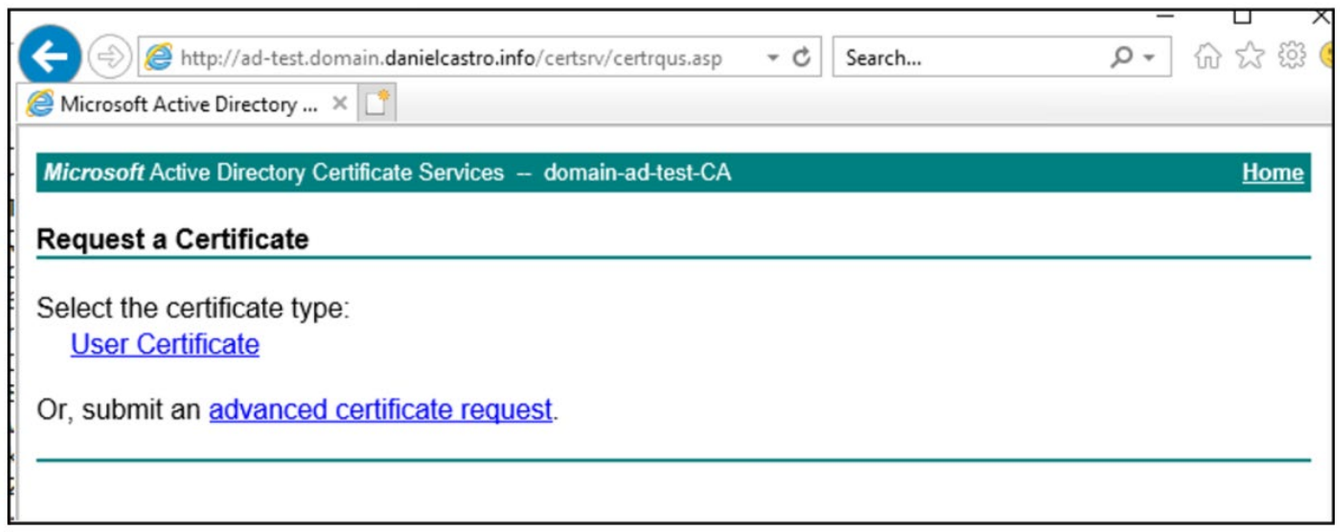
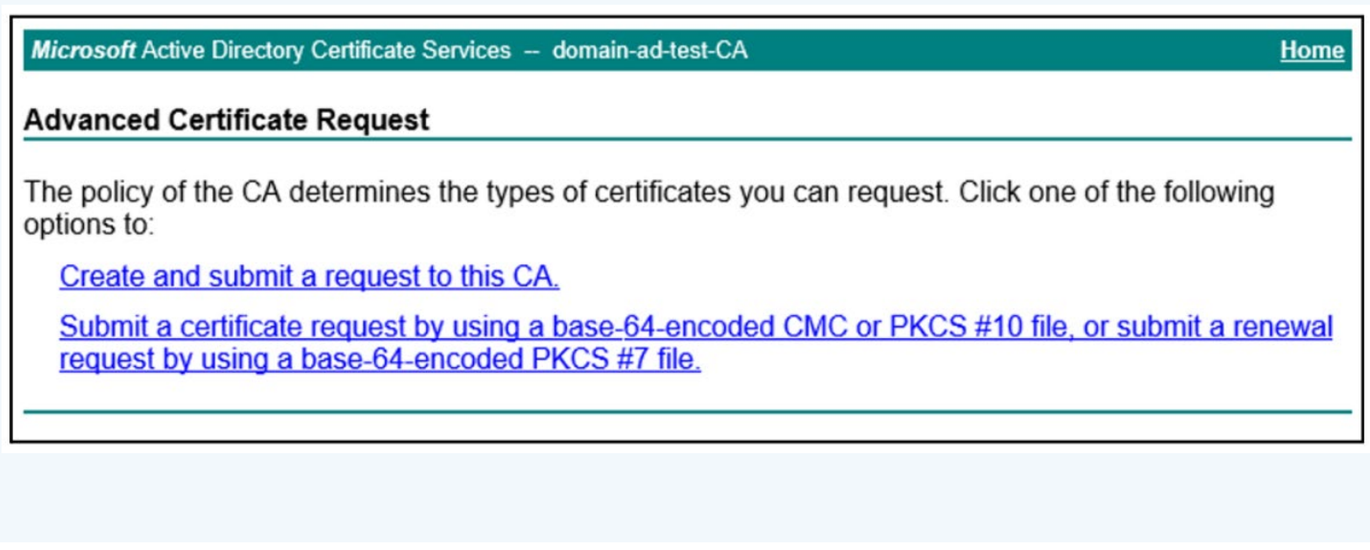
Step	Screenshot Reference
9. To make use of the certificate, use it as a SSL inspection policy action. The following screenshot is from the “SSL Inspection Policy” the policy.	

