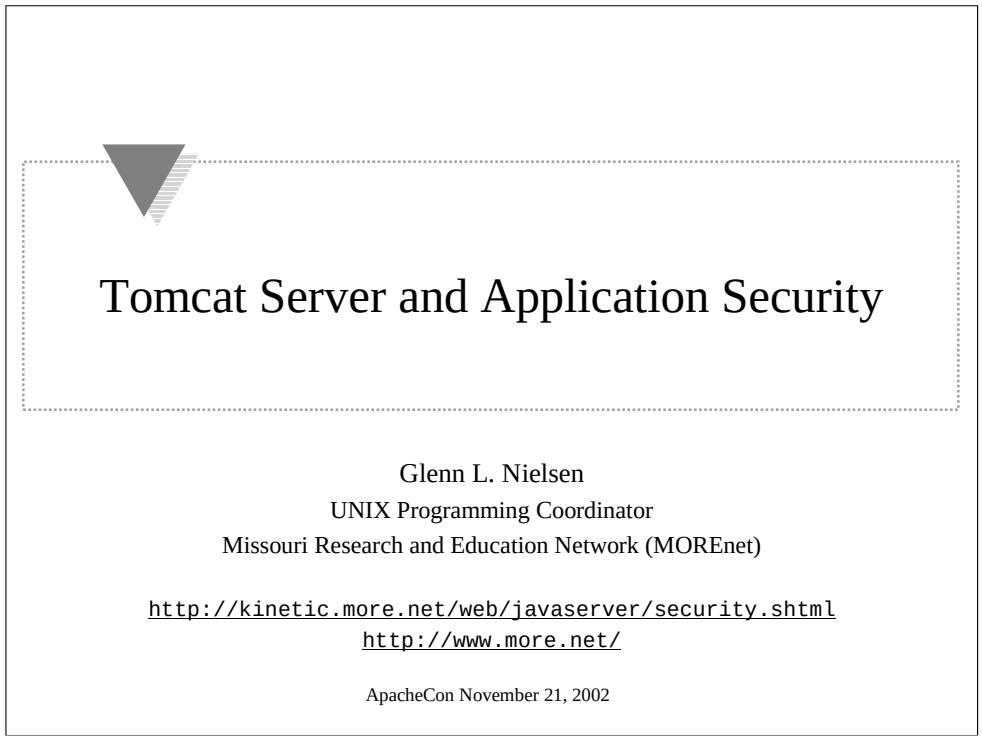


Notes

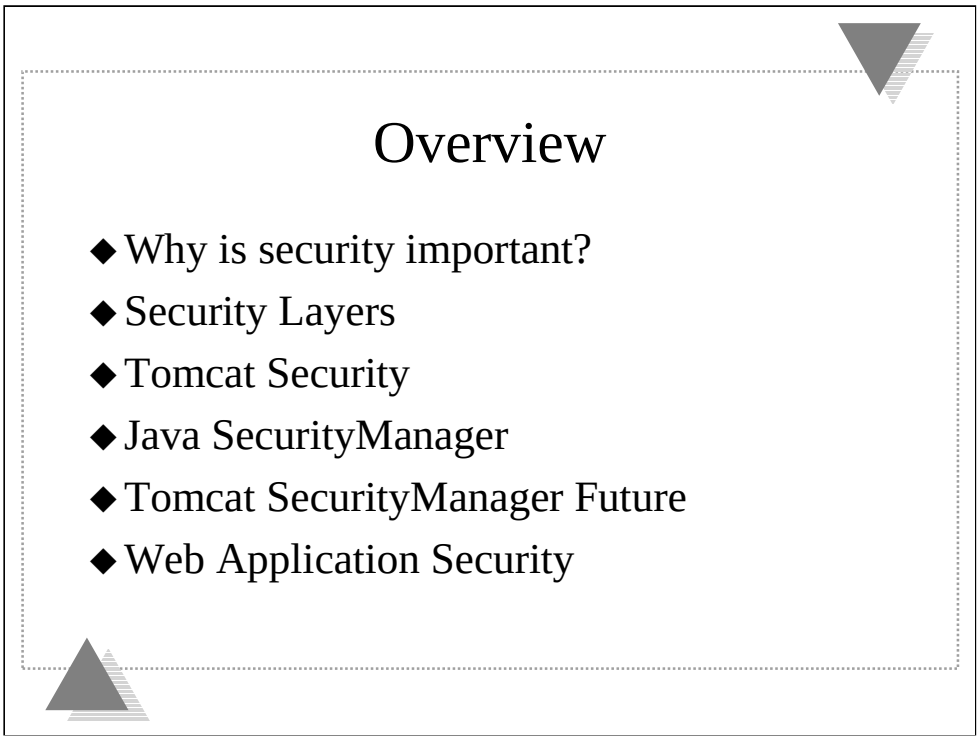
Is your Tomcat server and the applications it hosts secure from being compromised or defaced?

A Tomcat server running without a security policy is vulnerable to trojan Java packages, JSP tag libraries, and web applications.

The security policy determines the extent to which you trust JSP pages and web applications, and everyone who has permission to install them.

This presentation will address the ways insecurities can occur and methods which can be used to increase security.

We will only cover those security issues related to using Tomcat with Apache. This presentation doesn't cover general security or Apache Web Server security.



Notes

Why is security important?

- A brief look at security statistics and sources of information about security.

Security Layers

- The methods used to secure a site.

Tomcat Security

- Show methods and techniques to improve the security of your Tomcat servlet container.

Java SecurityManager

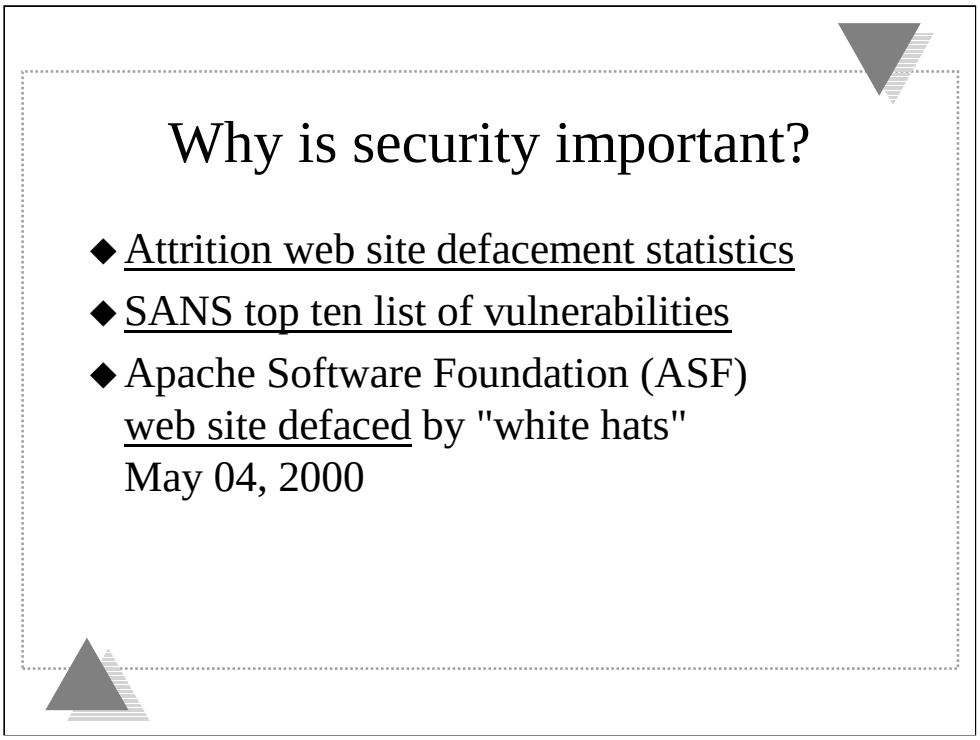
- How the Java SecurityManager works, the Java SecurityManager implementation in Tomcat 3.2 and Tomcat 4, and configuring Tomcat security policies.

Tomcat SecurityManager Future

- Tomcat 5
- Experimental XML-based policies

Web Application Security

- Web application security issues and how they can be addressed in the design of your web application.



Notes

Attrition Web Site Defacement Statistics

•We have all read about web site defacements in the news. As you can see, the trend is up. I don't want to be a statistic. The best way to prevent the site you are responsible for from becoming a statistic is knowledge about security issues. Statistics data ends December 21, 2000.

