[SRX] How to block specific HTTPS URLs through Application Firewall

[KB28761] Show KB Properties

SUMMARY:

Sometimes customers need to block sub-URLs of a website rather than blocking the whole site. This article provides the instructions to achieve that task using Junos AppSecure Application Firewall.

PROBLEM OR GOAL:

Customer needs to block the following URLs without blocking the whole site `creative.adobe.com`, i.e., users should still be allowed to browse `https://creative.adobe.com`.

https://creative.adobe.com/api/assets
https://creative.adobe.com/api/collections
https://creative.adobe.com/api/share
https://creative.adobe.com/files

CAUSE:

SOLUTION:

This can be accomplished by using the Junos AppSecure Application Firewall, but since the URLs are HTTPS, they are encrypted. AI requires decrypted data in order to recognize the HTTP pattern and content. The data must be decrypted first, using the SSL Forward Proxy feature of Junos OS.

The following steps summarize how the task can be accomplished:

1. Configure SSL Forward Proxy (SSLFP) so that SSL traffic gets decrypted into standard HTTP for inspection, that way the SRX can look at the sub URLs
2. Create a custom nested application to identify the specific sub URLs within the main site
3. Block the custom app through Application Firewall
4. Apply both AppFw and SSLFP profile to the relevant security policies

The following instructions describe the steps in detail, with a sample configuration. Refer to http://www.juniper.net/techpubs/en_US/junos12.1x44/information-products/pathway-pages/security/security-basic-ssl-proxy.pdf for detailed information about the Junos SSL Forward Proxy feature.

1. Generate the self-signed cert on the SRX.

```
SRX>request security pki generate-key-pair certificate-id ssl-inspect-ca size 1024 type
rsa
SRX>request security pki local-certificate generate-self-signed certificate-id ssl-
inspect-ca domain-name www.juniper.net subject "CN=www.junper.net,OU=IT,O=Juniper
Networks,L=Sunnyvale,ST=CA,C=US" add-ca-constraint
```

2. Configure the loaded self-signed cert as root-ca:

```
SRX#set services ssl proxy profile ssl-inspect-profile root-ca ssl-inspect-ca
```

3. Load ALL the trusted certificates of the browser onto the SRX. Follow KB23144 - How to download Trusted CAs from a browser to a SRX-series device and verify that all the certs are loaded onto SRX as instructed in that article..

4. Trust all the loaded trusted certs of the browser:

```
set services ssl proxy profile ssl-inspect-profile trusted-ca all
```

5. Create a whitelist to exempt the known sites from getting decrypted by SRX. This example shows that `www.juniper.net` is exempted:

```
set security address-book global address ssl-inspect-exempt dns-name www.juniper.net
set services ssl proxy profile ssl-inspect-profile whitelist ssl-inspect-exempt
```

6. Ignore errors encountered during server certificate verification process at the time of SSL handshake:

```
set services ssl proxy profile ssl-inspect-profile actions ignore-server-auth-failure
```

7. Here is the custom nested application that will block the specified sub-URLs, while still allowing access to the main site `https://creative.adobe.com`:

```
set services application-identification nested-application adobe-api-files protocol HTTP
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m01 context http-header-host
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m01 pattern "(.*\.)?creative\.adobe\.com"
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m01 direction client-to-server
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m02 context http-url-parsed
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m02 pattern "/(api/(assets|collections|share)|files)(/.*)?"
set services application-identification nested-application adobe-api-files signature
adobe-api-files member m02 direction client-to-server
set services application-identification nested-application adobe-api-files signature
adobe-api-files maximum-transactions 1
```

8. Application firewall config which blocks the above custom app and will allow everything else:

```
set security application-firewall rule-sets block-adobe rule r1 then deny
```

```
set security application-firewall rule-sets block-adobe default-rule permit
```

9. Apply AppFw and SSL proxy to security policy:

```
set security policies from-zone trust to-zone untrust policy trust then permit
application-services ssl-proxy profile-name ssl-inspect-profile
set security policies from-zone trust to-zone untrust policy trust then permit
application-services application-firewall rule-set block-adobe
set security policies from-zone untrust to-zone trust policy untrust then permit
application-services ssl-proxy profile-name ssl-inspect-profile
set security policies from-zone untrust to-zone trust policy untrust then permit
application-services application-firewall rule-set block-adobe
```

10. Commit the changes.

```
commit
```

Upon testing the issue, we can see that all the sub-URLs are blocked by AppFw, No files can be viewed, no files can uploaded, etc., while access to the main site `https://creative.adobe.com` itself is permitted.

```
root> show security application-firewall rule-set all
Rule-set: block-adobe
Logical system: root-logical-system
Rule: r1
Dynamic Applications: adobe-api-files
SSL-Encryption: any
Action:deny
Number of sessions matched: 12
Number of sessions redirected: 0
Default rule:permit
Number of sessions matched: 8
Number of sessions redirected: 0
Number of sessions with appid pending: 0
```

NOTE: It is important to load ALL the trusted certificates of the browser, otherwise SSL proxy will not work thereby blocking or allowing relevant traffic depending upon AppFW rules. In this specific example, if SSL FP is not able to decrypt the traffic properly for some reason (example missing certs), AI cannot recognize the custom app, thereby all traffic will fall under permit rule in this example.

PURPOSE:

Implementation

RELATED LINKS: