Jamf Professional Services

Technical Note

Network-Level Analysis and Filtering of Jamf Pro Server Communications

\

v2018-02-07/ol

Table Of Contents

Introduction

Audience and Purpose

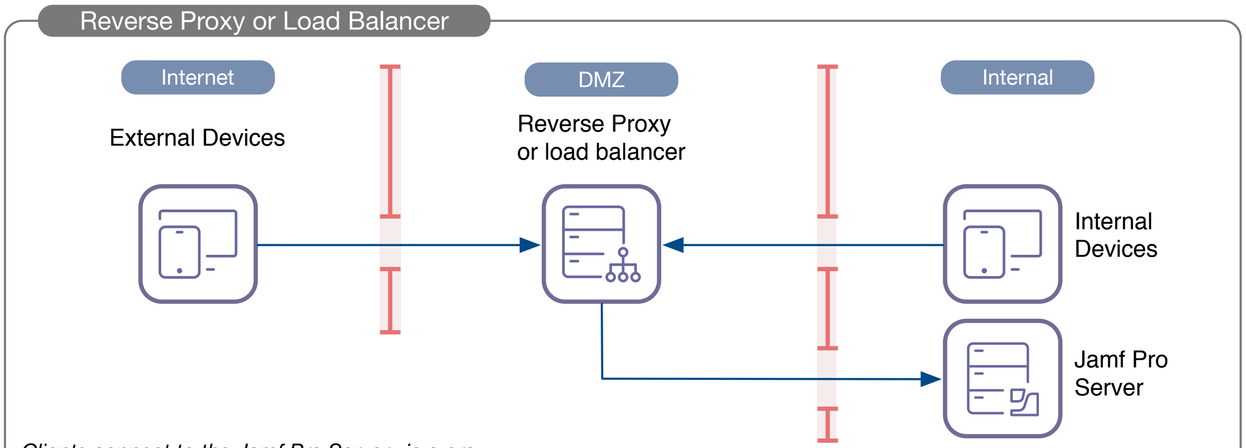
This document is intended for Information Security Professionals who are considering the use of a web application firewall to protect their implementation of Jamf Pro. We will discuss some relevant security concepts and demonstrate how to observe the traffic flowing between the Jamf Pro server and managed clients so that analysts can identify common patterns in their environment and implement precautions consistent with their standards.

Customer Objectives

Nearly every enterprise has some Apple-OS endpoints in its IT environment, and the number of users preferring these devices continues to increase, making a management framework necessary. The majority of these devices are highly mobile, which means that many Jamf customers choose to expose their management system to the internet.

Jamf's security conscious customers will be most interested in understanding the connections that are initiated by a remote host to the Jamf Pro server so they can prevent intrusion from unauthorized traffic. Traditionally, customers were often particularly interested in protecting the server from internet-based traffic, but with the increasing prevalence of compromised hosts even on internal, firewall-protected networks, it is now common that customers will wish to treat both internal and external connections with equal suspicion.

A common deployment configuration for customers hosting their own Jamf Pro server is to expose a reverse proxy or load balancer (either a classic layer 4/5/6 solution or a modern layer 7 traffic inspection product) to the internet and have that forward traffic to the Jamf Pro application. Firewalls are configured so the Jamf Pro server will only receive traffic sent through the reverse proxy, thus creating layers of security between the server and connecting hosts.