•http://www.attrition.org/mirror/attrition/defacements-graphs.html

System Administration and Network Security (SANS) Top Ten Security Vulnerabilities

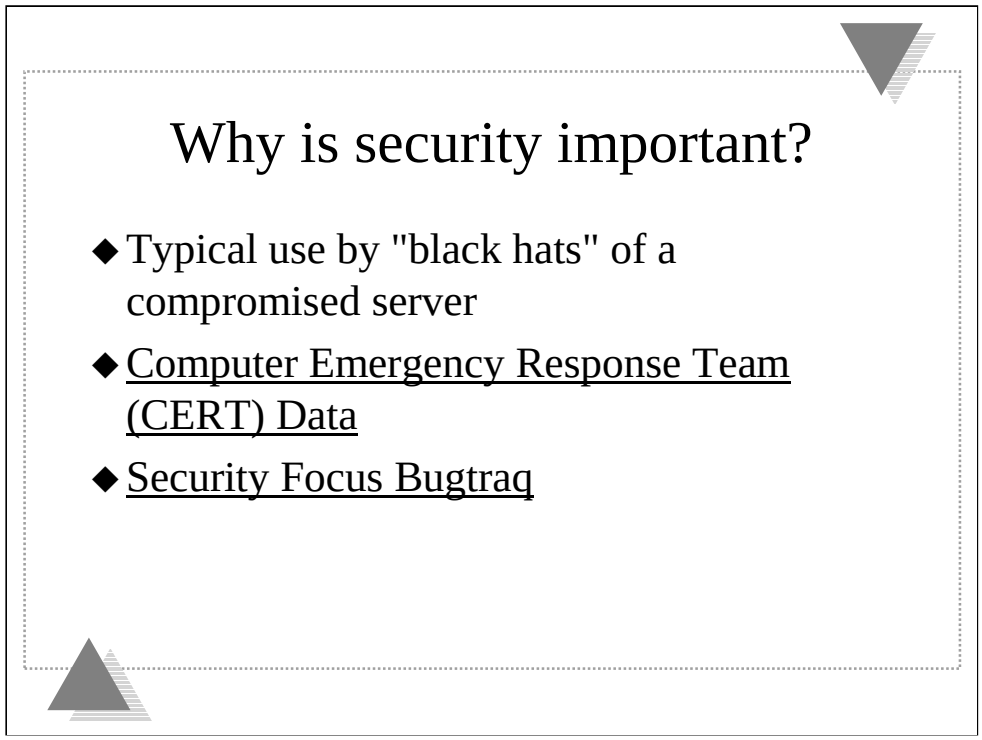
•Web applications are #2 on the list.

•http://www.sans.org/topten.htm

ASF apache.org web site defaced

•Fortunately the exploit was done by "white hats“ who informed the ASF so the security vulnerabilities could be fixed. They might have been "black hats" who could have modified ASF distributions adding back doors, data harvesting, and other malicious code to turn ASF software distributions into trojans.

•http://www.dataloss.nl/papers/how.defaced.apache.org.txt



Notes

Typical use by "black hats" of a compromised server

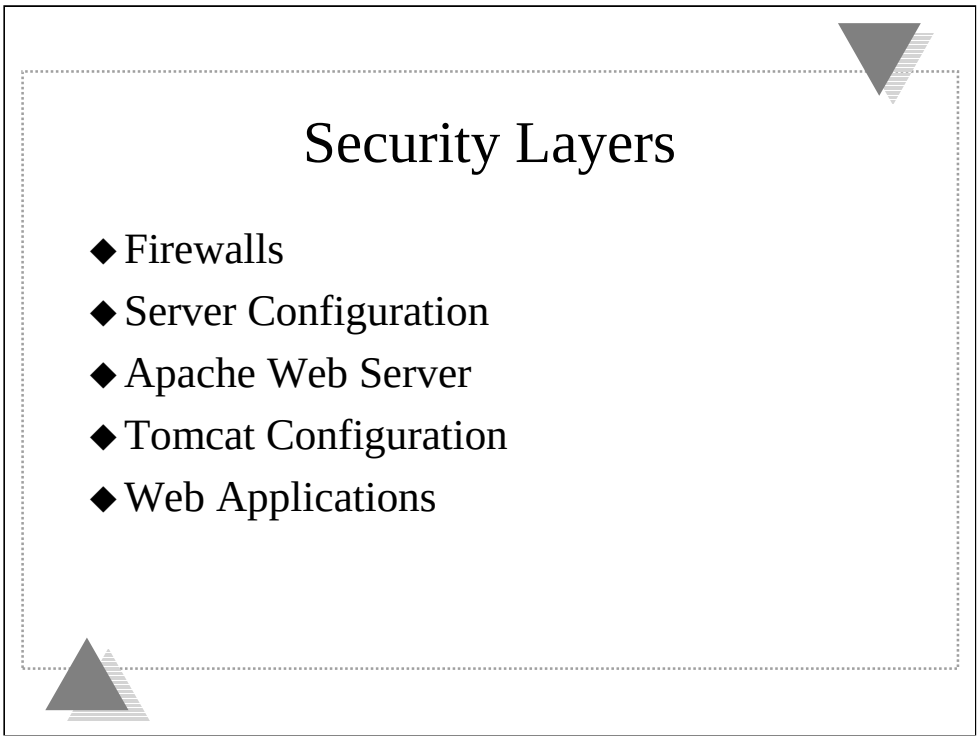
- •Install back doors. (Root kit)
- •Harvest data (Credit Card Numbers!) from server and send it to a remote host.
- •Deface the web site.
- •Be used to launch attacks on other servers

CERT

- •Mailing lists are a good source of information on known exploits and solutions.
- •They only release information about an exploit when there is a solution.
- •http://www.cert.org/

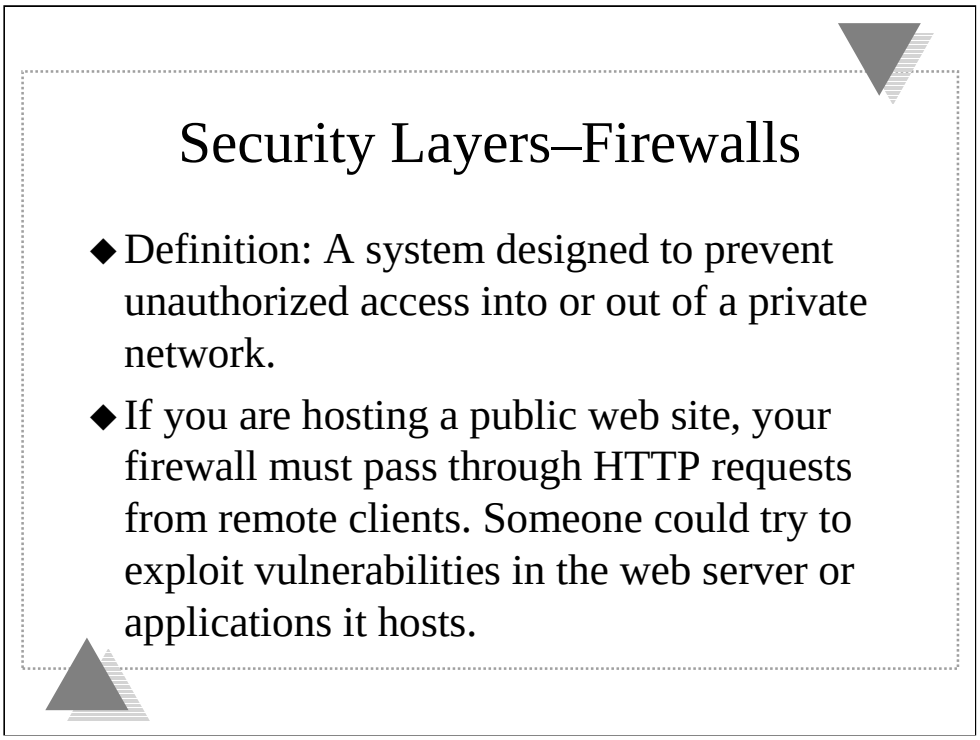
Security Focus Bugtraq

- •Mailing lists are a good source of current information on exploits.
- •http://www.securityfocus.com/



Notes

This isn't a question of which method is best, because security is only as good as the weakest link in the chain.



Notes

But some network traffic is allowed to pass through.