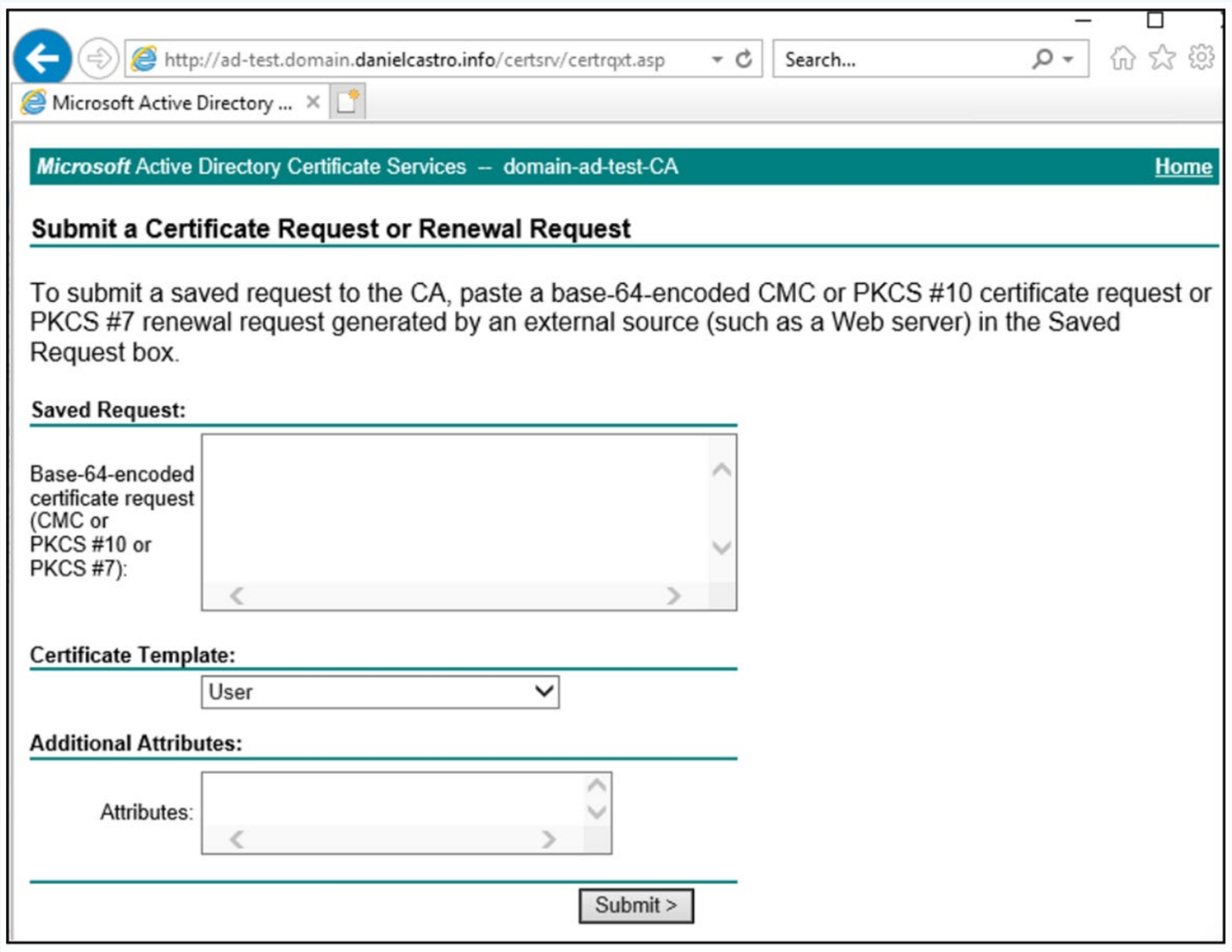
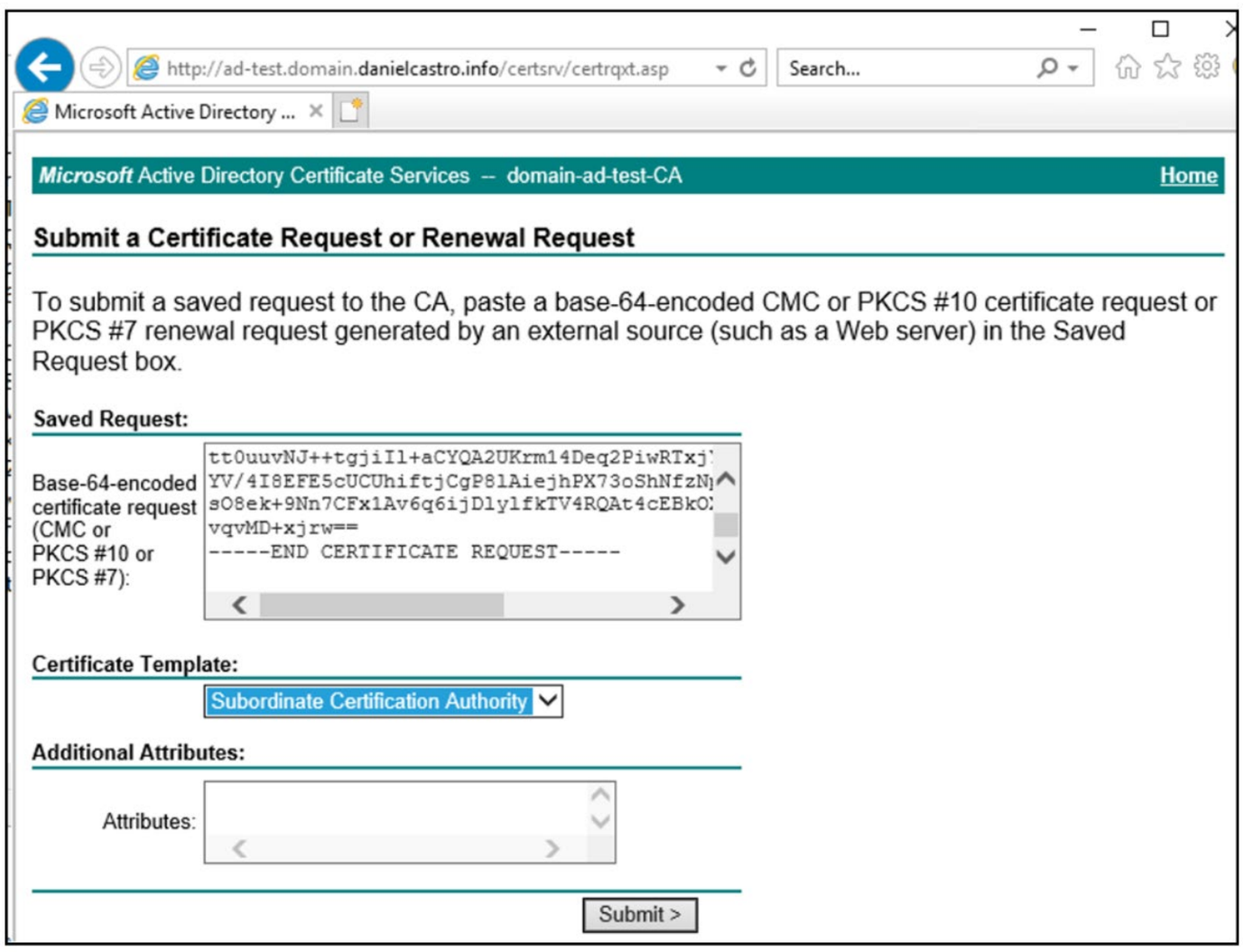
10. You can select now which Intermediate CA Certificate is used for SSL Interception.

