ADSelfService Plus

Post-deployment security measures

ManageEngine
ADSelfService Plus

www.adselfserviceplus.com

Table of Contents

1. Introduction 1

2. Steps to promote security during inbound connections 2

- Configure SSL, and add ciphers and protocols 2
- Apply security parameters 4
- Set cookies to HttpOnly 9

3. Measures to promote security during outbound connections 10

- Enable LDAPS 10
- Configure an SSL/TLS connection with the mail server 10
- Configure an SSL connection with the MS SQL server 11

4. Configure file permissions for the ADSelfService Plus installation directory 11

1 Introduction

After the deployment of ADSelfService Plus, there are a few measures that have to be carried out for a secure inbound connection between the ADSelfService Plus server, the user's web browser, and the ADSelfService app. It is also important to protect the outbound connection between the ADSelfService Plus server, the mail server, and the external database server. The ADSelfService Plus installation directory must also be guarded against access by unauthorized users.

This guide details the various steps for implementing these measures and protecting the ADSelfService Plus deployment in your enterprise.

2 Security features that need to be enabled during inbound connections

1. Configure SSL, and add ciphers and protocols

i. SSL configuration

To protect the data transferred between the ADSelfService Plus server, the user's web browser, and the ADSelfService Plus app, and to secure data during API access, SSL certificates should be installed and an HTTPS connection should be configured.

Check out the complete guide on installing SSL certificates for ADSelfService Plus.

To prevent man-in-the-middle attacks during communication between the ADSelfService Plus mobile app and ADSelfService Plus server, SSL pinning should also be enabled. Learn how to enable SSL pinning here.

ii. Add ciphers and protocols

Specific ciphers and protocols can be used to enable forward secrecy. Forward secrecy protects previously recorded traffic between the user's web browser and the ADSelfService Plus server from being decrypted and misused. To configure forward secrecy, add the necessary ciphers and protocols to the server.xml file using these two methods:

Method 1: Via the ADSelfService Plus admin portal

1. Log into the ADSelfService Plus admin portal.
2. Go to **Admin > Product Settings > Connection > Connection Settings.**
3. Click **Advanced Settings**. More settings will now appear.
4. Select **TLSv1.2** from the TLS Versions drop-down.
5. From the **Cipher Suites** drop-down, select the desired ciphers. The following ciphers are recommended as they are the most secure:

a. TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
b. TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
c. TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

Note: Applying these settings will overwrite the current cipher values in the **server.xml** file located under **conf** in the ADSelfService Plus installation directory folder.

Method 2: Manual addition

1. Open the **server.xml** file, located under conf folder in the ADSelfService Plus installation directory folder. Locate the following connector tag:

```
<Connector SSLEnabled="true"
```

Add the following ciphers and protocols to the connector tag:

```
protocol="org.apache.coyote.http11.Http11NioProtocol"
ciphers=HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!kRSA:!3DES:!DHE:
!AES128:!CBC:!TLSv1.0:!TLSv1.1"
allowUnsafeLegacyRenegotiation="false"
server="Adselfservice Plus"
sslProtocol="TLS"
compression="off"
```

```
SSLEnabledProtocols="TLSv1.2"
```

For example:

```
<Connector SSLEnabled="true" acceptCount="100" compression="off"
protocol="org.apache.coyote.http11.Http11NioProtocol"
ciphers="HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!kRSA:!3DES:!DHE:
!AES128:!CBC:!TLSv1.0:!TLSv1.1"
clientAuth="false" connectionTimeout="-1" debug="0"
disableUploadTimeout="true" enableLookups="false"
keystoreFile="./conf/server.keystore" keystorePass="adventnet"
maxSpareThreads="75" maxThreads="150" minSpareThreads="25"
name="SSL" port="9251" scheme="https" secure="true"
allowUnsafeLegacyRenegotiation="false" server="AdselfservicePlus"
sslProtocol="TLS" sslEnabledProtocols="TLSv1.2"/>
```

2. Apply security parameters

Security parameters define the headers of the HTTP response messages from the ADSelfService Plus server. They help mitigate attacks by instructing the end user's web browser how to handle the communication between the server and the web browser. Security parameters protect communication between the server and the web browser by:

1. Preventing the web browser from caching the server response.
2. Preventing XSS (cross-site scripting) attacks.
3. Preventing data and code injection attacks
4. Allowing only HTTPS connections and restricting HTTP connections.
5. Preventing clickjacking.

