Fairfield UNIVERSITY

International Journal of Computer and Systems Engineering

Volume 2 | Issue 1 Article 1

June 2021

An Appstore for Java

Douglas A. Lyon
Fairfield University, dlyon@fairfield.edu

Riley Fitzsimmons
riley.fitzsimmons@student.fairfield.edu

International Journal of Computer and Systems Engineering

Online at https://digitalcommons.fairfield.edu/ijcase/ Published by Fairfield University, **DigitalCommons@Fairfield © 2021**

September 2021

An AppStore for Java

Doug Lyon, Riley Fitzsimmons
dlyon@fairfield.edu, riley.fitzsimmons@student.fairfield.edu

An Appstore for Java

By Douglas Lyon, Riley Fitzsimmons

ABSTRACT

We describe a new App store for Java that makes use of OpenWebstart technologies to deploy Java to Windows, Mac and Unix. Apps are signed with certificates that are sourced from a Certificate Authority. The store is responsible for establishing developer identity. Once identity is established, the developer pays one low yearly fee to distribute up to 6 applications. The signing is done in a semi-automatic manner. If clients claim that harm is done by an application, the application is isolated until an examination is performed. Trusted Jar files can be run outside of the "sandbox" and thus be given full access to the target system. The Java app store addresses a subproblem of the Initium project. Initium is a Latin word that means: "at the start.

I. INTRODUCTION TO CODESIGNING CERTIFICATES

For applications to be trusted by the DocJava, Inc. App store, developers must identify that they are who they say that they are. Credentials are presented to notaries (picture government id, i.e., drivers' license or passport or similar) in order to have their identity verified. Proving the originator is the author of code does NOT prevent the developer from writing harmful code. When joining the DocJava Developer Program and accepting the Program License Agreement, developers agree to ensure that their software is safe and secure for their users. To deploy our apps we sign the Jar files using standard codesigning tools [JDK8].

Code signing is complex and expensive. Each Certificate Authority (CA) has a different process for retrieving certificates and importing them into keystores. When retrieving the certificate, a CA will send multiple different certificates that will all be chained together to create a keychain. Among these different certificates are a root certificate. A root certificate is a special certificate from a CA that verifies that the chain comes from a verified source. A CA sells a code-signing certificate. Certificates from Thawte, Comodo, and Sectigo all work for signing a Jar. However, once a certificate is purchased there are additional requirements that need to be met in order to obtain a certificate. For Jar signer to sign a Jar, you must generate a public and private key-pair. If a public and private key are not present in a keystore chained to the correct certificates, Java will label the Jar file self-signed and it will be untrusted.

II. ID VALIDATION

One of the problems with signing apps for others on an app store is proving that the developer is who they say that they are. There are several means of validating identification. For example, Sectigo uses a copy of government issued ID, such as a driver's license, passport, national ID or military ID that contains a name that matches the admin contact identity on the order. Considering that these are scanned copies, it would be an easy matter to falsify these ID artifacts. A phone number also appears in a Dun and Bradstreet listing, and Sectigo attempts to further validate the listing by placing a phone call to the published numbers, in the case of a company. Also used, by some certificate authorities is a copy of a utility bill. Again, since this is a scanned copy, it would be an easy matter to falsify this artifact. Certum, a Polish-based CA, will issue a certificate to open-source developers. For this, you need a valid email address and a place where an open-source project that you control resides and both sides of a scanned ID card. This is a pretty low bar, for ID validation and not hard to fake.

In the UK there is a government-run databased of all registered business in the UK called "Companies House". Typically, a fax from the listed phone number in that database is sufficient to validate that I represent the company. Middlemen are used to purchase certificates issued to fake companies on that database and these are resold to malicious actors that engage in mass spam botnet types of criminal malware activity.

The DocJava Appstore approach to validation is more rigorous. The application requires name, data of birth, and a national identification number. This can consist of a driver's license number, social security number or passport number. You must also enter an e-mail address. Other information needed includes: your phone number, your mother's maiden name, your father's middle name, what is the make of your fridge, etc. You are then asked to obtain a notarized copy of a form indicating that you have shown your ID to a notary that can be reached by phone and appears in a business phone director (branch manager at a bank, UPS store, etc.). This is done with 3 notaries, who are all contacted. This enables us to have reasonable assurance that you are who you say you are. The question of how good this system is for validation of ID remains open and we may need to modify it in the futures to change the level of rigor used (depending on abuse).

III. GENERATING AND IMPORTING CODESIGNING CERTIFICATES

This section outlines the specific process of generating a Sectigo code-signing certificate on Mac OS. For example, Sectigo requires that we create an account with *codesigncert.com.* We click on the GENERATE NOW words underneath of the *vendor#* column as shown in *Figure 1*.

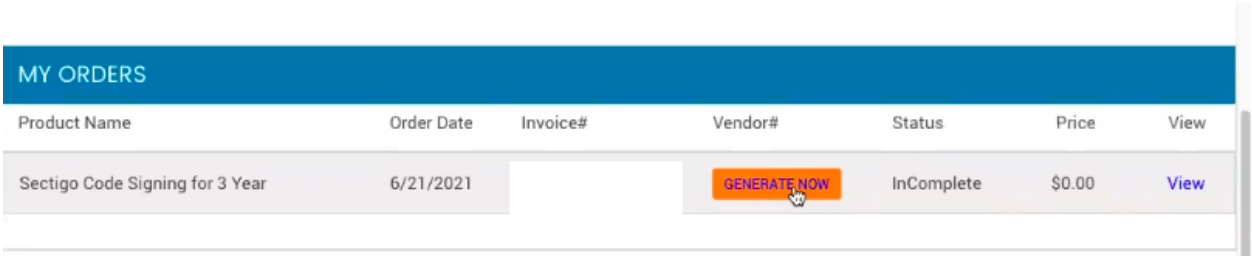

Fig. 1 Image of Sectico Generate button to generate code-signing certificate

We fill out the rest of the information accordingly for the correct company including the certificate signing request. After this has been completed, we log into the account and scroll down to your recent code signing orders. We find the certificate under "My Orders" and click view to the right of the cert.

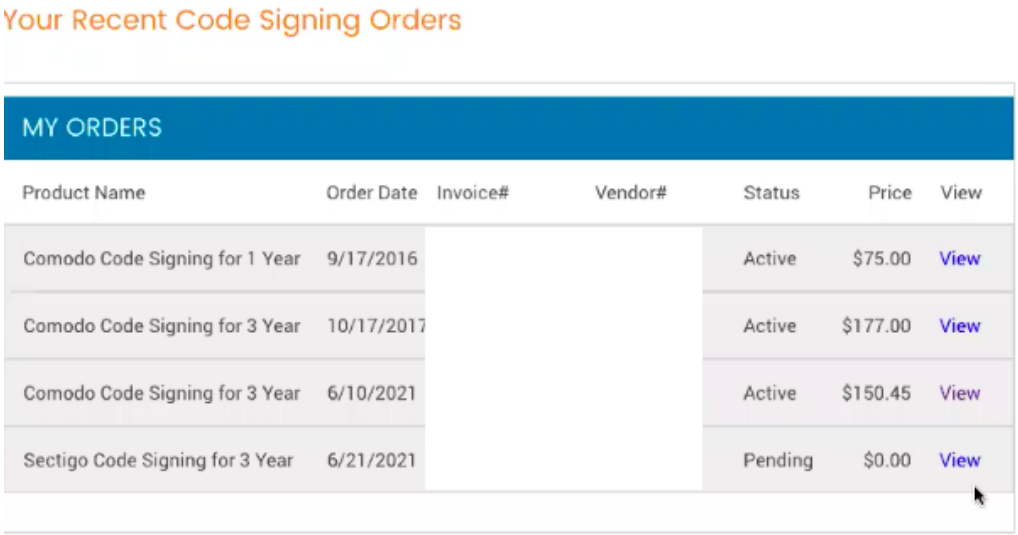

Fig. 2 The My Orders Dialog Box.

Figure 2 shows an image of the view dialog. This is where we can view the encoded certificate from Sectigo.

Fig. 3 uid and password Dialog Box

The UID and password dialog box is used to log into the vendor. Once logged into the vendor, we download the zip file to retrieve the certificates. The zip file contains 4 different certificates: one matching the certificate created, a root certificate, a public code signing certificate, and a public code signing root certificate. We then import the public code signing root certificate and then insert as shown in Figure 4.

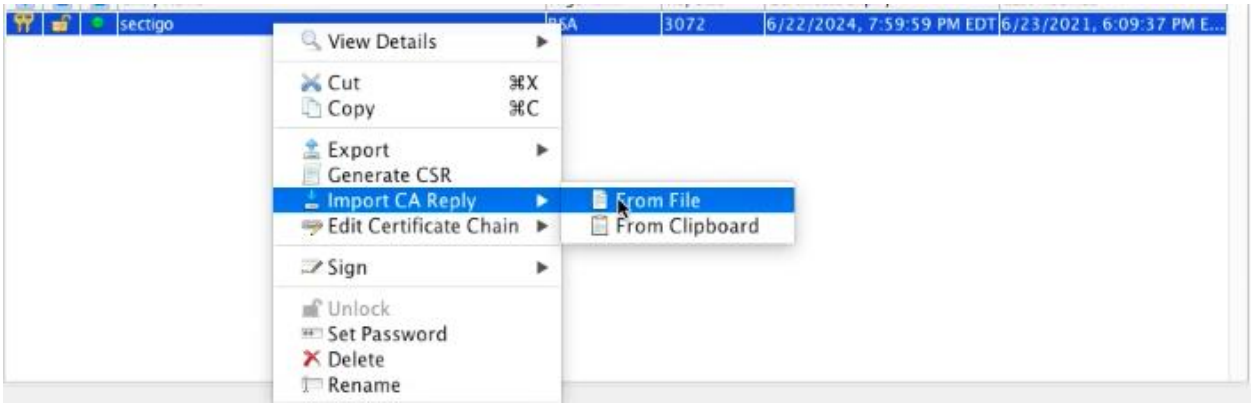

Fig. 4 Key Store certificate input menu Item

We import the hexadecimal response and a dialog box is displayed as shown in Figure 5.

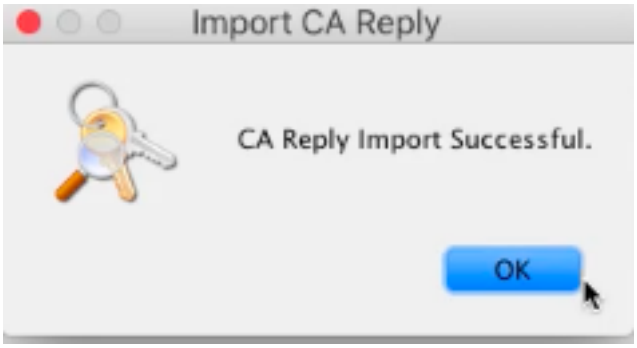

Fig. 5 Response that should appear if the certificate was successfully imported into the keystore.

Figure 6 shows how to append the *keychain* in the *keystore* to include the generated.

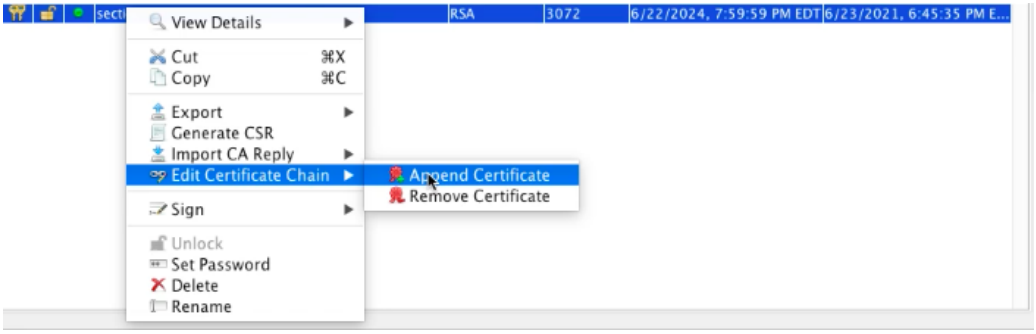

Fig. 6 Shows how append a certificate

Certificate order is important. First append the *Public Code Signing Certificate* followed by the *Public Code Signing Root Certificate* and then the *Root certificate.* This can be done by appending to the *keychain* using the appended certificate. By completing this process, the DocJava Appstore can provide valid code-signing certificates that can be applied to any application within the web of trust. Once this service is provided, the certificate which has been created vouches for the credibility and integrity of the application it has been signed to.

IV. SIGNING A JAR FILE

When signing a Jar file, the first thing that must be done is to make sure that the Jar file is unsigned. If it is not unsigned the expired certificate will be removed from the Jar. The DocJava

Appstore will sign a given application with a valid code signing certificate once it has been approved and accepted into the web of trust. Below is a sample csh script for signing Jar files:

```
# Usage: for_each_file file(s)
if ($#argv == 0) then # arguments?
 echo "Usage: $0 file(s)"
 exit 1
endif
rm -rf foo
mkdir foo
foreach file (*.Jar) # expand argument word list
 echo $file
 mv $file foo
 cd foo
 Jar -xf $file
 rm -rf META-INF
 rm $file
 Jar -cf $file .
 Jarsigner -storepass password $file docjava
 mv $file ..
 cd ..
end
```

Fig. 7 csh script for resigning a Jar file

Figure 7 creates a temporary directory named *foo*. The script will echo each Jar file in the directory and then move the file to the temporary directory and change the present working directory to the temp directory. It then removes the *META-INF* of the file and removes the file. The script then uses *Jarsigner* to sign the Jar with your chained certificate and moves the file back to the original directory.

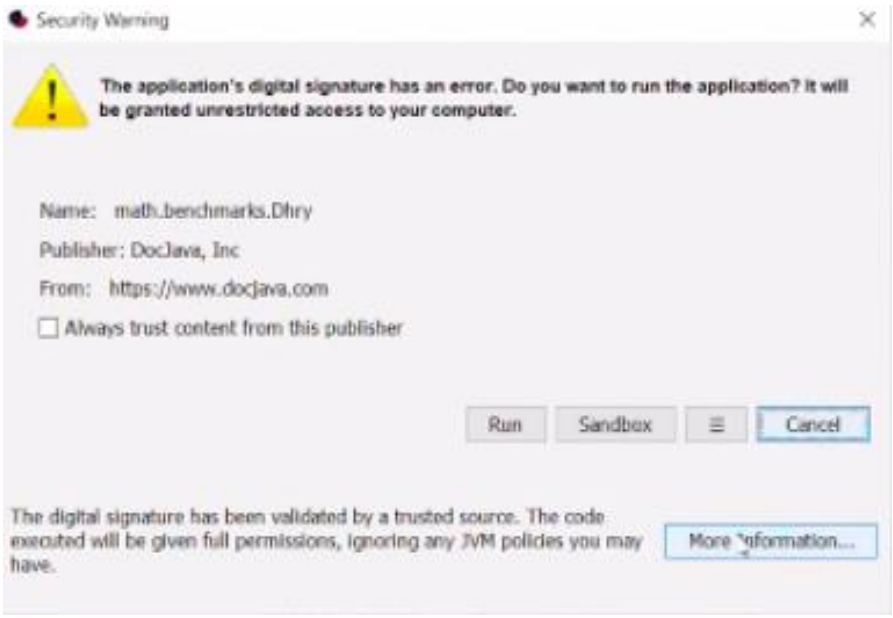

Fig. 8 Warning generated for a self-signed or unsigned Jar

Once the Jar file has been successfully signed by the DocJava Appstore it is added to the store.

V. DEPLOYMENT OF APPLICATIONS

We run applications using OWS (OpenWebStart). [OWS]
Clients need the OWS application installed on their system. If JDK8 is installed, jnlp files will be associated with the proprietary version of Oracle Java Web Start. To automatically open these files with OWS instead of Java Web Start we change the file association in the finder on Mac OS and in the file explorer on Windows OS. OpenWebStart identifies native libraries for Java3D applications which is shown in *Figure 9*.

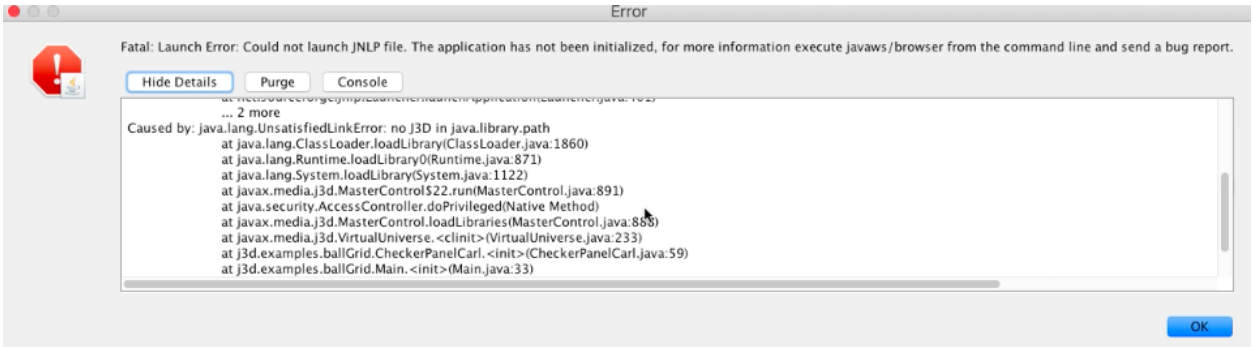

Fig. 9 Java3D errors

Via the href tag in the jnlp files, as show in Figure 10.

```
<resources>
 <j2se version="1.8+"
  initial-heap-size="64m"
  max-heap-size="384m"/>
     <jar href="libs/jogl/j3dcore.jar" download="eager"/>
     <jar href="libs/jogl/j3dutils.jar"  download="eager"/>
     <jar href="libs/jogl/vecmath.jar"  download="eager"/>
     <jar href="libs/jogl/jogamp-fat.jar"     download="eager"/>
 <jar href="j3d.examples.ballGrid.Main.jar" />
</resources>
```

Fig. 10 Additional Jar href tags

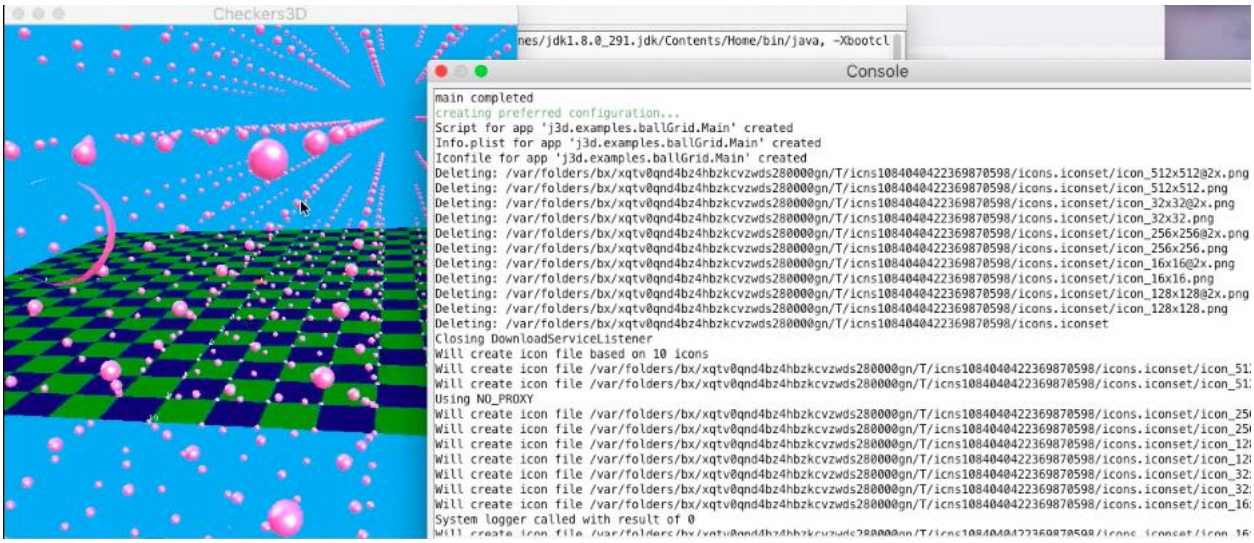

Fig. 11 Shows a screenshot of a Java3D application being run from the Appstore

Figure 11 shows an interactive Java 3D application deployed from the DocJava Appstore.