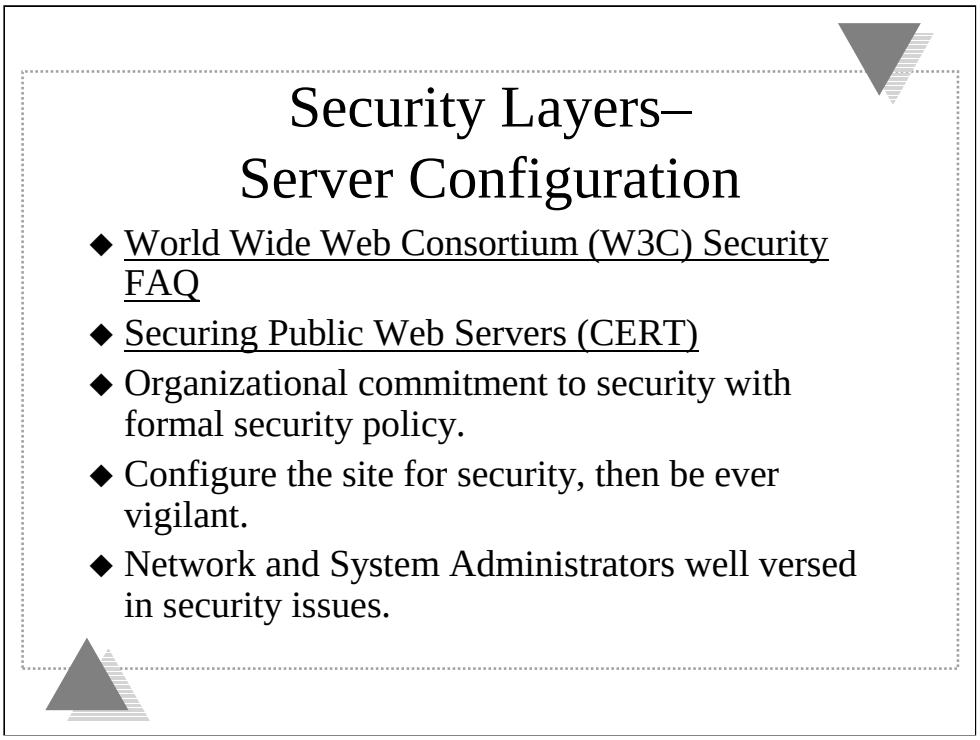
The term "firewall" implies some impenetrable barrier of fire.

I think of it as a mountain range made up of steep cliffs with a few well guarded mountain passes.

Depending on your firewall configuration, it could be just a set of low hills.

A well configured firewall can make it more difficult to compromise a server or deface a web site.

So if you allow HTTP requests behind the firewall, your server is still vulnerable.



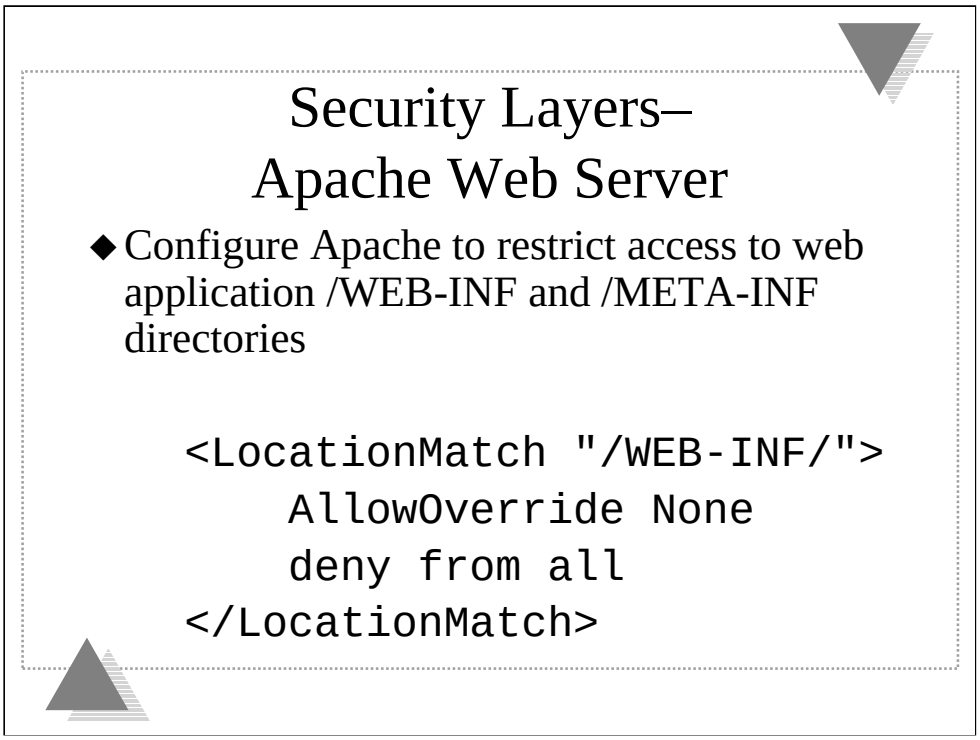
Notes

W3C Consortium

- Introductory Security FAQ—a good place to start.
- http://www.w3.org/Security/faq/www-security-faq.html

CERT

- http://www.cert.org/security-improvement/modules/m11.html



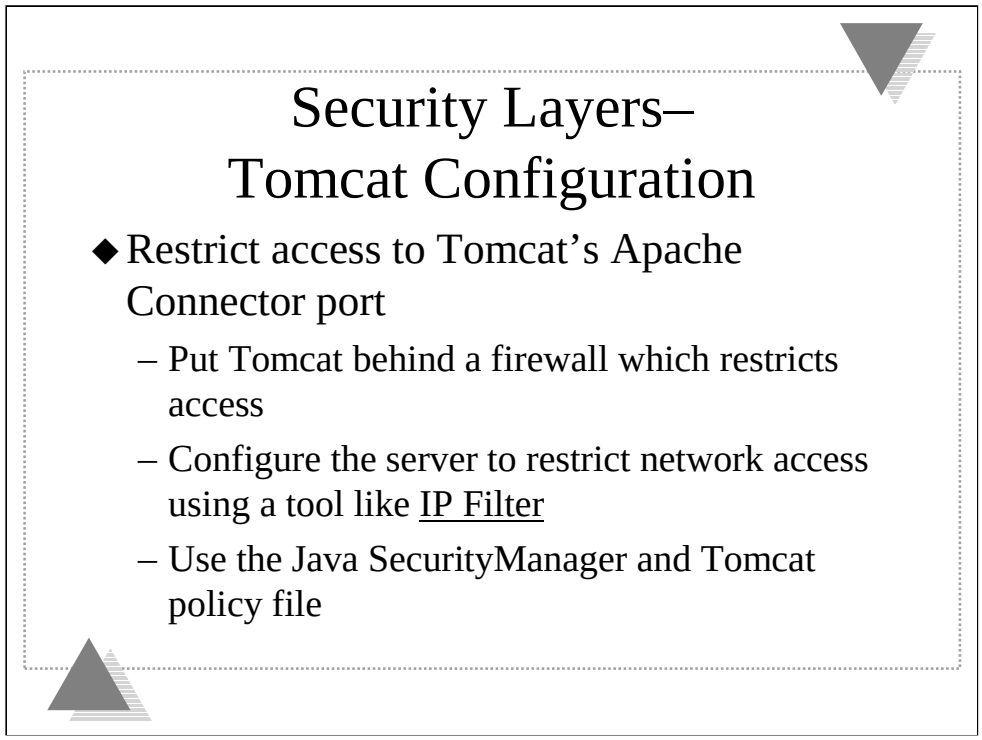
Notes

Configure Apache to restrict access to web application /WEB-INF directory:

```
<LocationMatch "/WEB-INF/">
    AllowOverride None
    deny from all
</LocationMatch>

<LocationMatch "/META-INF/">
    AllowOverride None
    deny from all
</LocationMatch>
```

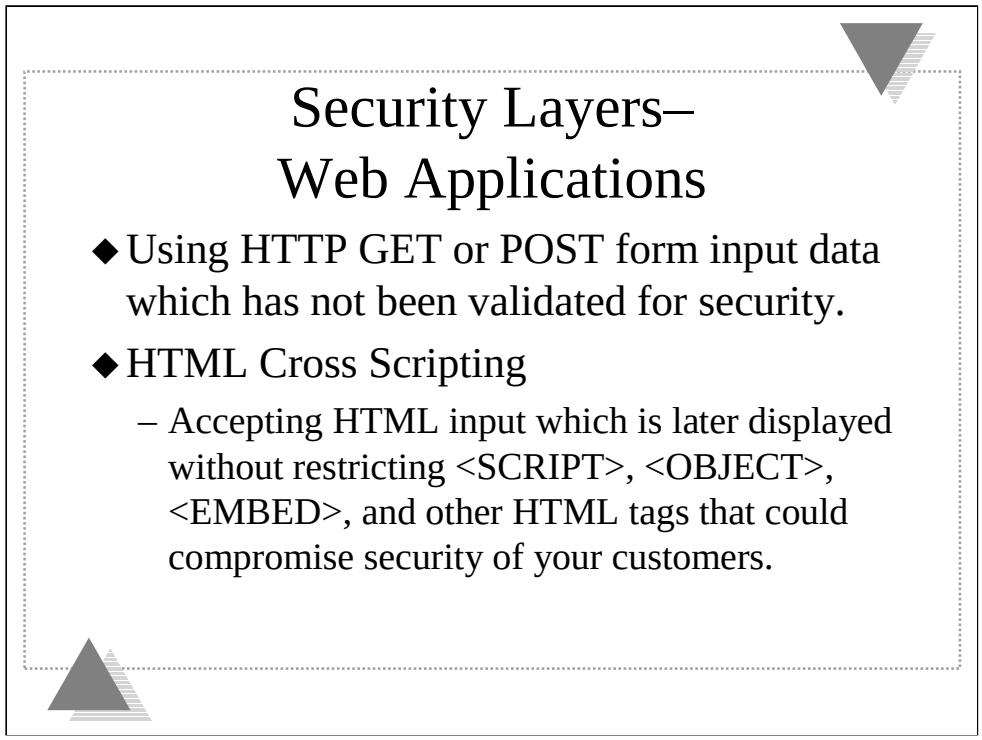
Tomcat internally prevents access to the /WEB-INF directory. If your web applications are aliased into the Apache document space, you will need to restrict access.