Follow these steps to successfully apply the security parameters:

Case 1: For ADSelfService Plus build versions 6304 and above, the security parameters file named security_params.conf is found under <Install_directory>/conf.

Case 1: For ADSelfService Plus build versions below 6304, follow these steps:

1. Download the security parameters file from here.
2. Go to the <Install_directory>/conf folder, and place the downloaded file in it.

The following are the headers present in the ADSelfService Plus security_params.conf file.

Cache-control and pragma

The cache-control and pragma headers hold instructions on the use of cache by the browser for the application. Both headers address the same purpose. While cache-control is used by the latest versions of HTTP, pragma is used for backwards compatibility with HTTP version 1.0. The following directives are present in this header:

no-cache: Stores caches but the browser still looks for updates in the application server before reusing the stored data.

no-store: Caches are not stored.

In the security_params.conf file this policy is defined as:
cache-control= no-cache, no-store
pragma=no-cache

X-Content-Type

The x-content-type header specifies that the Multipurpose Internet Mail Extensions (MIME) types mentioned in the content-type HTTPS header must be followed without deviation. This header helps prevent MIME type sniffing attacks. This header requires the nosniff directive to be set.

nosniff: Blocks an HTTPS request that is of the type style and if the MIME type is not text/css or a JavaScript MIME type.

In the security_params.conf file this policy is defined as:

x-content-type-options=nosniff

Strict-Transport-Security

The HTTPS Strict Transport Security (HSTS) header enforces HTTPS during all attempts to connect to a web page. During the first connection to the web page, the HSTS header is sent back to the end user's web browser. This header instructs the browser to connect to the webpage using only HTTPS. As a result, only HTTPS connections to the webpage are successful, other connections are terminated. This helps prevent man-in-the-middle attacks like cookie hijacking.

The HSTS header in the security_params.conf files consists of the following policy directive:

max-age=<expire-time>: Sets the maximum time, in seconds, for which the end user's browser remembers that the webpage must be only accessed using HTTPS.

includeSubDomains: Enforces this header across all subdomains in the webpage.

In the security_params.conf file this policy is defined as:
Strict-Transport-Security=max-age=31536000; includeSubDomains

X-Frame-Options

The X-Frame-Options header specifies whether the end user's browser can be allowed to embed and display the webpage in another webpage. This helps prevent clickjacking by ensuring both the embedded page and the parent page are secure. The header can contain one of these two policy directives:

SAMEORIGIN: Allows the webpage to be embedded in another webpage from the same domain

DENY: Prevents the webpage from being embedded in another webpage.

In the security_params.conf file this policy is defined as:

#x-frame-options=SAMEORIGIN

The header is commented by default, and will not be applied unless you uncomment it by removing the # symbol in the beginning. If you are uncommenting this header, you must modify frame-src and frame-ancestors directives in the Content-Security-Policy header as mentioned below.

x-frame-options=SAMEORIGIN

X-XSS-protection

This header is specific to legacy versions of web browsers such as Internet Explorer 8+, Chrome, Edge, Opera, and Safari. It is dedicated to filtering and preventing cross-site scripting (XSS) attacks. It is generally not required in latest browsers versions and its functionality is covered by the Content-Security-Policy header instead. The following policy directives are a part of this header:

1 (default): Enables XSS filtering

mode=block: Blocks the page from rendering if a cross-site-scripting attack is detected.

This header is a part of the security.params file to accommodate to environments that continue to use legacy browser versions. In the security_params.conf file this policy is defined as:

X-XSS-Protection=1; mode=block

Referrer-Policy

The Referrer-Policy header controls how much referrer information must be added to the HTTP request as part of the Referer header which contains the origin address of the HTTP request. While the Referer header is used for analytics, logging, or optimized caching, it can be exploited for data theft. The Referrer-Policy header decided what type of information can be carried by the Referer header thereby preventing data exposure and exploitation.

ADSelfService Plus offers the following directive:

strict-origin-when-cross-origin: Sends the origin, path, and querystring when performing a same-origin request. For cross-origin requests sends only the origin if when the HTTPS remains the same. Doesn't send the Referer header to less secure destinations that don't support HTTPS.

In the security_params.conf file this header is defined as:

Referrer-Policy=strict-origin-when-cross-origin

Content-Security-Policy

The Content-Security-Policy header is used to define trusted sources for the resources to be rendered on a webpage. Once these sources are defined, only resources from them can be executed on the webpage. This helps prevent code injection attacks like XSS attacks and clickjacking.

This header is commented by default, and will not be applied unless you uncomment it by removing the # symbol in the beginning.

Also, the ADSelfService Plus server is the only source defined as trusted resources for content rendered in the ADSelfService Plus portal. This can be modified as described below.

The following policy directives are a part of this header in the security_param.conf:

img-src: Defines the trusted sources for image resources.

script-src: Defines the trusted sources for Javascript resources.

worker-src: Defines the trusted sources for Web Worker scripts.

connect-src: Defines the trusted sources for API scripts.

style-src: Defines the trusted sources for stylesheets.

frame-src: Defines the trusted sources for HTML frame resources.

frame-ancestors: Specifies the parent HTML webpages where the webpage can be embedded

default-src: Acts as a fallback for other directives by defining the trusted source for all resources

By default, under the Content-Security-Policy header in the security_params.conf file, the ADSelfService Plus server is the only source defined as trusted resources for content rendered in the ADSelfService Plus portal. The ADSelfService Plus server's access URL is used here. Ensure that the domain address mentioned in the URL also contains the subdomain name if present.

Note: Modify the Content Security Policy header

1) Modifying the source

In the existing security_params.conf file, 'self' is defined as the trusted source for the rendered resources. The source 'self' refers to the webpage's origin. In this case, it refers to the ADSelfService Plus server's access URL. Here is the existing Content-Security-Policy header:

```
#Content-Security-Policy=default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:; connect-src 'self'; img-src 'self' data:; style-src 'self' https://fonts.googleapis.com 'unsafe-inline'; frame-src 'self' https://*.duosecurity.com/; frame-ancestors 'self';
```

For improved security, organizations can replace the source from 'self' to the exact domain address of the origin domain. In case any other domain gets designated as the origin, this prevents its content from being executed in the ADSelfService Plus portal.

Consider an example where ADSelfService Plus is deployed in the domain abcdcorp.com in an organization. This domain can be exclusively defined as the source for the content executed by replacing 'self' with the domain's complete domain address as mentioned below:

```
#Content-Security-Policy=default-src https://abcdcorp.com; script-src https://abcdcorp.com 'unsafe-inline' 'unsafe-eval'; worker-src https://abcdcorp.com' blob:; connect-src https://abcdcorp.com; img-src https://abcdcorp.com data:; style-src https://abcdcorp.com https://fonts.googleapis.com 'unsafe-inline'; frame-src https://abcdcorp.com https://*.duosecurity.com/; frame-ancestors https://abcdcorp.com ;
```

2) Modifying the frame-src and frame-ancestors directive

- If ADSelfService Plus is integrated as part of the AD360 suite, the hostnames of the integrated ManageEngine products' servers must be mentioned as trusted resources for frame-ancestors and frame-src as well.
- If MFA for OWA logins is enabled, the Exchange server's hostname must be mentioned as a trusted source for frame-ancestors as well.

3) Modifying the img-src, style-src, and script-src directive

If you are including images, stylesheets, and Javascript resources from other trusted sources, you must mention their source URL in the next to these directives.

3. Set cookies to HttpOnly

Setting cookies to HttpOnly permits only the ADSelfService Plus server to access the cookies and blocks any script from the web browser side from accessing it. From builds 6304 and above, cookies are automatically set to HttpOnly.

For builds lower than 6304, set the cookies to HttpOnly by running the following query in the database:

```
"insert into systemparams values((select max(system_param_id) from
systemparams)+1,'ENABLE_HTTPONLY','true');"
```

Note:

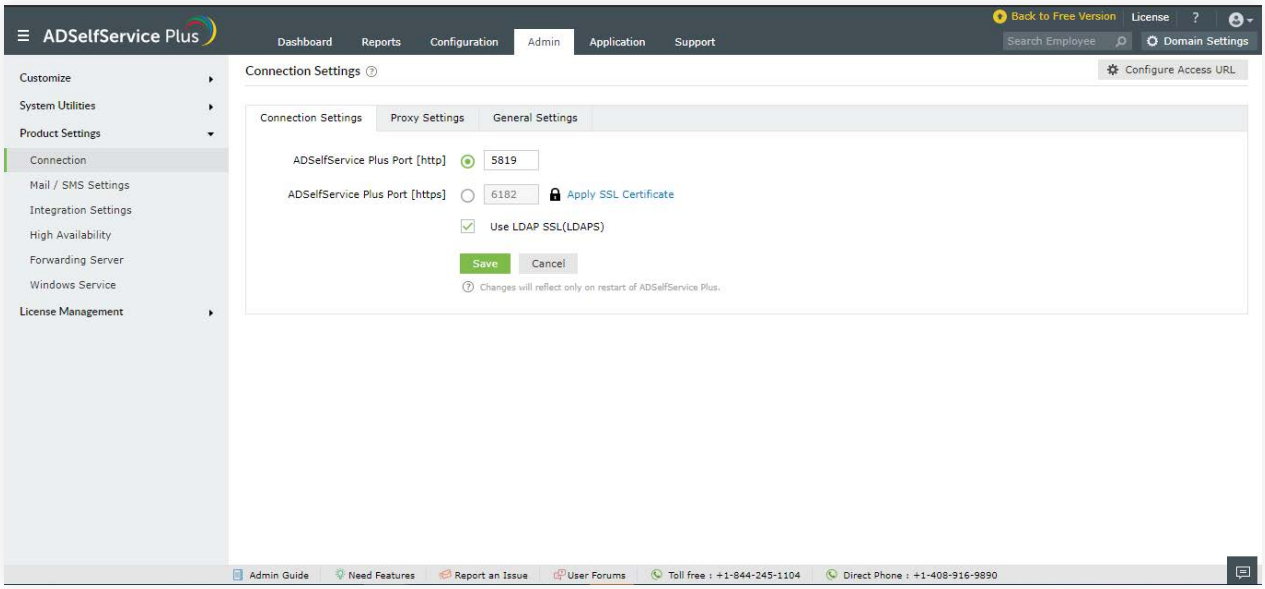
Since the adscsrf and _zcsr_tmp cookies are required to be accessed by the web browser for the functioning of the ADSelfService Plus portal, HttpOnly will not be set for them.

3 Measures to promote security during outbound connections:

1. Enable LDAPS

When the Active Directory domain controller has SSL enabled (recommended), a Secure Lightweight Directory Application Protocol (LDAPS) connection can be configured in the ADSelfService Plus Connection settings to ensure a secure connection between ADSelfService Plus and Active Directory. Follow these steps to enable LDAPS connection:

1. In the ADSelfService Plus administrator portal, open the **Admin** tab.
2. Go to **Connection (Admin > Product Settings > Connection).**
3. Check the Use **LDAP SSL (LDAPS)** box.
4. Click **Save.**



2. Configure an SSL/TLS connection with the mail server

After deploying the mail server with a specific protocol (SSL/TLS), you need to configure the same protocol in the ADSelfService Plus Mail Server settings. This is done to establish a secure connection between the ADSelfService Plus server and the mail server. Check out this article for details on how to enable an SSL/TLS connection between ADSelfService Plus and the mail server.

3. Configure an SSL connection with MS SQL Server

ADSelfService Plus supports MS SQL in addition to the built-in PostgreSQL. To secure the data transferred between the ADSelfService Plus server and MS SQL Server, it is necessary to configure an SSL connection between them. This is done by applying an SSL certificate in SQL Server. This guide offers a detailed explanation on how to secure the connection between ADSelfService Plus and MS SQL using SSL.

4 Configure file permissions for the ADSelfService Plus installation directory

The ADSelfService Plus installation directory contains important files and folders, including the license file and files that are used to start and stop the product. Administrators need to be provided with file permissions to the folder where the installation directory is located (for example, the C:\Program Files\ManageEngine folder) to be able to access and modify the folder contents. Assigning Full Control permissions gives the administrator all the access they need and more. However, this may lead to security vulnerabilities. To overcome this, we recommend delegating only the permissions they need by following the steps in this document.

Our Products

AD360 | Log360 | ADManager Plus | ADAudit Plus | RecoveryManager Plus | M365 Manager Plus

ManageEngine
ADSelfService Plus

ADSelfService Plus is an identity security solution to ensure secure and seamless access to enterprise resources and establish a Zero Trust environment. With capabilities such as adaptive multi-factor authentication, single sign-on, self-service password management, a password policy enhancer, remote work enablement and workforce self-service, ADSelfService Plus provides your employees with secure, simple access to the resources they need. ADSelfService Plus helps keep identity-based threats out, fast-tracks application onboarding, improves password security, reduces help desk tickets and empowers remote workforces. For more information about ADSelfService Plus, visit https://www.manageengine.com/products/self-service-password.

$ Get Quote | Download | Support