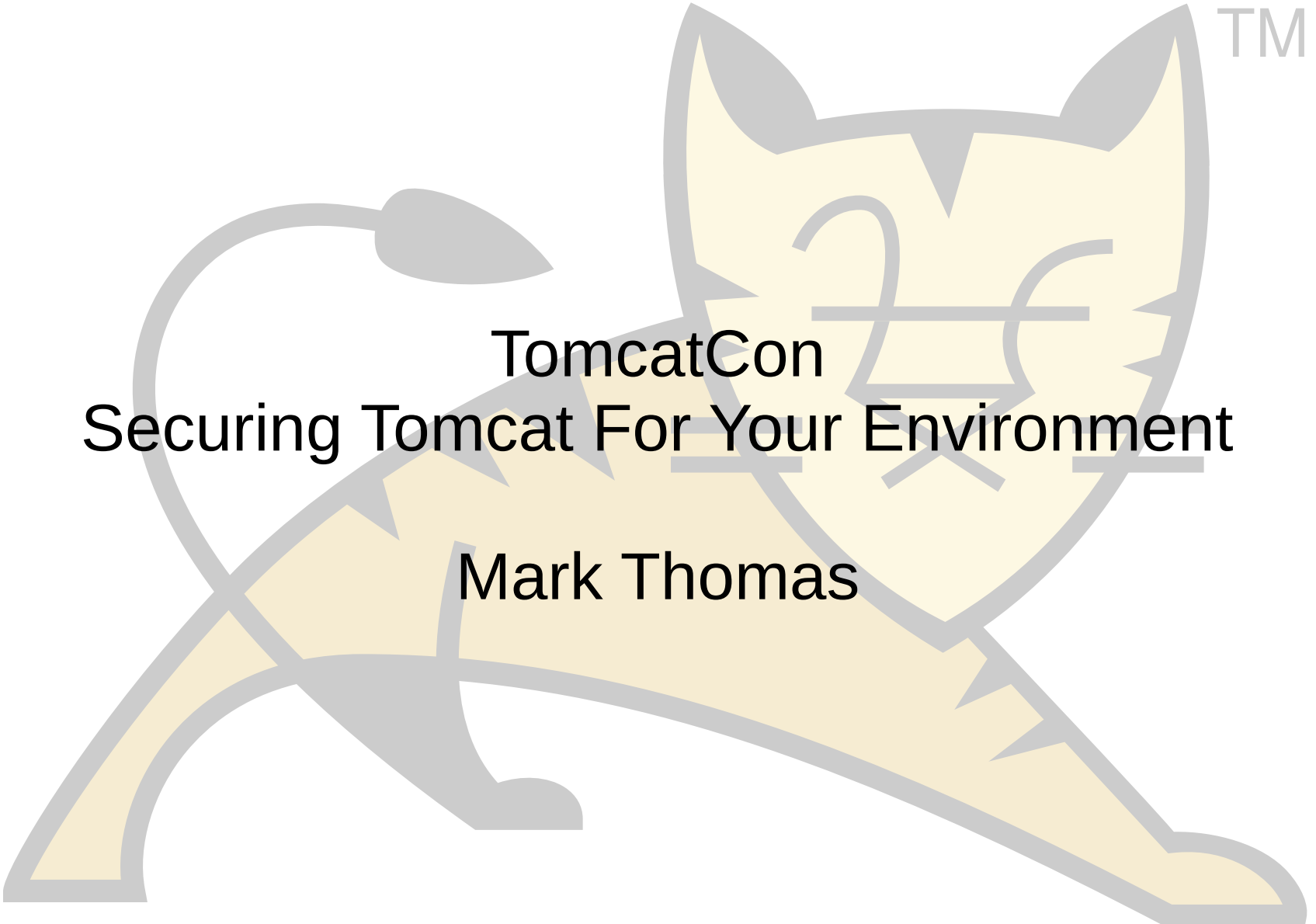
