Let’s Encrypt Apache Tomcat*

* Tomcat will not actually be encrypted.

Christopher Schultz
Total Child Health, Inc.
ASF Member, Tomcat PMC, Security Team

https://people.apache.org/~schultz/ApacheCon NA 2019/Let's Encrypt Apache Tomcat.pdf

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

Why Encrypt

- Security for services that need security
  - Obvious
- Security for *users* of sites that do not need security
  - Not so obvious
  - MitM is easy
  - MitM = pwned
  - https://www.troyhunt.com/heres-why-your-static-website-needs-https/

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

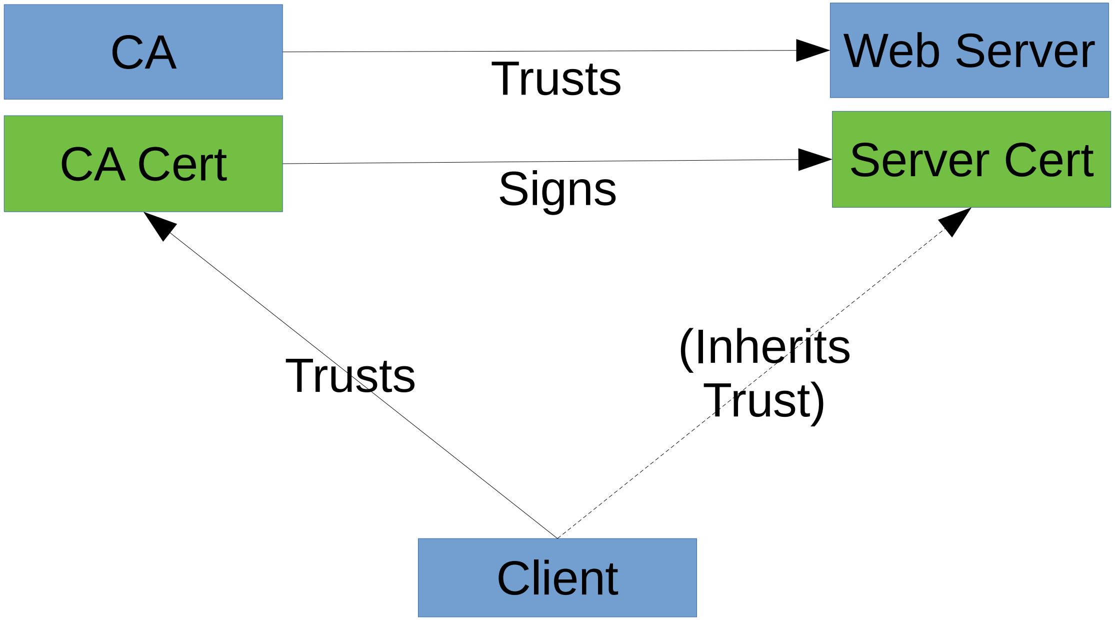
Transport Layer Security

- Client and server negotiate a “cipher suite”
  - Protocol (e.g. TLSv1, TLSv1.2, TLSv1.3, etc.)
  - Authentication (e.g. X.509 with RSA/DSA or EC)
  - Key exchange (e.g. 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



CA

Web Server

Trusts

CA Cert

Server Cert

Signs

Trusts

(Inherits Trust)

Client

Public Key Infrastructure

- Certificate Authorities
  - Have nearly universal (client) trust
  - Provide multiple levels of authentication
    - Domain-Validated (DV)
    - Organization-Validated (OV)
    - 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 (an existing CA)
  - Most browsers and OSs now include LE root certs
- Provides a single level of authentication
  - Domain-Validated
- Requires automated interaction for requests, issuance
- Issues certificates valid for 90-day intervals
- Charges no fee for issuance

Let’s Encrypt

- Not replacing CAs
  - No Organization-Validation or Extended-Validation certificates
  - No code- or email-signing certificates
- 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 Keystores
  - certbot produces plain-old PEM files
- Tomcat’s “graceful reload” isn’t super convenient
  - httpd has this, and certbot uses it

Tomcat Solutions

- Port binding
  - jsvc
  - iptables
- Java Keystores
  - Can import PEM files
- Tomcat reloads
  - Can be done
  - Without downtime
  - In-process requests will complete

Getting that first LE Cert

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

Getting that first LE Cert

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

Getting that first LE Cert

- 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

Getting that first LE Cert

- 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

Getting that first LE Cert

- 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
    ```

Reconfiguring Tomcat’s TLS

- Start with self-signed certificates

  -
    ```
    keytool -genkeypair \
    -keystore conf/keystore.p12.1 \
    -storetype PKCS12 \
    -alias tomcat -keyalg RSA \
    -sigalg SHA256withRSA \
    -keysize 4096 -validity 10
    ```
  - Hostname: localhost
  - Organizational Unit: Keystore #1

Reconfiguring Tomcat’s TLS

- Generate a second keystore

  -
    ```
    keytool -genkeypair \
    -keystore conf/keystore.p12.2 \
    -storetype PKCS12 \
    -alias tomcat -keyalg RSA \
    -sigalg SHA256withRSA \
    -keysize 4096 -validity 10
    ```
  - Hostname: localhost
  - Organizational Unit: Keystore #2