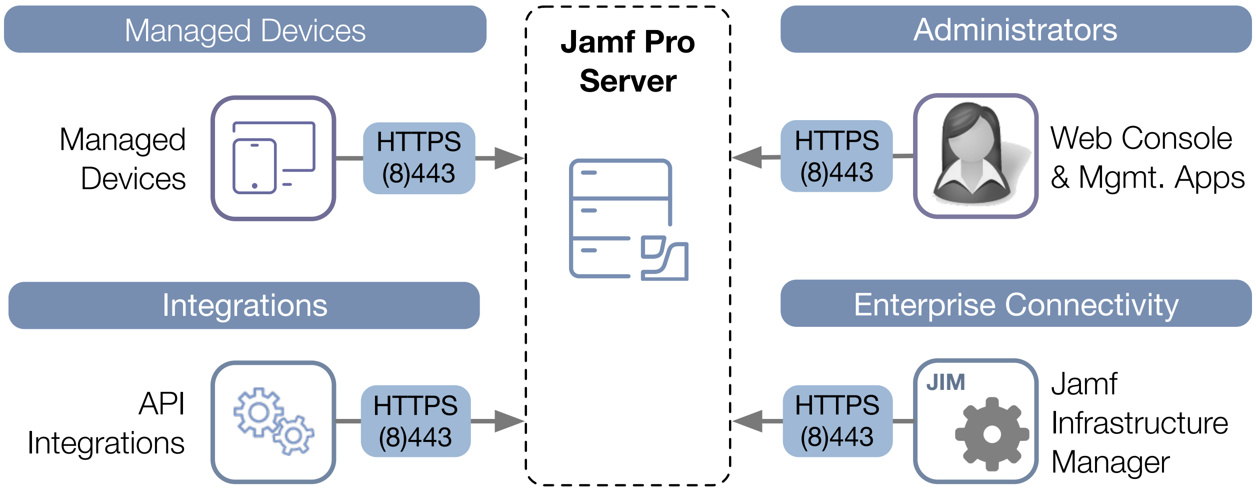
Clients connect to the Jamf Pro Server via a proxy.
Customers using this configuration might also use an internal "management only" master Jamf Pro server.

Monitoring Jamf Pro Server Communications

Sources of remotely-initiated network connections to the Jamf Pro server

Inbound-initiated connections to the Jamf Pro server originate from several sources. These include:

- Administrators connecting to the admin console from a web browser or via a Jamf macOS app (Jamf Admin, Remote, Recon, and Imaging)
- Users running the Jamf Self Service app on iOS or macOS
- The Jamf Agent software running on macOS
- The mdmAgent component of iOS and macOS
- API integrations that customers may choose to deploy, such as SCCM and ServiceNow
- Enrollment and periodic checkin from a Jamf Infrastructure Manager



Summary of Non-WireShark Methods

A number of solutions can be used to record Jamf Pro traffic, but not all of these are appropriate for a production environment as they involve using weaker ciphers or sharing keys. Any research project of this type should be supervised by a security professional as they will be familiar with potential vulnerabilities that might be created. In the following sections, we will demonstrate how to monitor traffic in a lab environment. This will be sufficient to observe the traffic you should expect to occur between Jamf Pro and managed clients.

Configuring Apache Tomcat to Run on a Non-Encrypted Port – Lab environment only

It is possible to run Jamf Pro's Apache Tomcat service on HTTP rather than HTTPS. Obviously, you would never do this on a production system or using production data or credentials. Once HTTP is configured, you can use any TCP capture utility to watch the server traffic without any need for decryption. To do this, open a firewall (if used) to allow a non-encrypted port, and add an HTTP listener to Apache Tomcat's server.xml file (in this example, 8080 is used)…

```
<Connector URIEncoding="UTF-8" port="8080" executor="tomcatThreadPool"
protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />
```

Use the Jamf Pro console to change the management url to “http://<hostname>:<port>/“. Then you can conduct many workflows as you normally would and observe the traffic in its unencrypted form.

Decrypting Proxies (“Man in the Middle”)

Logging debugger proxies (e.g. Fiddler/Charles), and enterprise application delivery controllers with traffic inspection features (Big-IP, Catalyst, NetScaler, et. al.) are commonly used to observe web application traffic. These techniques will work well to monitor and log Jamf Pro communications, but, as with any web application running over TLS, standard requirements to install and trust any internal root and intermediate certificates on connecting devices apply. For some workflows, the proxy will need to present a TLS identity keypair obtained from a well-known third-party CA so that the trust chain will already be installed on your Apple OS devices. For example, when enrolling a new device, you will not have had an opportunity to install your own internal CA’s root certificates prior to enrolling the device through Apple’s Device Enrollment Program.