Notes

IP Filter

•http://coombs.anu.edu.au/ipfilter/



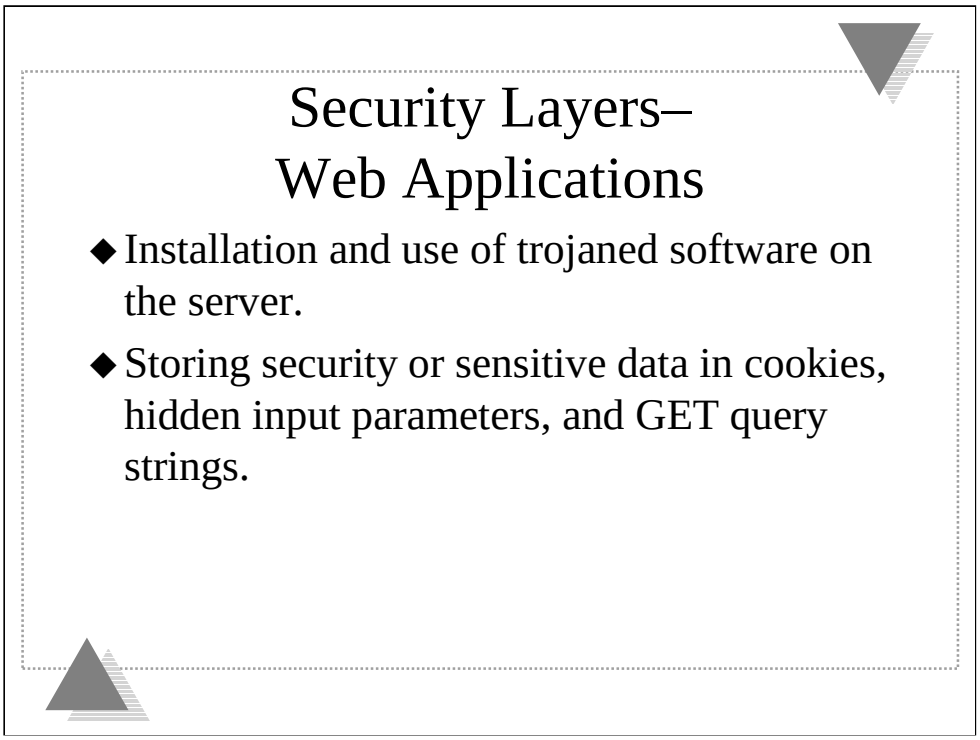
Notes

Using HTTP GET or POST form input data which has not been validated for security.

- Execution of malicious shell commands by CGI scripts which execute shell commands using arguments provided by an HTML form.
- Reading of or writing to system files by applications which use HTML form input parameters to construct directory paths or file names.
- Execution of malicious database SQL commands by applications which construct SQL query strings using input from an HTML form.

HTML Cross Scripting

- Accepting HTML input which is later displayed without restricting <SCRIPT>, <OBJECT>, <EMBED>, and other HTML tags that could compromise security of your customers.
- http://www.cert.org/advisories/CA-2000-02.html



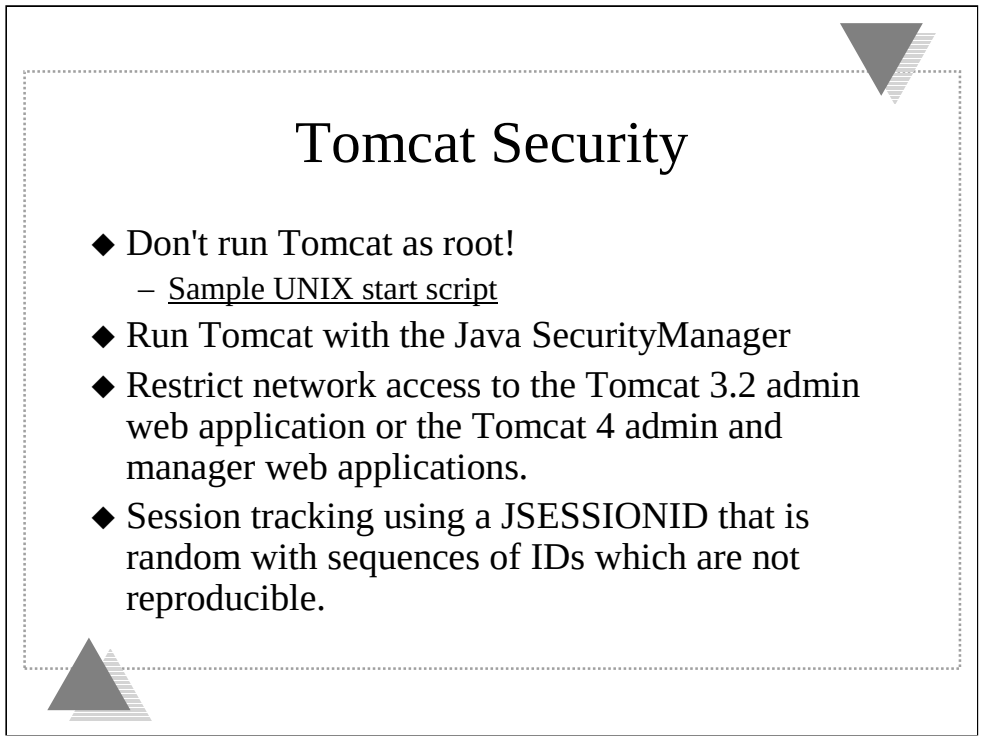
Notes

Installation and use of trojaned software on the server.

- Do all the applications you use come with source?
- If you have the source, do you have time to evaluate each and every one for security?
- Running Tomcat with the Java SecurityManager can help protect you from trojan Java API's, servlets, Java ServerPages (JSP), and JSP tag libraries.

Storing security or sensitive data in cookies, hidden input parameters, and GET query strings.

- Leave any security or sensitive data on the server. Use session tracking to tie the remote client to the session data on the server.



Notes

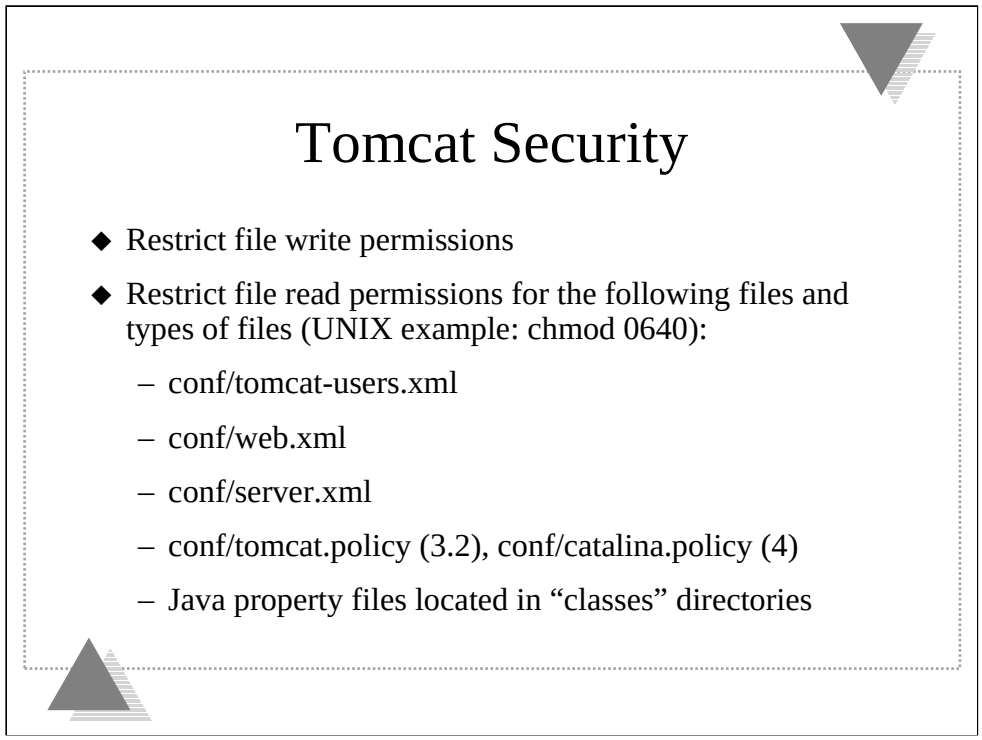
Don't run Tomcat as root!

