Locking-Down Apache Tomcat

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

https://people.apache.org/~schultz/ApacheCon NA 2019/Locking-Down Apache Tomcat.pdf

How *not* to Architect Security

1. Build Application
2. Build Deployment
3. Sprinkle Secure On Top

Know Your Enemy

- Perform a Risk Assessment
  - No, really
- What are you trying to protect?
  - … and from whom?
- Don’t forget insider threats
- Is your *whole* internal network trustworthy?
  - Forever?

Taking a Layered Approach to Security

- Physical
- Physical Network
- OS and User/Group/ACL
- Server Software
  - Application server (Tomcat)
  - Application (your fine web application)

Physical (Quick)

- Site
- Room
- Cage

Physical Network (Quick)

- Public-Facing
- LAN-facing
- VLANs for segmentation

OS and User/Group/ACL (Quick)

- Keep OS up-to-date
  - Can’t have a secure server & application if the system is leaky
- Make the server as dedicated as possible
  - Extra stuff can jeopardize your install

OS and User/Group/ACL (Quick)

- Use a dedicated Tomcat user or users
  - Don’t use root
  - Don’t use the Tomcat user for other stuff
- Protect configuration files
  - Readable by Tomcat user
  - Writable by … nobody?
- Protect directories
  - Writable by … nobody?
  - Tomcat needs write-access to `work/` directory

Server Software

- The Real Meat
  - Tomcat configuration and services
  - Your web application

Apache Tomcat

- Reasonably Secure out-of-the-box
  - (Lack of) default credentials
  - Localhost-only user-access to privileged applications
  - Applications
  - File permissions
  - Ports (localhost::8005)
- Best way to stay secure: stay current

Apache Tomcat

- (Potentially) Dangerous Features
- Default applications
- SecurityManager
- Connectors
- Authentication
- Services (e.g. MailSession, JNDI, JDBC)
- Clustering

Tomcat’s Sharp Edges

- APR connector / tcnative
- Manager app
- Server-Side Includes
- Read/write WebDAV
- CGIServlet
- JMX

Tomcat – Remove Applications

- Do you need the Manager?
  - It’s handy for deployment
  - And monitoring
- Do you need the host-manager?
  - Really?

Tomcat + SecurityManager

- Works great!
- Your application? YMMV

Tomcat Connectors

- Choice of Network Interface(s)
- Consider IP filtering
  - Whole server
  - Per application
- Considerations for Reverse-Proxying
- TLS – could be a whole lecture*
- AJP *does not* provide encryption

* See my presentation on *Let’s Encrypt Apache Tomcat* or any of the TLS presentations on the Tomcat web site.

Tomcat Connectors

- Protect shutdown port with a password
  - `<Server port="8005" shutdown="s3kret">`
  - Not super secure; anyone can connect to your socket
- Disable shutdown port
  - `<Server port="-1">`
  - Requires signals to stop the service
- Protect the AJP port with a password
  - `<Connector port=”8009” requiredSecret=”s3kret2” ...`
  - `worker.ajp12.secret=s3kret2`

Tomcat Connectors

- Options for Secure AJP
  - “secret” is exactly as secure as HTTP Basic auth
  - stunnel works *very* well
  - ssh tunneling is possible (but stunnel is better)
- Maybe it’s time to switch to HTTPS proxying

stunnel – An aside

- stunnel wraps any connection in TLS
  - Encryption
  - Authentication
  - Including mutual authentication (ooooh)
- Use stunnel with client certificates!

stunnel – An aside

- Quick magic stunnel sauce:

```
CAFile = /path/to/trusted-client-certificates
# 4=client certificate must be present in CAFile
verify=4

[ajp]
accept=0.0.0.0:8008
connect=localhost:8009
```

```
$ cat /path/to/trusted-client-certificates
-----BEGIN CERTIFICATE-----
.......
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
.......
```

Tomcat Authentication

- Container-managed authentication and authorization
- Protects Tomcat’s bundled applications
- Can protect your application as well
- Not every application needs a third-party “Security” component added to it

Tomcat Authentication

- Back-ends
  - File-based (tomcat-users.xml)
  - JDBC-based
  - LDAP-based
  - JASPIC-based
- Credential protection
  - Hashing / KDF
  - Modern Tomcat is *much* better than old-school MD5*

* See my presentation on *Seamless Upgrades for Credential Security* on the Tomcat web site.

Tomcat Clustering

- Membership
  - Static versus Multicast
- Serialization
  - Commons-collections attack vector
  - What others might exist?

Tomcat Clustering

- Encryption
  - Use the EncryptInterceptor

```
<Channel>
  ...
  <Interceptor
    className="org.apache.catalina.tribes.group.interceptors.EncryptInterceptor"
    encryptionKey="[lots of random bytes]" />
  <Interceptor
    className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"
/>
  ...
</Channel>
```

Application

- Make good decisions, here
- Tomcat can provide some protections
  - Authentication (don’t roll your own)
  - Access control via RemoteIPValve/Filter (RemoteIPValve, etc.)
  - DOS protection via SemaphoreValve (as a base)

Application

- Additional protections Tomcat can provide
  - CSRF protection via CsrfPreventionFilter/RestCsrfPreventionFilter
  - CORS sharing via CorsFilter
  - HSTS/X-Frame-Options/X-Content-Type-Options/X-XSS-Protection via HttpHeaderSecurityFilter

Application

- Tomcat protections require
  - Understanding of the terminology
  - Understanding of the technology
  - Understanding of proper configuration

Protection Examples

- Cross-Side Request Forgery (CSRF)
  - Attack vector is you/your browser
  - Attacker tricks you into making a request to your own service – logged-in as yourself
  - Mitigation is to sign all URLs generated by your application, checking the signatures on *every request*

https://en.wikipedia.org/wiki/Cross-site_request_forgery

Protection Examples

- HSTS/X-*Whatever*
  - Gives hints to your clients in response headers
  - Need to understand the implications of proper configuration
    - And improper configuration – HSTS can bite you

https://en.wikipedia.org/wiki/Cross-site_request_forgery

Specific Recommendations

- Always run the latest release of Tomcat
  - Cannot stress this enough
  - This means administrative *competence*
  - Become an expert at deployments
    - Upgrade/downgrade
    - Split CATALINA_HOME/CATALINA_BASE helps a lot

Specific Recommendations

- Apply encryption everywhere
  - HTTPS (duh)
  - stunnel or similar for AJP
  - JDBC – usually provided by the JDBC driver
  - LDAP – Usually supported by Java via `ldaps://`
  - SMTP – Supported by JavaMail, configuration is tedious

Specific Recommendations

- Write an auditing plan
  - Realtime monitoring
  - Periodic auditing scans
- *Execute* your auditing plan!
  - As often as possible
  - Automate if possible

Tools

- ssllabs.com/ssltest
- Nagios, Ichinga, Zabbix, etc.
- nmap, Nikto, OpenVAS, Nexpose, etc.
- OSSEC, Tripwire, etc.
- Bro, Snort, Security Onion, etc.
- ModSecurity in Apache httpd (+IIS?!)

Resources

- Apache Tomcat
  - http://tomcat.apache.org/tomcat-9.0-doc/security-howto.html
  - http://tomcat.apache.org/presentations.html
  - Apache Tomcat Users’ Mailing List
- Open Web Application Security Project (OWASP)
  - https://www.owasp.org/index.php/Securing_tomcat
- Many security “guides” are out of date
  - Educate yourself, be paranoid

Questions

https://people.apache.org/~schultz/ApacheCon NA 2019/Locking-Down Apache Tomcat.pdf