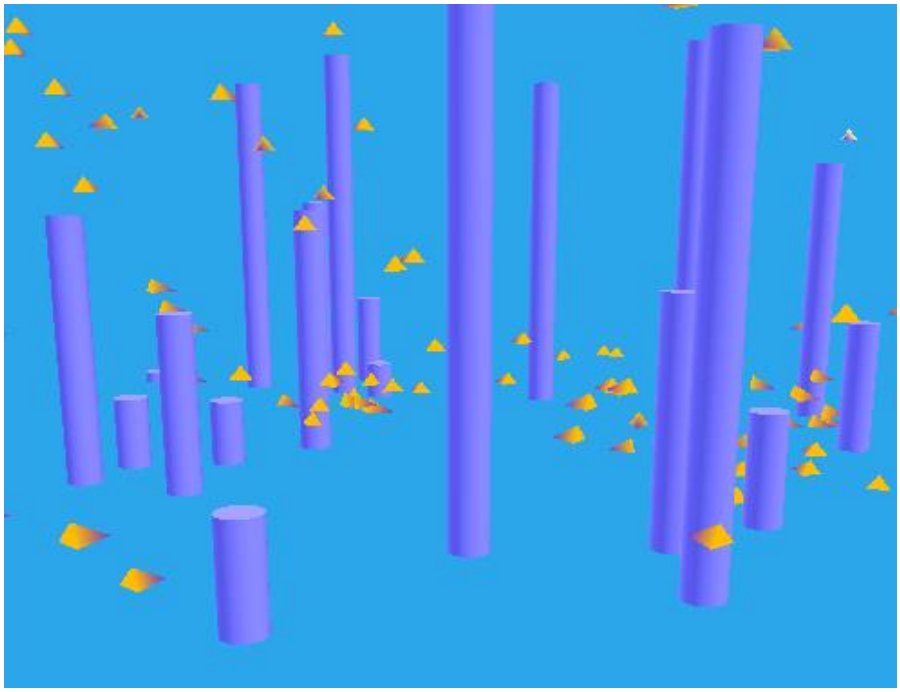

Fig. 12 Java3D and Boids

Figure 12 shows another of several Java3D applications, called Boids. We have ported several of the legacy APIs to Java Webstart in order to facilitate deployment to the app store.

VII. CONCLUSION

We have built a new app store for Java to assist developers with monetization and deployment. Developers are encouraged to release their premium apps to give evaluators a trial or feature restricted experiencing prior to requesting monetary support. Apps on our store deploy as trusted applications on the desktop except for an initial “unknown developer” dialog on the mac. Developers have their identity verified prior to App deployment. The app store depends on OpenWebStart being deployed on the client’s system. This is a critical first step and need only be done once.

REFERENCES

[JDK8] https://www.oracle.com/java/technologies/javase/javase8u211-later-archive-downloads.html.

[Lyon 2004A] “Project Initium: Pragmatic Deployment” by Douglas A. Lyon, Journal of Object Technology, Vol. 3, No. 8, September-October 2004, pp. 56-69

[Lyon 2004B] “The Initium X.509 Certificate Wizard” by Douglas A. Lyon, Journal of Object Technology, vol. 3, no. 10, November-December 2004, pp. 75-88. http://www.jot.fm/issues/issue_2004_11/column6/

[Lyon 2008] “I Resign! Resigning Jar Files with Initium.” by Douglas A. Lyon, Journal of Object Technology, Vol. 7, No. 4, May-June 2008, pp. 19-27

[OWS] https://openwebstart.com/download/

Author Information

Dr. Douglas Lyon, is a professor of Biomedical and Electrical Engineering at Fairfield University. He is a licensed professional engineer, a senior member of the IEEE and President of DocJava, Inc. Dr. Lyon received the PhD, MS and BS degrees in computer and systems engineering from Rensselaer Polytechnic Institute and has worked at AT&T Bell Laboratories and the Jet Propulsion Laboratory. He has authored four books and over 50 journal publications.

Riley Fitzsimmons is an undergraduate student at Fairfield University pursuing a BS in computer science. He is a member of the Dean’s List and the varsity men’s lacrosse team.