Of note: To create strong protection against MITM attacks on it’s cloud-based services, Apple employs certificate pinning in its OSs. If you place a decrypting outbound proxy layer (e.g. Blue Coat) at the edge of your network for inspection of outbound packets from Apple clients, the APNS, App Store, and VPP features of Apple’s operating systems will not connect to Apple, even if you have pre-trusted your proxy’s trust chain on your devices. The hash for these certificates will not be accepted as valid by Apple operating systems for these MITM-protected services.

Monitoring Jamf Pro Server Communications with WireShark

Overview

WireShark is a capable and commonly-used network monitoring utility. This section will explain how to configure Tomcat and WireShark to monitoring an HTTPS application such as Jamf Pro

The session-key logging feature available in some web browsers is commonly used to monitor web applications from the client-side. That will work well when using a browser to connect to the Jamf Pro Admin console, but not for any other Jamf Pro communications. It is easier to monitor traffic from the server side by running your expected endpoint management workflows against a lab server.

Configuring Tomcat Ciphers

WireShark can only decrypt traffic if we do not enforce forward secrecy. We will need to use an RSA key exchange mechanism. This may be done in a lab setup — in production, Jamf Pro communications are exclusively via SSL and strong ciphers should be used.

To configure the ciphers, open ./tomcat/conf/server.xml and edit the SSL connector so that the ciphers are as follows:

```
ciphers="SSL_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_AES_128_CBC_SHA,
SSL_RSA_WITH_3DES_EDE_CBC_SHA”
```

For details, please see: “Configuring Supported Ciphers for Tomcat HTTPS Connections” https://www.jamf.com/jamf-nation/articles/384/configuring-supported-ciphers-for-tomcat-https-connections.)

Configuring WireShark for TLS/SSL Decryption

Step 1: Obtain the Private Key from Apache Tomcat

In most Jamf Pro implementations, the server’s TLS keypair is stored in a java keystore. The Java keytool utility won’t export individual private keys, but it will export the store to a PKCS12. Use the following command to convert .jks to PKCS12 (.p12/.pfx) format. (Note that paths shown are typical for a macOS installation of Jamf Pro server and will vary elsewhere.)

```
$ sudo keytool -importkeystore -srckeystore "/Library/JSS/Tomcat/
TomcatSSLKeystore" -destkeystore "/Users/admin/Desktop/tomcat.p12" -
srcstoretype JKS -deststoretype PKCS12 -deststorepass 'changeit' -srcalias
tomcat -destalias tomcat -destkeypass 'changeit'
```

You can view the content of the pkcs12 keystore with:

```
$ openssl pkcs12 -in "/Users/admin/Desktop/tomcat.p12"
```