•Create an unprivileged user with a name like “tomcat”. Disable the user by “starring” the password and assigning a non-existent shell.

•See the sample UNIX start script in Appendix A.

Session tracking using a JSESSIONID that is random with sequences of IDs which are not reproducible.

•This prevents a remote user from hijacking a client's session.



Notes

Restrict file write permissions

•Only give the Tomcat user file write permissions to those directories and files which are required by Tomcat to operate. Don't give Tomcat write access to configuration files.

Restrict file read permissions for the following files and types of files (use chmod 0640 to set UNIX permissions):

conf/tomcat-users.xml

•Contains user id's and passwords for security roles.

conf/web.xml

•Could contain sensitive global configuration parameters

conf/server.xml

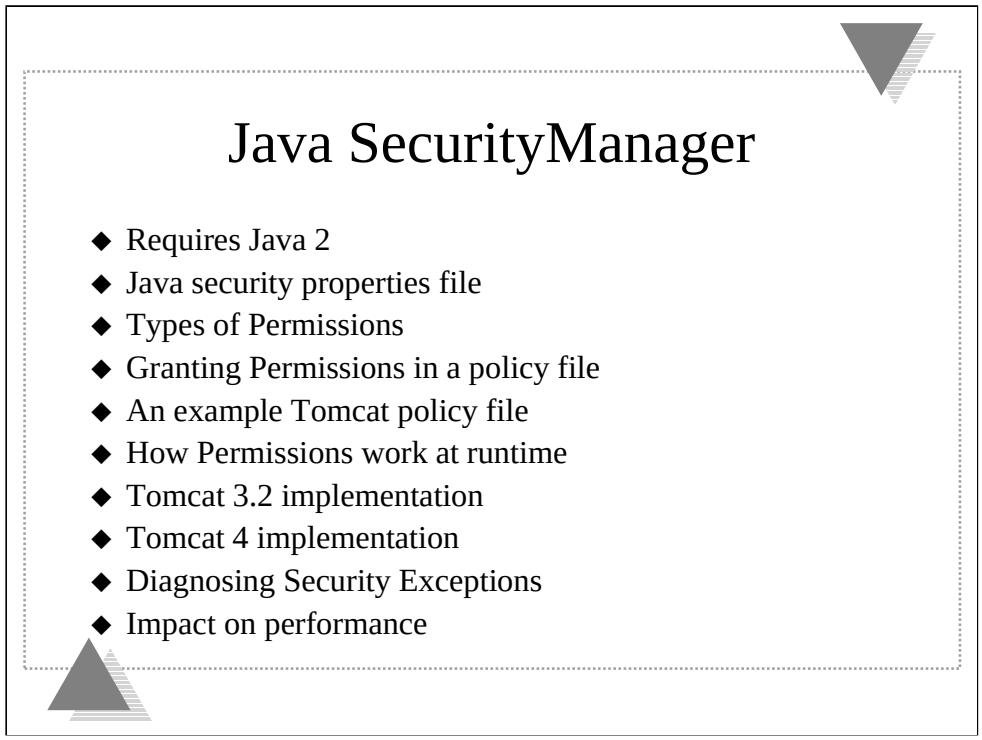
•Could contain sensitive configuration parameters. With Tomcat 4.0 you can configure JNDI named JDBC DataSources. This could include a database userid and password.

conf/tomcat.policy (3.2), conf/catalina.policy (4)

•Contains information about your Java SecurityManager security policies.

Java property files located in “classes” directories

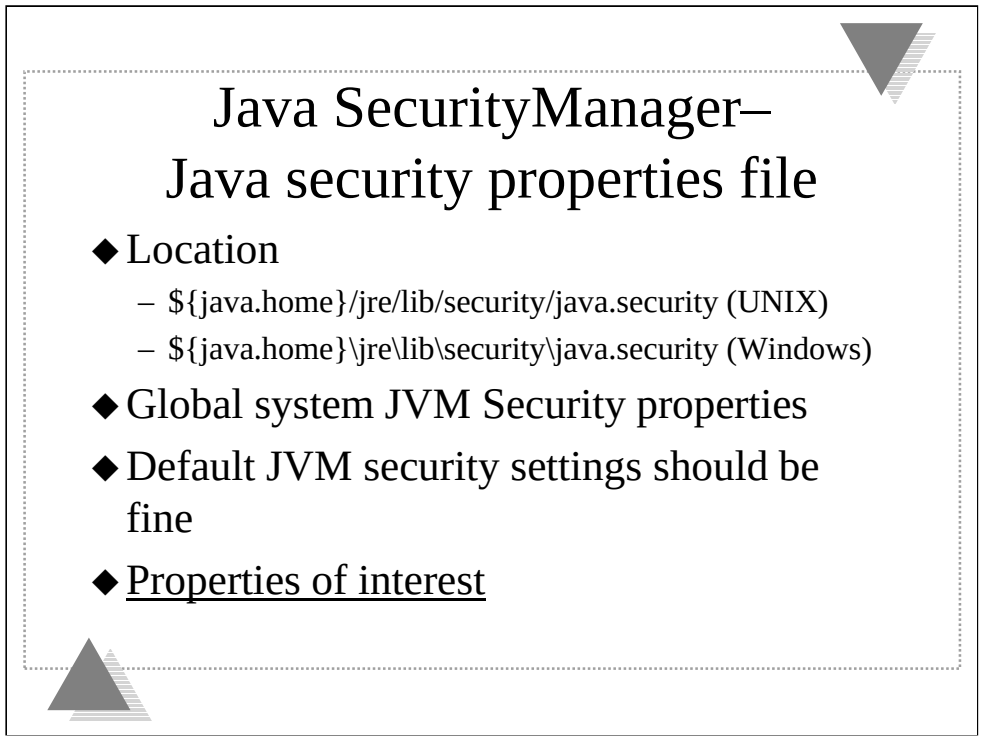
•Could contain sensitive configuration information



Notes

Requires Java 2

•The required security features only became available in Java 2.



Notes

More information on the java.security file from Sun

•http://java.sun.com/docs/books/tutorial/security1.2/summary/files.html

Properties of interest

See the example Java security properties file in Appendix B.

•policy.expandProperties=true

•Allows use of properties such as ${java.home}, ${file.separator} (${/}), ${tomcat.home}, and ${catalina.home} in your tomcat policy file.

•Makes configuring your policy file easier.

•policy.allowSystemProperty=true

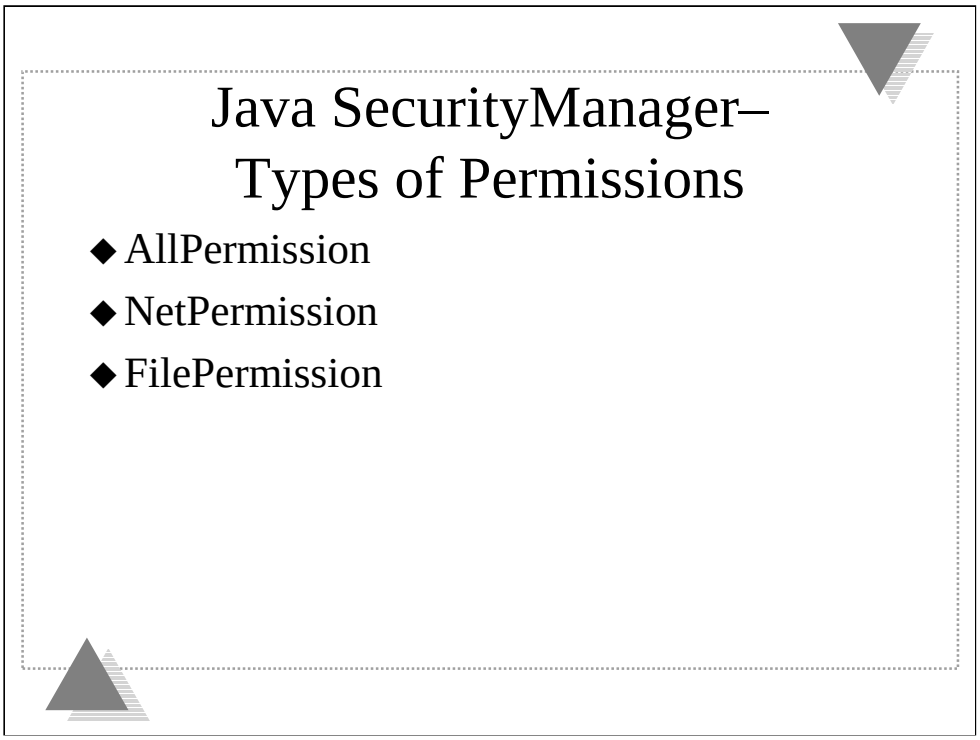
•Allows setting of an alternate security policy file at JVM start using the java "-Djava.security.policy==tomcat_policy_file" system property. Required by Tomcat.