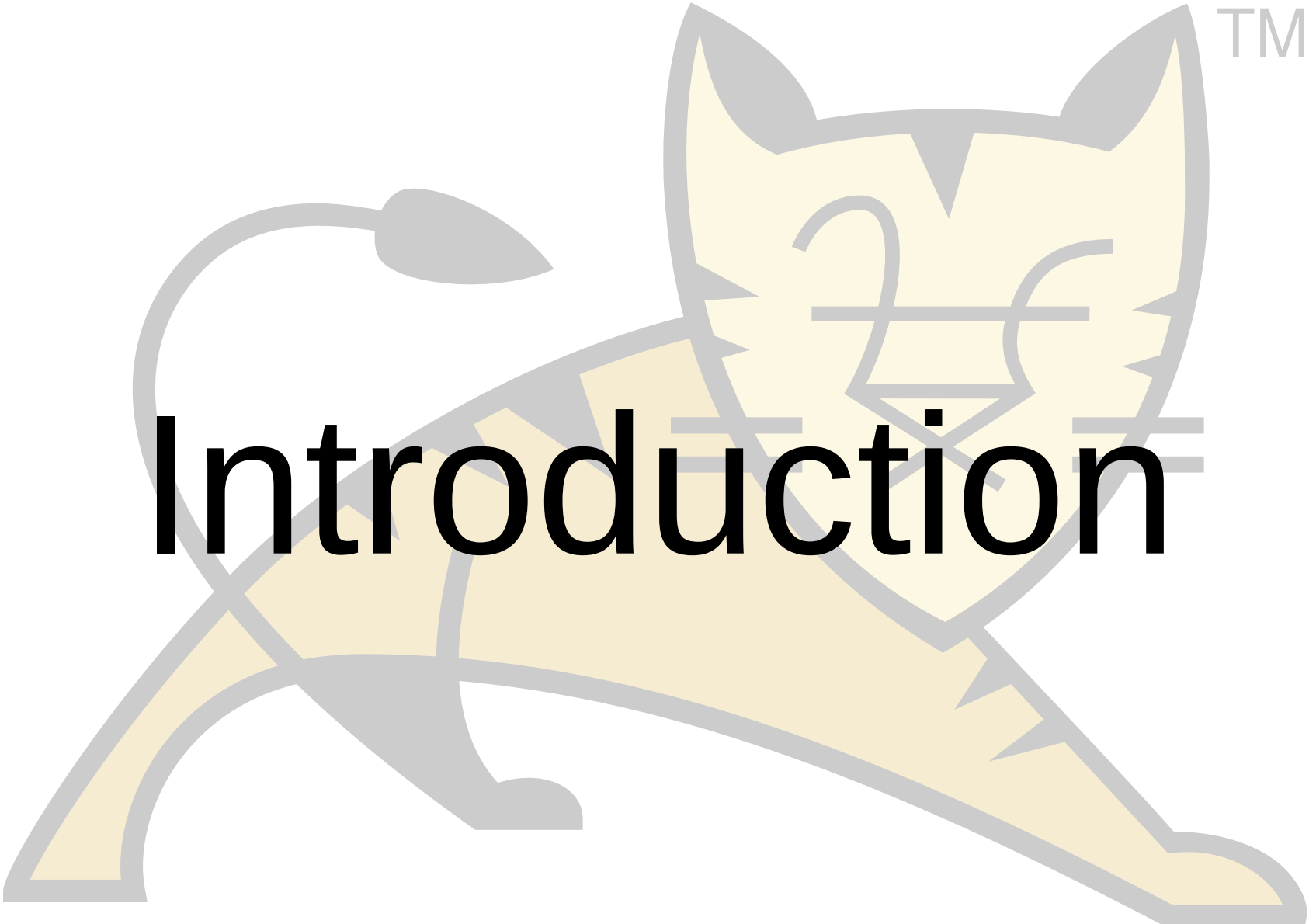