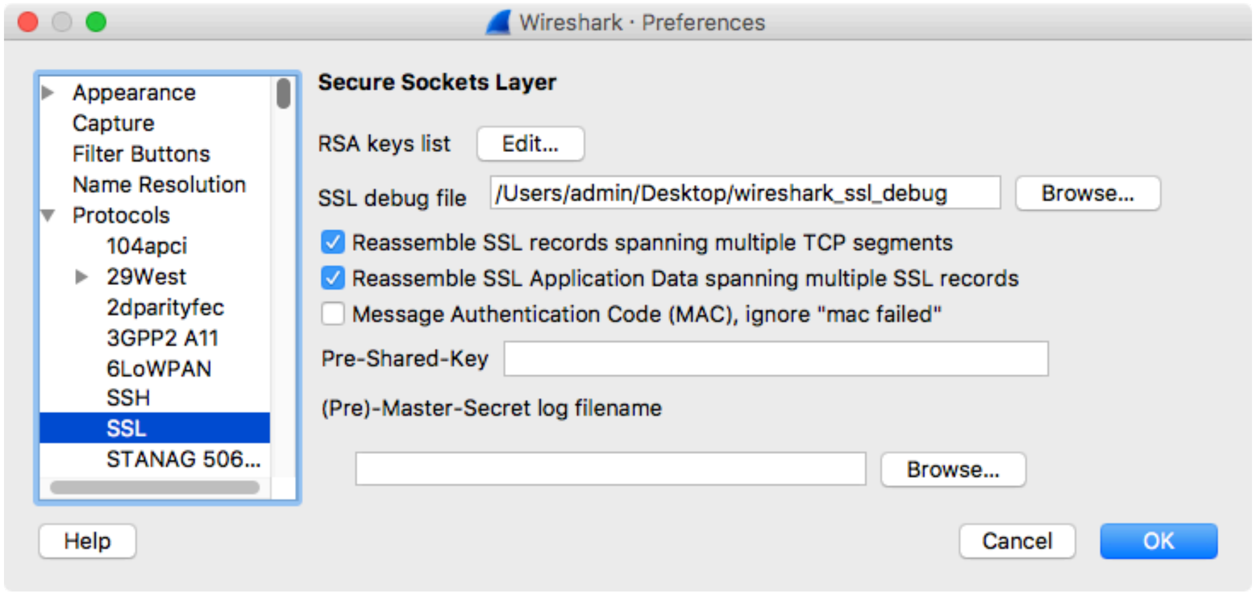
Export the private key. You will be prompted for the password of the p12 and a password for the .pem.

```
$ openssl pkcs12 -in mykeystore.p12 -nocerts -out key.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

You can also export keys with no DES by adding the “-nodes” parameter. This was required by old versions of WireShark, but it’s bad practice and no longer required. No matter the method used, be sure to protect this file.

Step 2: Install the private key in WireShark

Open WireShark Preferences and select “SSL” under protocols:



Secure Sockets Layer

RSA keys list

SSL debug file: /Users/admin/Desktop/wireshark_ssl_debug

✓ Reassemble SSL records spanning multiple TCP segments

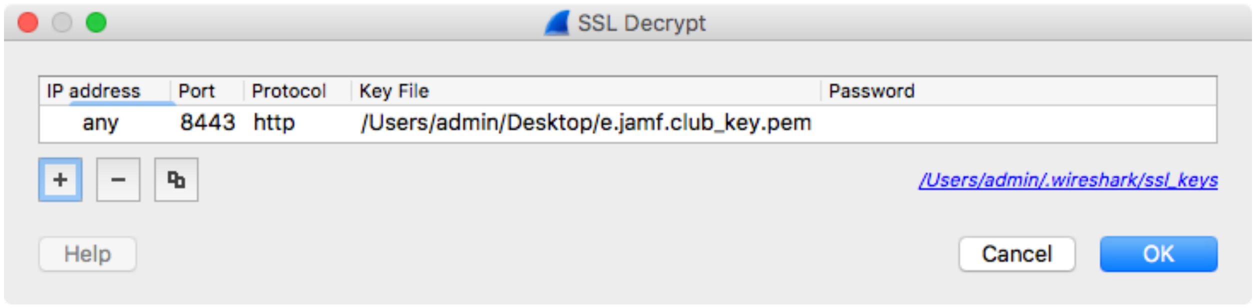
✓ Reassemble SSL Application Data spanning multiple SSL records

☐ Message Authentication Code (MAC), ignore "mac failed"

Pre-Shared-Key

(Pre)-Master-Secret log filename

In the SSL Decrypt screen, click the “+” button at lower left and enter the IP address of the host you are monitoring (or “any” if your host has just one IP or uses multiple IPs), the TLS port you want to decrypt, set protocol to “http” (note the lower-case). Supply the private key file created in Step 1, and the DES password.



IP address	Port	Protocol	Key File	Password
any	8443	http	/Users/admin/Desktop/e.jamf.club_key.pem	

/Users/admin/.wireshark/ssl_keys

Step 3: Configure WireShark Dissectors

The above configurations will let us view communications between the Jamf Agent on macOS, but some additional setup is needed to observe communications between the Jamf Pro Server and Apple OSs’ mdmclient subsystem because custom mime-types are used. When viewing traffic between these two parties, you will notice custom media types being returned from the server. WireShark will not know how to parse these so the traffic will not be displayed in a human-readable form. These media types include:

- application/x-x509-ca-cert
- application/octet-stream
- application/x-pki-message
- application/x-apple-aspen-mdm
- application/x-apple-aspen-mdm-checkin

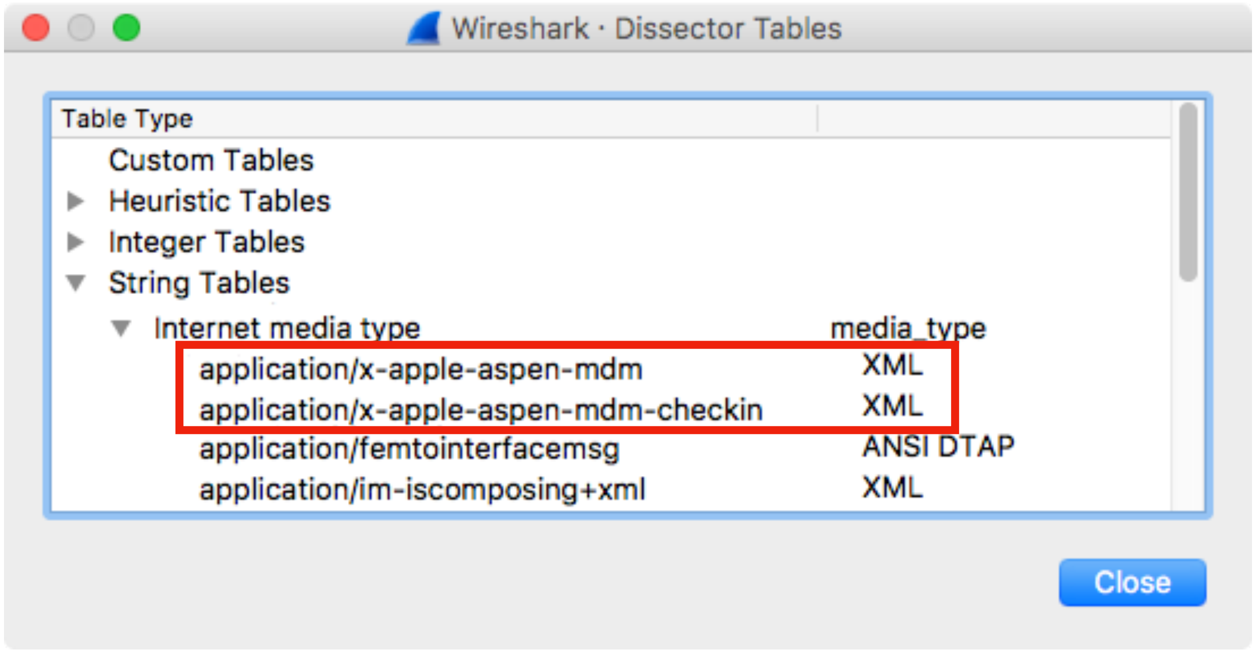
The first three will be standard SCEP certificate deployment transactions. The aspen packets are Apple MDM protocol XML sent via HTTP. These are shown in detail in Appendix 2 of this document.

Use the "View" menu > "Internals" submenu > "Dissector Tables" menu item > "String Tables" > "Internet media items" to see that there is no entry available to map MDM mime types to the XML dissector.

To correct this, we can use WireShark Lua commands to register additional mime-type entries for the XML dissector. For example, to read "x-apple-aspen-mdm" packets as XML, select "Tools" menu > "Lua" > "Evaluate" and run the following command:

```
DissectorTable.get("media_type"):add("application/x-apple-aspen-mdm",
Dissector.get("xml"))
```

After running this command, we can observe that the apple mdm type will now be processed with the XML dissector ("View" menu > "Internals" > "Dissector Tables"):



Internet media type	media_type
application/x-apple-aspen-mdm	XML
application/x-apple-aspen-mdm-checkin	XML
application/femtointerfacemsg	ANSI DTAP
application/im-iscomposing+xml	XML

Documentation of MDM message formats is available from: https://developer.apple.com/library/content/documentation/Miscellaneous/Reference/MobileDeviceManagementProtocolRef/3-MDM_Protocol/MDM_Protocol.html

Step 4: Run WireShark Capture

Start a WireShark capture on the network interface used by your web server or service with a capture filter for the port to which Tomcat is listening. For example,"port 8443". You can remove TCP (ACK, PSH) packets from the display with a display filter of "http".

[End of Document]

This page intentionally blank.