•package.access=sun.

•The Tomcat ClassLoaders implement the RuntimePermission ("accessClassInPackage."+package). This property determines which Java packages the SecurityManager will restrict access to by other Java classes loaded by the JVM.

•package.definition=

•The Tomcat ClassLoaders implement the RuntimePermission ("defineClassInPackage."+package). This property determines which Java packages a Java class can load and instantiate in the JVM. This protects you from a trojan replacing important security related classes such as java.security with their own trojaned version.



Notes

These permissions are for Java 2 version 1.3 and are listed using their Java class name.

- http://java.sun.com/j2se/1.3/docs/guide/security/permissions.html (Java 2 version 1.3)
- http://java.sun.com/products/jdk/1.2/docs/guide/security/permissions.html (Java 2 version 1.2)

AllPermission

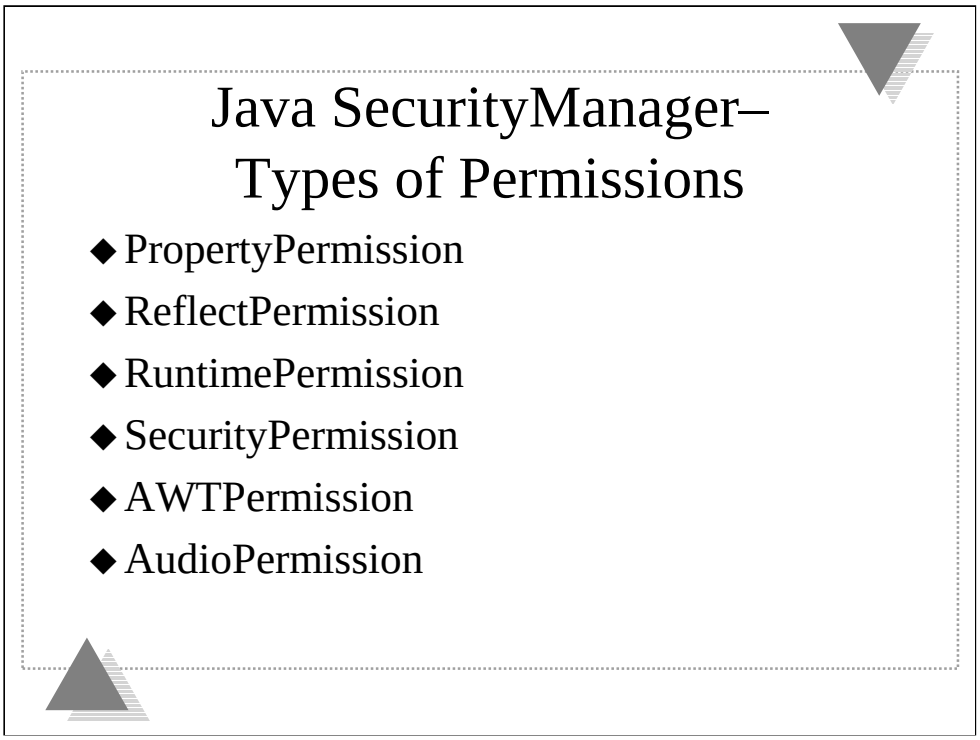
- Classes loaded from a CodeBase granted AllPermission has no security restrictions.
- Should be granted to JVM libraries and JVM extensions.
- Consider impact on Tomcat security when granting AllPermission to a CodeBase other than the JVM and its extensions.
- Never grant AllPermission to a web application.

NetPermission

- Restricts setting of HTTP Authenticators and URL stream handlers.
- Required by Tomcat 4.0
- Should not be given to web applications.

FilePermission

- Used to restrict read, write, delete, and execute privileges to directories and files.
- Permission for Tomcat to read its files and write or delete files in some directories.
- Limit FilePermission's granted to a web application to its document root directory.



Notes

PropertyPermission

- Ability to read or write system properties such as ${os.name} or ${tomcat.home}.
- Tomcat needs permission to read and write all properties.
- Web applications are granted permission to read a safe subset of the system properties.

ReflectPermission

- Ability to introspect protected and private classes and methods.
- Tomcat needs permission to use reflection.
- Web applications should not be given permission to use reflection.

RuntimePermission

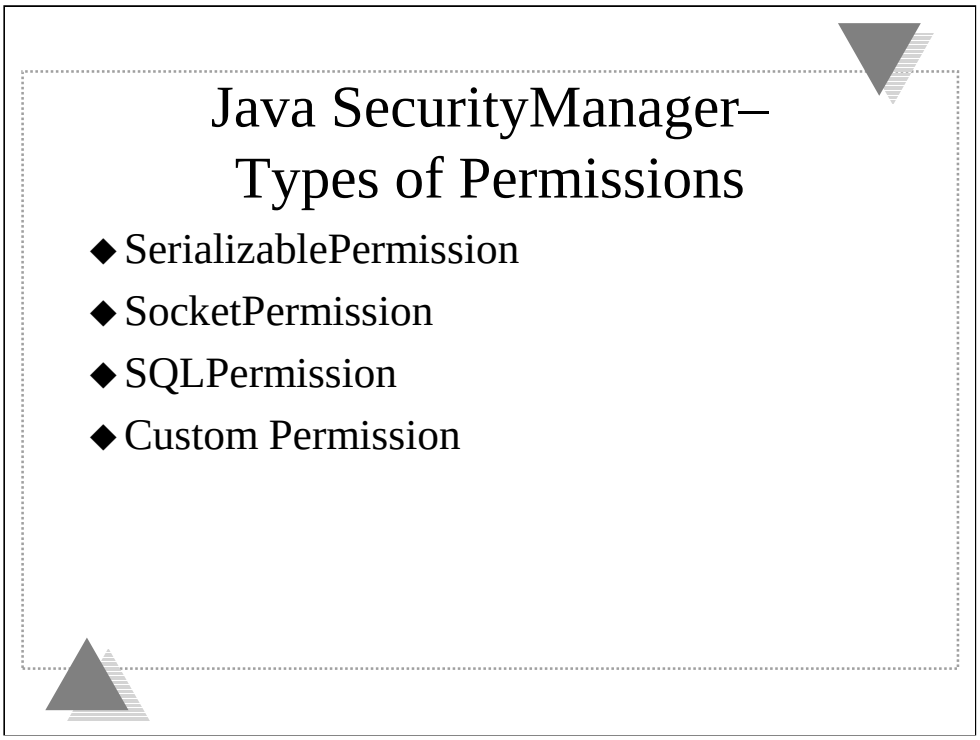
- Permission to do a variety of things at runtime such as exit the JVM, execute a system command, create a ClassLoader, etc.
- Tomcat can be given all runtime permissions.
- Web applications should rarely be given any runtime permissions.

SecurityPermission

- Security related permissions to do things like set or get the Java policy file, read or write security related properties.
- Tomcat can be given all security permissions.
- Web applications should not be allowed any security permissions.

AWTPermission and AudioPermission

- Not applicable to Tomcat



Notes

SerializablePermission

- Setting of default classes used for object serialization.
- Tomcat can be given all serializable permissions.
- Web applications should not be allowed any serializable permissions.

