

Let's Encrypt Apache Tomcat*

* Full disclosure: Tomcat will not actually be encrypted.



Christopher Schultz Chief Technology Officer Total Child Health, Inc.

* Slides available on the Linux Foundation / ApacheCon2017 web site and at http://people.apache.org/~schultz/ApacheCon NA 2017/Let's Encrypt Apache Tomcat.odp

Apache Tomcat

- Java Web Application Server
- Implements J2EE API Specifications
 - Java Servlet
 - Java ServerPages (JSP)
 - Java Expression Language (EL)
 - Java WebSocket

Apache Tomcat

- Provides Services
 - Java Naming and Directory Interface (JNDI)
 - JDBC DataSource, Mail Session via JNDI
- Provides Client Connectivity
 - HTTP, AJP
 - HTTPS using SSL/TLS

Transport Layer Security (TLS)

- Formerly known as "Secure Sockets Layer"
- Provides authenticated and confidential conversations
 - Client and server can authenticate each other
 - Conversation is encrypted

Transport Layer Security

- Client and server negotiate a "cipher suite"
 - Protocol (e.g. TLSv1, TLSv1.2, TLSv1.3, etc.)
 - Authentication (X.509 with RSA/DSA)
 - Key exchange (RSA, DHE, ECDHE, etc.)
 - Bulk encryption algorithm (e.g. AES, 3DES, CHACHA20, etc.)
 - Message authentication code (e.g. SHA-1, SHA-2, etc.)

Public Key Infrastructure

- Delegated Trust Model
 - Server produces certificate
 - Server authenticates to Certificate Authority (CA)
 - Certificate Authority signs Server's certificate
 - Server presents CA-signed certificate to client when a client initiates a connection
 - Client trusts the Certificate Authority
 - Client therefore trusts Server

Public Key Infrastructure

- Certificate Authorities
 - Have nearly universal (client) trust
 - Provide multiple levels of authentication
 - Domain-Validated
 - Extended Validation (EV)
 - Require human interaction for requests, issuance
 - Issue certificates for several years
 - Charge a fee for a issuance

Let's Encrypt

- Wanted widespread TLS
 - Free
 - Easy
 - Makes the Web a safer place
- Questioned CA's
 - Signing-request and issuance processes
 - Fees for freely-available crypto
- Built a better mousetrap

Let's Encrypt

- Near-universal trust
 - Cross-signed certificate from IdenTrust
- Provides a single level of authentication
 - Domain-Validated
- Requires automated interaction for requests, issuance
- Issues certificates valid for 60-day intervals
- Charges no fee for issuance

Let's Encrypt

- Not replacing CAs
 - No Extended-Validation certificates
 - No code- or email-signing certificates
 - No wildcard certificates
 - SANs are supported
- Merely reduces the financial barrier for mundane TLS to zero

The Plan

- Once
 - Request a certificate from Let's Encrypt
- Periodically (~50 day intervals)
 - Request a certificate renewal
 - Deploy the new certificate into Tomcat

The Plan

- Request a certificate from Let's Encrypt
 - Easy: use EFF's certbot tool
- Periodically request a renewal
 - Easy: Use cron + EFF's certbot tool
- Install the new certificate into Tomcat
 - Not straightforward

Tomcat Troubles

- Tomcat usually doesn't bind to port 80
 - Might be tricky to renew certificates
- Tomcat uses Java Keystores
 - certbot produces plain-old PEM files
- Tomcat has no "graceful reload"
 - httpd has this, and certbot uses it

Tomcat Troubles

- Port binding
 - jsvc
 - iptables
- Java Keystores
 - Can import PEM files
- Tomcat reloads
 - Can be done
 - Short downtime
 - Will kill in-process requests

- iptables
 - More than just a firewall
 - Can perform routing and forwarding
 - Need a few commands to redirect port 80 \rightarrow 8080

- iptables magic sauce
 - NAT PREROUTING 80 \rightarrow 8080
 - NAT OUTPUT 8080 \rightarrow 80
 - NAT PREROUTING 443 \rightarrow 8443
 - NAT OUTPUT 8443 \rightarrow 443
 - Also may require:
 - FILTER FORWARD 80 ACCEPT
 - FILTER FORWARD 443 ACCEPT

- iptables magic sauce
 - HTTP
 - iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
 - iptables -t nat -A OUTPUT -o lo -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
 - HTTPS
 - iptables -t nat -A PREROUTING -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 8443
 - iptables -t nat -A OUTPUT -o lo -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 8443

- iptables magic sauce
 - Also might need
 - iptables -A FORWARD -p tcp -m tcp –dport 80 -j ACCEPT
 - iptables -A FORWARD -p tcp -m tcp -dport 443 -j ACCEPT

- Now we can run certbot-auto to get a new certificate
 - certbot-auto certonly --webroot \
 --webroot-path "\${CATALINA_BASE}/webapps/ROOT" \
 -d www.example.com \
 --rsa-key-size 4096

- Start with self-signed certificates
 - keytool -genkeypair -keystore conf/keystore.jks.1
 -alias tomcat -keyalg RSA -sigalg SHA256withRSA
 -keysize 4096 -validity 10
 - Hostname: localhost
 - Organizational Unit: Keystore #1

- Generate a second keystore
 - keytool -genkeypair -keystore conf/keystore.jks.2
 -alias tomcat -keyalg RSA -sigalg SHA256withRSA
 -keysize 4096 -validity 10
 - Hostname: localhost
 - Organizational Unit: Keystore #2

- Symlink conf/keystore.jks.1 \rightarrow conf/keystore.jks
- Configure the connector in Tomcat
 - <Connector port="8443" keystoreFile="conf/keystore.jks" ... />
- Start Tomcat
- Verify connection
 - openssl s_client -no_ssl3 -connect localhost:8443

- Remove existing symlink
- Symlink conf/keystore.jks.2 \rightarrow conf/keystore.jks
- Now what?