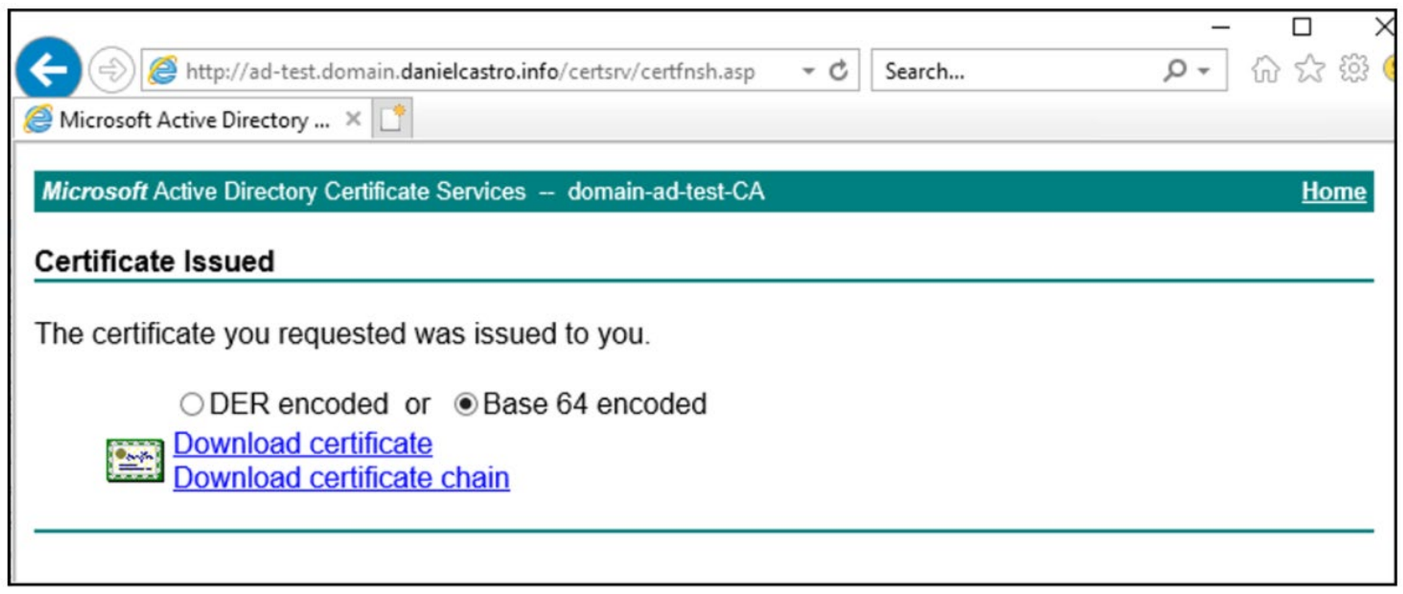
5.5.8 Create an Intermediate Certificate at Active Directory Certificate Services Web Service