SocketPermission

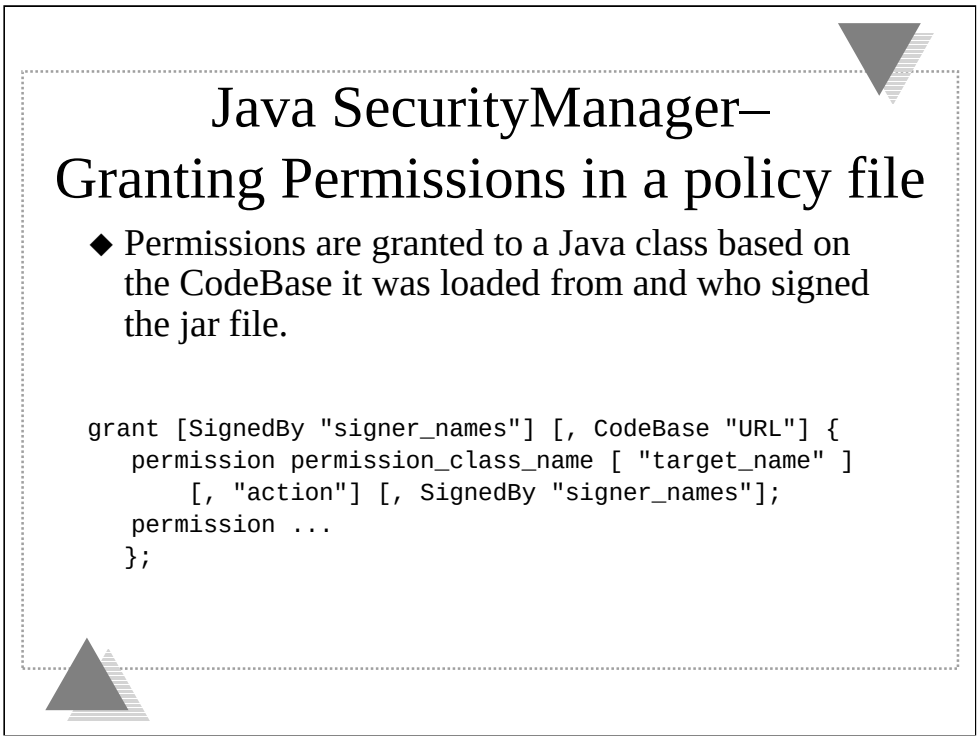
- Permission to accept, connect, listen, or resolve network socket connections.
- Tomcat could be granted all Socket permissions.
- SocketPermissions could be used to restrict access to the Tomcat-Apache Connector port.
- Be careful granting SocketPermissions to web applications. A malicious web application could use a network connection to a remote host to harvest sensitive data.

SQLPermission

- Used to allow setting of the JDBC Driver or DataSource LogWriter.
- Tomcat should be granted permission to set the log writer.
- Web applications should not be allowed permission to set the LogWriter.

Custom Permission

- The java.security API provides Java interfaces and classes which can be used to write your own custom Permission class.



Notes

Syntax for granting security permissions to a CodeBase

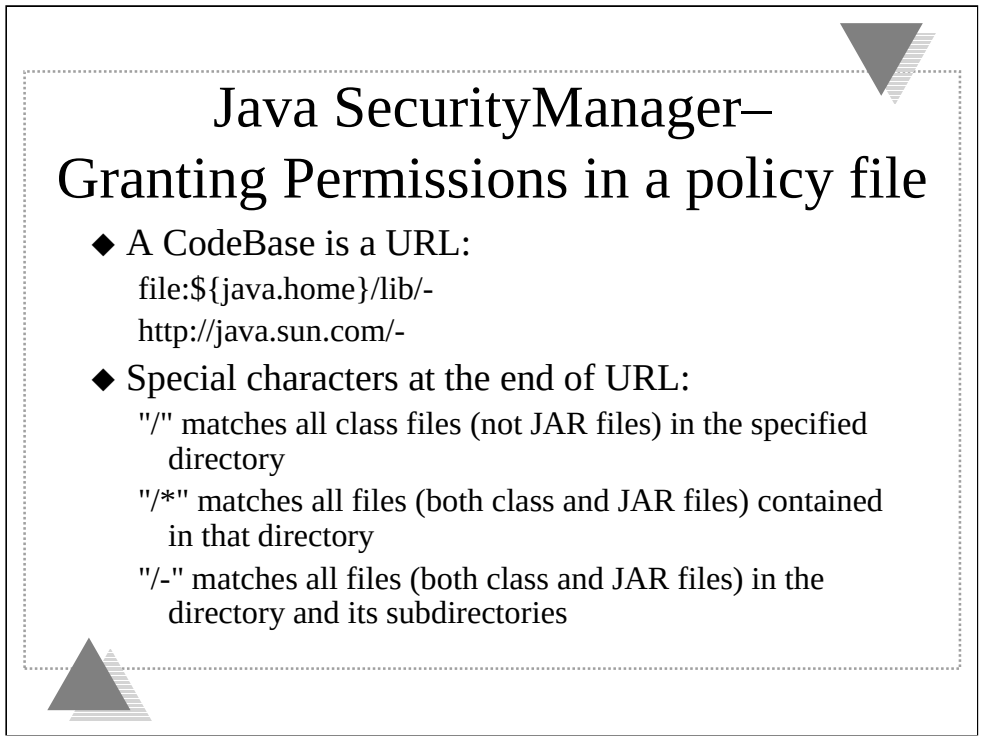
•http://java.sun.com/j2se/1.3/docs/guide/security/spec/security-spec.doc3.html

SignedBy

The SignedBy sets the names of aliases into your keystore for public certificates of who can sign code originating from the CodeBase. SignedBy is optional, if not configured the code does not have to be signed with a certificate.

Permission_class_name

The permission_class_name is the package and class name for one of the permissions we discussed previously. Each target_name and action are specific to a Permission class.



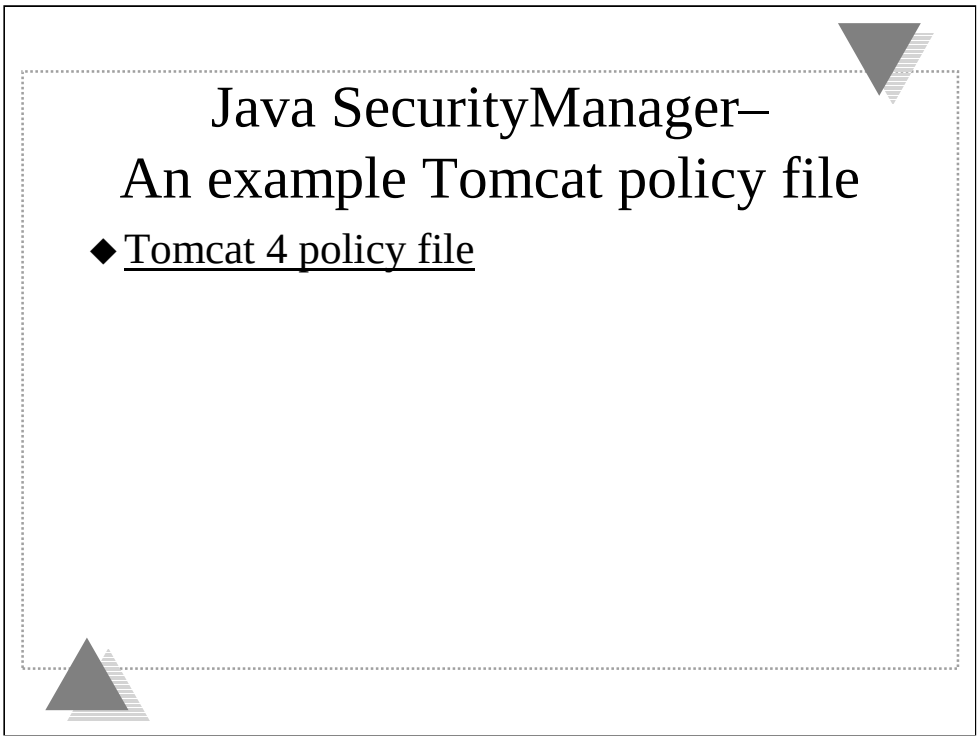
Notes

CodeBase

The CodeBase is a URL, typically a file URL like "file:${java.home}/lib/-" but could be a URL like "http://java.sun.com/-". The CodeBase is optional, if there is no CodeBase these permissions apply to all Java class files no matter where they originated from.

The exact meaning of a CodeBase URL value depends on the characters at the end. A CodeBase with a trailing "/" matches all class files (not JAR files) in the specified directory. A CodeBase with a trailing "/*" matches all files (both class and JAR files) contained in that directory. A CodeBase with a trailing "/-" matches all files (both class and JAR files) in the directory and its subdirectories.

When using a file URL on UNIX, none of the path components can be a symbolic link.



Notes

See the example catalina.policy file in Appendix C. This example is based on Tomcat 4 and the directories where it installs jar's. Tomcat 3.2 uses a different layout for installation of jar files; see the Tomcat 3.2 user documentation.