TomcatCon
Securing Tomcat For Your Environment

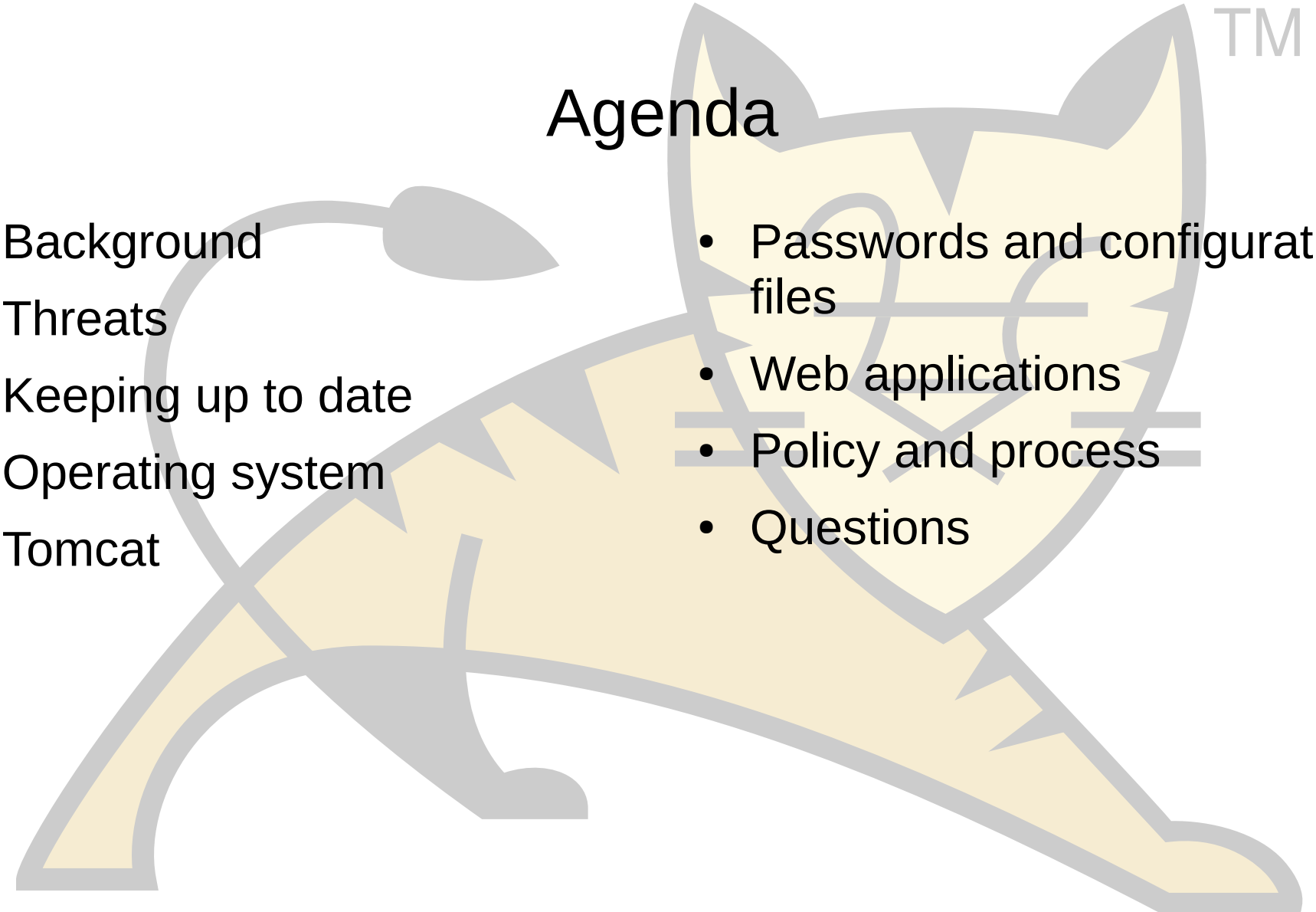
Mark Thomas



Introduction

Agenda

- Background
- Threats
- Keeping up to date
- Operating system
- Tomcat
- Passwords and configuration files
- Web applications
- Policy and process
- Questions



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
- 2014-06 malicious files created in bin directory
  - No further details provided
- 2011-11 malicious path parameters
  - Unable to reproduce
- 2010-11 response mix-up
  - Some follow-up but went silent before details were provided
- 2008-06
  - Brute force attacks against the Manager app

Threats

- Bugs with security implications are more likely
  - Send file CVE-2017-5647, CVE-2017-5651, CVE-2016-8745
- Slow trickle of vulnerabilities reported by security researchers
  - 2016: 16 2015: 5 2014: 11 2013: 6
- Vulnerabilities in dependencies
  - OpenSSL
  - NSIS
  - JRE

Threats

- Far more likely to see availability issues
  - Tomcat bugs
  - Application bugs
  - Downtime
  - Performance issues
  - Impact is typically lower than for an exploited vulnerability

Keeping up to date

- Tomcat Announcements mailing list
  - announce-subscribe@tomcat.apache.org
  - Very low traffic (15 messages in three months)
  - Every release
  - Every security vulnerability
- Other sources of information
  - ASF announcements list, Twitter
  - oss-security@lists.openwall.com, bugtraq@securityfocus.com

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
- Occasionally find bugs – security exceptions in Tomcat code
- Likely to break your web application

Tomcat: Logging

- Use the AccessLogValve (enabled by default)
- 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 (default is localhost only from 8.5.x)
  - 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 (enabled by default)
- 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
  - JASPIC

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
  - Multi-threaded 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
  - Removed in 9.0.x onwards (status messages removed)
  - 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” />
  - Not sent by default from Tomcat 8.5 onwards

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 (IntrospectionUtils.PropertySource)
  - 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 (inc 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

- 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” … />
  - Some features (e.g. CGI) require this

Webapps: Miscellaneous

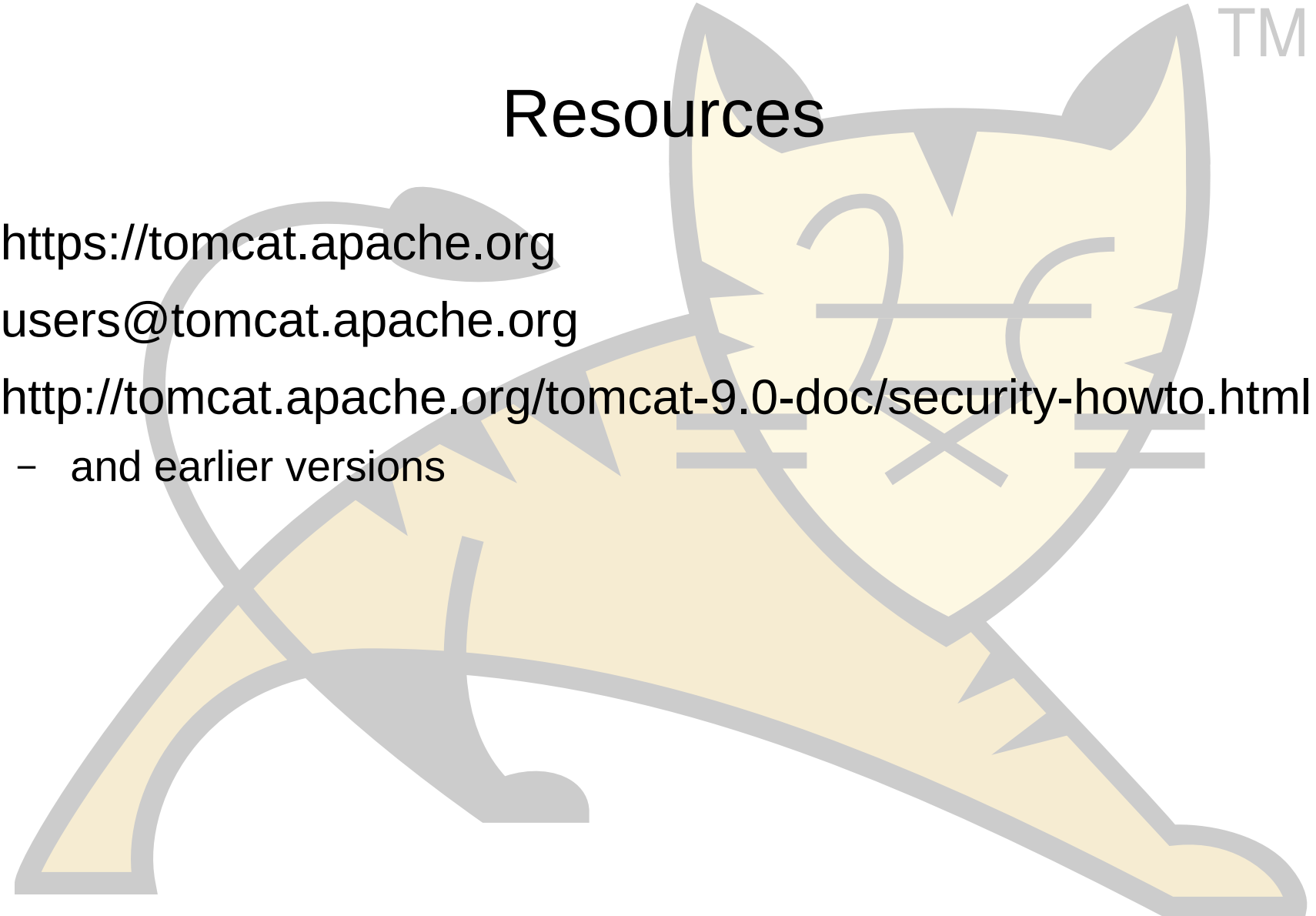
- Invoker Servlet
  - Hopefully seen the last of this
  - Bypasses security constraints
- XSS, SQL injection etc.
  - Don't trust user input
  - Protection needs to be in the application
- Java Deserialization
  - Must filter the allowed classes with a white list

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?

Resources

- https://tomcat.apache.org
- users@tomcat.apache.org
- http://tomcat.apache.org/tomcat-9.0-doc/security-howto.html
  - and earlier versions



Questions