Active Directory Certificate Services (ADCS) is a Windows Server role for issuing and managing Public Key Infrastructure (PKI) certificates used in secure communication and authentication protocols. This is usually hosted on a MS Windows server and exposes a Web Server that allows anonymous and authenticated users to perform PKI operations on the Certificate Authority (CA). There can be multiple ADCS as part of a Windows Domain, where each ADCS operates over a part of the certificate chain. In other words, there could be one ADCS for the root CA and multiple ADCS for subordinate intermediate CA that issue server certificates from those intermediate certificates.

If you need instruction on how to install ADCS please see WIndows Certificate Authority Installation Procedure. We will now show how to request an Intermediate Certificate. The steps for both ZIA and ZPA are very similar since the process starts by requesting a CSR from the service. Once you have the CSR file it can be used to sign a certificate at the CA.

Step	Screenshot Reference
1. Get the certificate from Microsoft Certificate Enrollment Services, which allows the usage of a web form to submit the CSR .	
2. Select the advanced certificate request option.	 

Step	Screenshot Reference
3. Use the second option to submit a certificate request.	
4. Paste the contents of the CSR file into the “Saved Request:” section. C Change the “Certificate Template” to “Subordinate Certification Authority” and submit.	

Step	Screenshot Reference
5. Once you submit the request, either the CA is configured to issue the certificate automatically or an administrator must approve the certificate and issue it. You then will login into Microsoft Certificate Enrollment Services and see your approved certificates and be able to download the issued certificate.	

Note: Be careful with the format you download the certificate, as ZIA and ZPA only accept Base64 encoded PEM. If you see the certificate file name extension as p7b you used the wrong format, the correct filename ends with cer.

5.5.9 Create an Intermediate Certificate with OpenSSL

Open SSL is a software library for applications that enables secure communications over computer networks. It allows the creation and verification of certificates and cryptographic keys. In this guide, this will be used to create the root certificate that can be used to sign other certificates.

If you need instruction on how to create a CA using OpenSSL please see OpenSSL Certificate Authority Creation. The rest of this section shows how to create an intermediate certificate using OpenSSL and a pre-existing CA certificate.

In order to obtain an Intermediate certificate using OpenSSL you need to have defined the attributes in the configuration file. The CSR contains the “want” information for the future certificate, and the configuration is the “possibilities” for the future certificate. When generating the certificate, the CSR makes a request and the configuration determines if that is an attribute that can be used in the certificate.

The following is the command that will be used in this example:

```
OpenSSL x509 -in from_zia.csr -out cert.pem -req -signkey root.key
-days 1001 -config rootCA_OpenSSL.conf -key root.key -extensions v3_ca
```

The output of the command will be the certificate in Base64 encoded PEM. Copy and paste the output into a file.

- **days** determine for how many days the certificate will be valid. It is paramount that the number of days is less than the days your root CA is valid, otherwise the resulting certificate will not be accepted by ZIA.

- **config** determines what and how OpenSSL operates on the given command. The configuration file is used from creating the initial root certificate to any other certificate that the CA creates.

- **extensions** determine what attributes will be met for this particular certificate. The possible extensions are defined in the configuration file.

+1 408.533.0288 **Zscaler, Inc. (HQ) • 120 Holger Way • San Jose, CA 95134** **zscaler.com**

Act Fast.
Stay Secure.