- Tomcat
 - Exposes Connectors via JMX





openssl s_client -no_ssl3 -connect localhost:8443

- Manual Deployment
 - Inconvenient (VisualVM in production?)
 - Time-consuming
 - Required with irritating frequency
 - every 8 weeks
 - for every server
 - Doesn't scale

- Automation is Required
 - 1. Renew certificate from Let's Encrypt (certbot)
 - 2. Build a new Java keystore (openssl/keytool)
 - 3. Bounce Tomcat's Connector

Let's Encrypt Renewals

- Invoke certbot-auto renew
- Celebrate!

Build a new Java Keystore

- Package server key and certificate into PKCS#12 file
 - openssl pkcs12 -export -in [cert] -inkey [key] -certfile [chain] -out [p12file]
- Import PKCS#12 into Java Keystore
 - keytool -importkeystore -srckeystore [p12file] -destkeystore conf/mykeystore.jks
- Celebrate!

Bounce Tomcat's Connector

- Tomcat Manager to the Rescue
 - JMXProxyServlet
- Enable Manager Application
 - Need to configure a <Realm>
 - Security!
 - Need to configure a second <Connector>
 - Don't pull the rug from underneath yourself

Bounce Tomcat's Connector

- Stop Connector
 - curl http://localhost:8080/manager/jmxproxy?invoke=Catalina %3Atype%3DConnector%2Cport%3D8443&op=stop
- Start Connector
 - curl http://localhost:8080/manager/jmxproxy?invoke=Catalina %3Atype%3DConnector%2Cport%3D8443&op=start
- Celebrate

Automated Deployment

- Scripting* will set you free
 - certbot-auto renew
 - openssl pkcs12 -export -in [cert] -inkey [key] -certfile [chain] -out [p12file]
 - keytool -importkeystore -srckeystore [p12file] -destkeystore conf/mykeystore.jks
 - curl http://localhost:8080/manager/jmxproxy?invoke=Catalina%3Atype %3DConnector%2Cport%3D8443&op=stop
 - curl http://localhost:8080/manager/jmxproxy?invoke=Catalina%3Atype %3DConnector%2Cport%3D8443&op=start

^{*} The actual script has a lot more detail that won't fit here.

- Between stop and start, Tomcat is not accepting any requests
- Stopping the connector immediately terminates all in-use connections

- Between stop and start, Tomcat is not accepting any requests
- Stopping the connector immediately terminates all in-use connections
- What about "graceful" re-initialization?

• SSL engine *can* be swapped-out from a configured Connector

• SSL engine *can* be swapped-out from a configured Connector ... but it requires additional tracking of in-flight requests using the SSL engine.

• SSL engine *can* be swapped-out from a configured Connector

... but it requires additional tracking of in-use connections using that SSL engine.

• SSL engines must be cleanly shut-down *after* all in-use connections have been closed.

- Tracking connections using SSL engines can be complicated
 - "Classic" HTTP/1 request/response is "easy"
 - Don't forget KeepAlives
 - Servlet 3+ async can be tricky
 - HTTP/2 can be tricky

- Tracking connections using SSL engines can be complicated
 - "Classic" HTTP/1 request/response is "easy"
 - Don't forget KeepAlives
 - Servlet 3+ async can be tricky
 - HTTP/2 can be tricky
- Must add reference-counting to SSL engine management

- Tracking connections using SSL engines can be complicated
 - "Classic" HTTP/1 request/response is "easy"
 - Don't forget KeepAlives
 - Servlet 3+ async can be tricky
 - HTTP/2 can be tricky
 - Websocket can be tricky
- Must add reference-counting to SSL engine management ... which adds overhead that some people won't need.

- Reference-counting for SSL engines
- Configurable so only users who need it suffer management overhead
- Should allow truly "graceful" connector reloads

- Bonuses
 - Allows CRL reloading (if you like that kind of thing)
 - Allows on-the-fly TLS reconfiguration
 - Protocols
 - Cipher suites
 - Allows additional certificates to be added (e.g. EC)
 - ... anything else encapsulated by the SSL engine

- Will work for all connector types
 - NIO/NIO2
 - APR

- Will work for all connector types
 - NIO/NIO2
 - APR
- Probably will land in Tomcat 9
- Probably will be back-ported to 8.5 once deemed reliable

Let's Encrypt Apache Tomcat

- Let's Encrypt provides free (beer) certificates
- Automation is required for issuance and renewal
- Tomcat presents some challenges to overcome
- Those challenges have some solutions available now
- We can improve these solutions over time



Questions

Slides available on the Linux Foundation / ApacheCon2017 web site and at http://people.apache.org/~schultz/ApacheCon NA 2017/Let's Encrypt Apache Tomcat.odp Sample code available in the same directory.