Introductions

- Apache Tomcat Committer and PMC member
- Tomcat Security Team member
- Apache Software Foundation (ASF) Member
- ASF Security Team member
- SpringSource Security Team lead
- Also contribute to:
  - Commons POOL, Commons DBCP, Commons Daemon, Apache Infrastructure
Agenda

- Background
- Threats
- Keeping up to date
- Operating system
- Tomcat
- Passwords in configuration files
- Web applications
- Policy & Process
Background

- There is no one right security configuration
- Security always requires trade-offs
- Don't assess systems in isolation
- Remember:
  - Confidentiality
  - Availability
  - Integrity
- Tomcat is reasonably secure by default
- Tomcat can't protect against a fundamentally insecure web application
Threats

- Rarely receive reports of threats / attacks in the wild
- Last instance was June/July 2008
- Multiple reports of 'rouge' web applications
  - Provided shell access to server via a JSP
- Infection traced to weak passwords on internet facing manager application
  - Why was the manager application publicly accessible?
  - Why were the passwords so weak?
  - Why were the brute-force attacks not prevented?
Threats (cont.)

- More likely to see availability issues
  - Application bugs
  - Performance issues
  - Downtime for maintenance / upgrades
  - Impact is likely to be lower than for a security breach

- Slow trickle of vulnerabilities reported by security researchers
  - CVE-2008-5515 RequestDispatcher directory traversal
  - CVE-2009-0033 mod_jk DOS via HTTP headers
Keeping up to date

- Tomcat Announcements mailing list
  - announce-subscribe@tomcat.apache.org
  - Very low traffic (9 messages in three months)
  - Every release
  - Every security vulnerability

- Other sources of information
  - bugtraq, full-disclosure, ASF announcements list
Operating system

- Standard advice applies
  - Do not run Tomcat as root
  - Use a user with the minimum necessary permissions

- Listening on privileged ports
  - JSVC from Commons Daemon
  - Front using Apache httpd
  - Use iptables to map ports
Operating system (cont.)

- Does the tomcat user need to be able to anything more than read files?
  - Modify start-up scripts?
  - Modify configuration files?
  - Add new web applications?

- OS level firewall
  - Block everything by default and then allow the bare minimum
  - Outgoing http requests (often used by malicious software)
Tomcat: Deployment

- Host settings
  - autoDeploy
  - deployOnStartup
  - deployXML

- How much do you trust your web applications?
- If you don’t, you should be using a security manager
Tomcat: SecurityManager

- Runs all web applications in a sandbox
- catalina.policy file controls what each web application is permitted to do e.g.:
  - File & network access
  - Calling System.exit()
- Not widely used
- Not tested as thoroughly
- Prior to Tomcat 6.0.21 (not yet released) several EL failures with a security manager
- Likely to break your web application
Tomcat: Logging

- Enable the AccessLogValve
- If using Tomcat behind a reverse proxy (httpd, IIS, etc) enable access logging there too
- Useful diagnostics tool, not just for security breaches
- Usually configured per Host but can be configured at Engine or Context level if preferred
Tomcat: Manager application

- If you don't need it, don't deploy it
- If you do need it:
  - Limit access to known IP addresses
  - Use strong passwords
  - Don't browse untrusted sites whilst logged in to the manager application
  - Log off (close your browser) when you are done
  - Use a lock-out realm (see realm slides)
- The same guidelines apply for any administrative application
Tomcat: Realms

- Tomcat provides a number of Realm implementations
- Don't use:
  - MemoryRealm
  - JDBCRealm
- Be careful with the JAASRealm
- That leaves:
  - UserDatabaseRealm
  - JNDIRealm
  - DataSourceRealm
Tomcat: Realms (cont.)

- **UserDatabaseRealm**
  - Replacement for MemoryRealm
  - Based on `tomcat-users.xml`
  - Convoluted to update user database (via JMX)
  - Good for small numbers of fairly static users

- **DataSourceRealm**
  - Replacement for the JDBCRealm

- **JNDIRealm**
  - Effectively single threaded
Tomcat: Realms (cont.)

- Issues with all of the Realms
  - Allow unlimited authentication attempts
  - You could only have one Realm per Engine, Host or Context
- Unlimited authentication attempts permit brute force attacks
  - Made attacks in June 2008 easier
- Introduced LockOut realm to address this
  - Additional benefit was the creation of the CombinedRealm that allows multiple Realms to be used together
Tomcat: System properties

- **org.apache.catalina.**
  STRICT_SERVLET_COMPLIANCE
  - Will add a character encoding header when calling `getWriter()` - reduces exposure to UTF-7 XSS

- **org.apache.coyote.**
  USE_CUSTOM_STATUS_MSG_IN_HEADER
  - Ensure ISO-8859-1 encoding
Tomcat: Miscellaneous

■ Disable shutdown port
  ▸ `<Server port="-1" ...` /`

■ Do connectors have to listen on all interfaces?
  ▸ `<Connector address="..." ...` /

■ Pros and cons of advertising server version
  ▸ `<Connector server="Apache-Coyote/1.1` /`
Tomcat: Passwords

- server.xml or context.xml

- Why is the password in plain text?
  - Tomcat needs the plain text password to connect to the external resource
  - Encrypting the password means Tomcat would need a decryption key – back to the original problem

- Consider the risks
  - Remote information disclosure
    - Is the password usable remotely? If yes, why?
  - Local information disclosure
    - There are likely to be bigger issues to worry about
Tomcat: Passwords (cont.)

- There are potential solutions
  - Enter password at Tomcat start
  - Requires custom code
  - Password still in memory
  - Tomcat restart requires manual intervention

- Encode the password
  - Requires custom code
  - Encoding is not encryption
  - May prevent some accidental disclosures
Webapps: Authentication

■ BASIC & FORM
  ▸ Must use SSL

■ DIGEST
  ▸ SSL not required

■ CLIENT-CERT
  ▸ Already using SSL

■ Session identifier (Cookie or URL parameter also needs protection)

■ Don't switch back to HTTP from HTTPS once user has been authenticated
Webapps: SSL

- Be careful when moving from http to https
- When using a transport guarantee:
  - HTTP request (incl. body) sent in clear to Tomcat
  - HTTP request headers parsed
  - Request mapped to context
  - Transport guarantee identified
  - HTTP redirect (302) issued to https
  - HTTP request resent over https
  - HTTP response sent over https
- The request is sent in the clear
Webapps: context.xml

- Protect session cookies
  - `<Context useHttpOnly="true" ... />`
  - Will default to true from Tomcat 7.0.x onwards
- Permitting cross-context request dispatching
  - `<Context crossContext="true" ... />`
- Permitting symlinks has security side-effects
  - `<Context allowLinking="true" ... />`
- Allow access to Tomcat internals
  - `<Context privileged="true" ... />`
Webapps: Miscellaneous

- Invoker Servlet
  - Bypasses security constraints
- XSS, SQL injection etc.
  - Don't trust user input
  - Protection needs to be in the application
Policy & Process

- Review your logs
  - Access logs
  - Application logs
  - Tomcat logs
  - System (eg firewall) logs

- What do you do if you find an attempted attack?
- What do you do if you find a successful attack?
- What do you do if a Tomcat vulnerability is announced?
Questions?

- markt@apache.org
- users@tomcat.apache.org
- http://people.apache.org/~markt/presentations

- mark.thomas@springsource.com
- http://www.springsource.com/webinars