Reconfiguring Tomcat’s TLS

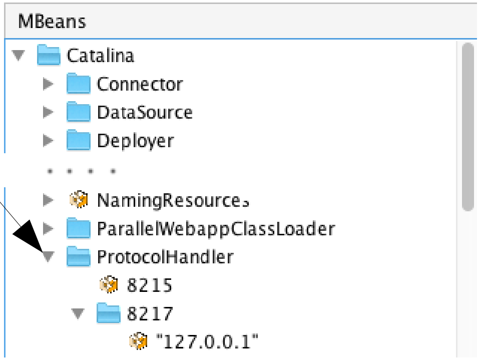
- Symlink conf/keystore.p12.1 → conf/keystore.p12
- Configure the connector in Tomcat
  - <Connector port=”8443” keystoreFile=”conf/keystore.p12” … />
- Start Tomcat
- Verify connection
  - openssl s_client -no_ssl3 -connect localhost:8443
  - openssl s_client -no_ssl3 -connect localhost:443

Reconfiguring Tomcat’s TLS

- Remove existing symlink
- Symlink conf/keystore.p12.2 → conf/keystore.p12
- Now what?

Reconfiguring Tomcat’s TLS

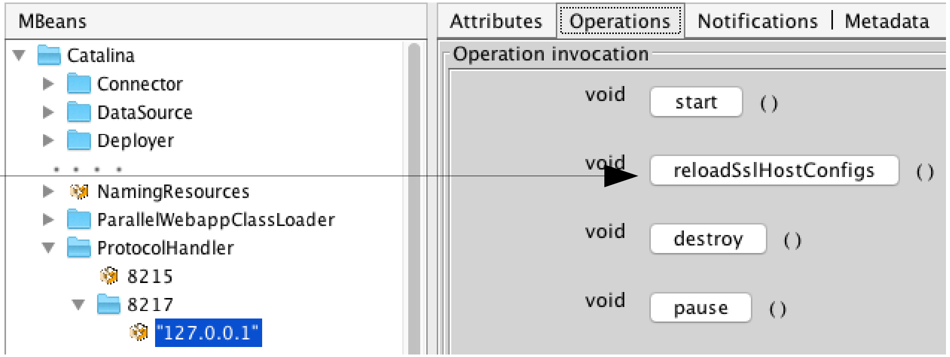
- Tomcat
  - Exposes ProtocolHandlers via JMX
- ProtocolHandlers via JMX
  - reloadSslHostConfigs
  - … in Tomcat 8.5.32+
  - … or Tomcat 9.0.??



```
MBeans
▼ Catalina
    ▶ Connector
    ▶ DataSource
    ▶ Deployer
    . . . .
    ▶ NamingResources
    ▶ ParallelWebappClassLoader
    ▼ ProtocolHandler
        8215
      ▼ 8217
          "127.0.0.1"
```

Reconfiguring Tomcat's TLS

- Connect to Tomcat via JMX
- Navigate to the proper ProtocolHandler
- Invoke the reloadHostConfigs operation
- Verify Connection
  - openssl s_client -no_ssl3 -connect localhost:443



Reconfiguring Tomcat’s TLS

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

Reconfiguring Tomcat’s TLS

- Automation is Required
  1. Renew certificate from Let’s Encrypt (certbot)
  2. Build a new keystore (openssl)
  3. Reload Tomcat’s Keystore

Let’s Encrypt Renewals

- Invoke certbot-auto renew
- Celebrate!

Build a new Keystore

- Package server key and certificate into PKCS#12 file

  -
    ```
    openssl pkcs12 -export -in [cert] -inkey [key] -certfile [chain] -out [p12file]
    ```
- Celebrate!

Reload Tomcat’s Keystore

- Tomcat Manager to the Rescue
  - JMXProxyServlet
- Enable Manager Application
  - Need to configure a <Realm>
    - Security!

Reload Tomcat’s Keystore

- Invoke reload method

  -
    ```
    curl https://localhost/manager/jmxproxy?invoke=Catalina%3Atype%3DProtocolHandler%2Cport%3D8443%2Caddress%3D%22127.0.0.1%22&op=reloadSslHostConfigs
    ```
- Celebrate

Automated Deployment

- Scripting* will set you free
  - certbot-auto renew
  - openssl pkcs12 -export -in [cert] -inkey [key] -certfile [chain] -out [p12file]
  - curl https://localhost/manager/jmxproxy?invoke=Catalina%3Atype%3DProtocolHandler%2Cport%3D8443%2Caddress%3D%22127.0.0.1%22&op=reloadSslHostConfigs

* The actual script has a lot more detail that won’t fit here.

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

Bonuses

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

Let’s Encrypt Apache Tomcat

- Let’s Encrypt provides free (beer) certificates
- Automation is required for issuance and renewal
- Tomcat is somewhat more complicated than e.g. httpd
- Those complications can be overcome

Questions

https://people.apache.org/~schultz/ApacheCon NA 2019/Let's Encrypt Apache Tomcat.pdf
Sample code available in the same directory.