•The java compiler is located in ${java.home}/lib/tools.jar. It needs AllPermission so that Tomcat's JSP engine (Jasper) can compile JavaServer Pages.

•The JVM and its extensions are located in ${java.home}/jre/lib/. The JVM and extensions can be granted AllPermissions.

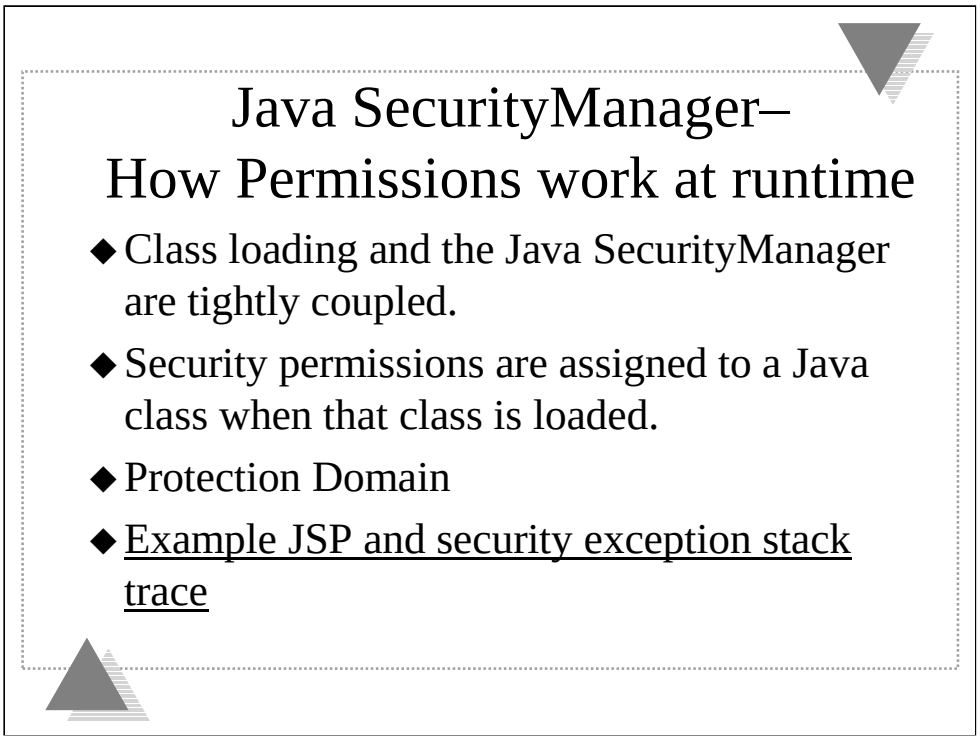
•You could grant Tomcat AllPermissions, but to be more secure, you can limit Tomcat to just the permissions it needs.

•You can use the ${file.separator} or ${/} to make a FilePermission portable between UNIX and Windows. The trailing "/-" grants the permission to that directory and all sub directories. A trailing "/*" grants the permission to just the contents of that directory.

•A SocketPermission specifies a remote host or IP address, with an optional port number. You can use "*" as a component in a host name or IP address to do pattern matching. Port numbers can specify a range using "-", "1024-" are all ports >= 1024, "8000-8010" would be all ports 8000 <= port <= 8010.

•The grant without a CodeBase contains the default permissions used for web applications.

•You can give a web application additional permissions by creating a grant for its web application directory.



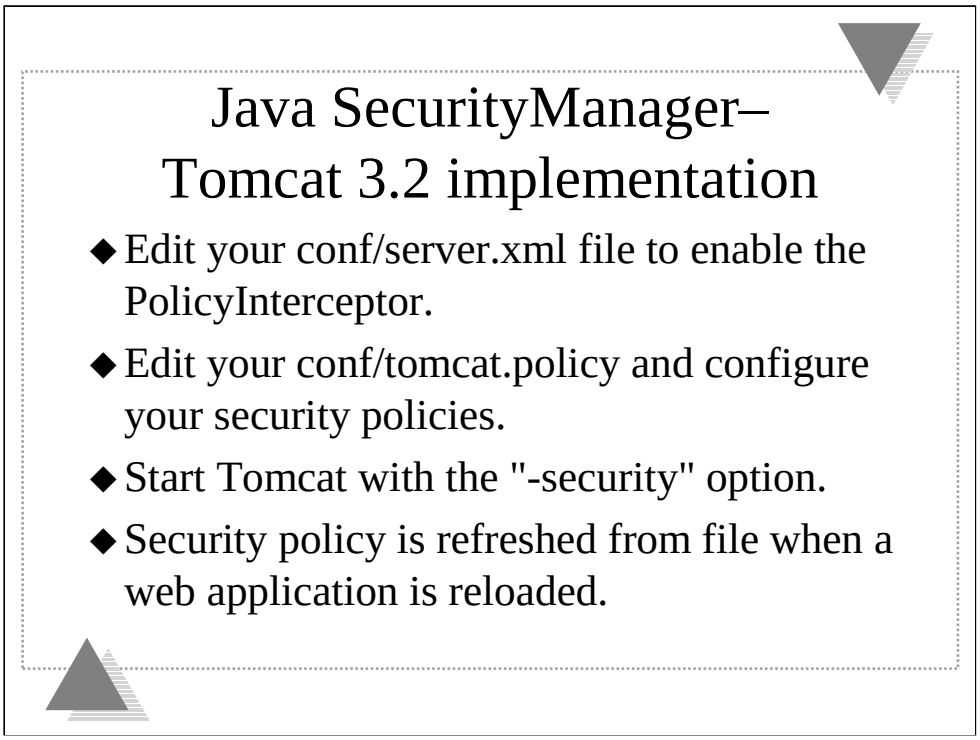
Notes

Protection domain

•Permissions are enabled for a Codebase at runtime based on the intersection of the permissions granted to all Codebase's on the Java runtime stack.

See Example JSP and security exception stack trace in Appendix D.

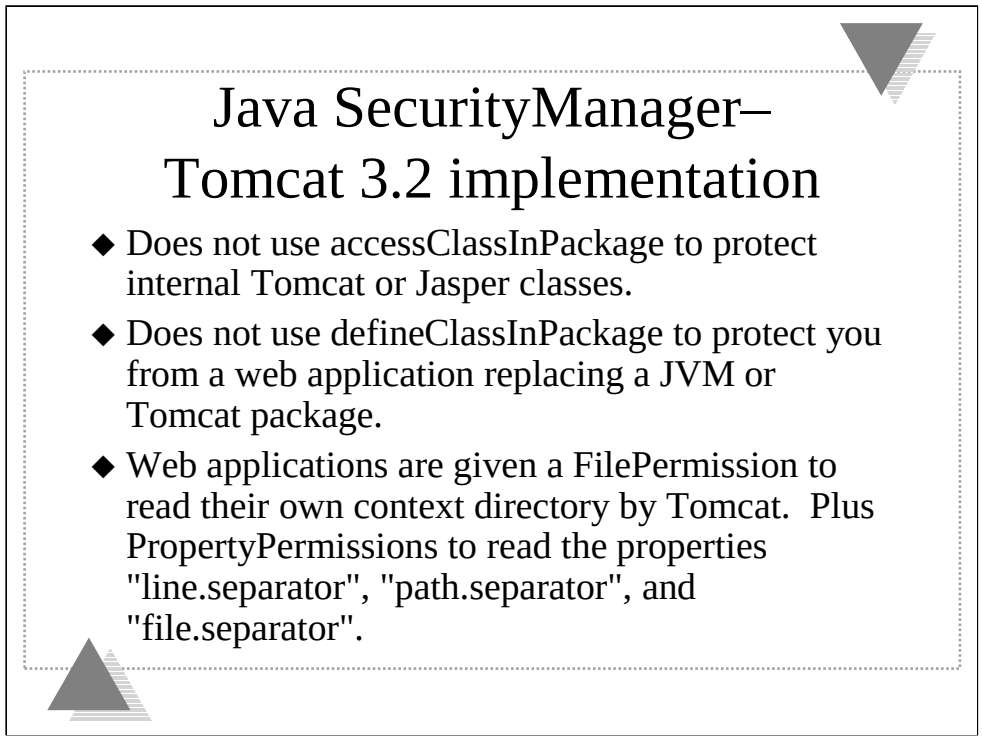
If a servlet or JSP performs an include or forward, methods in the Tomcat core classes have to be used. However, the Servlet or JSP may not have the permissions needed. In this case, the Tomcat core uses the java.security.AccessController.doPrivileged method. This resets the permissions on the Java stack to those of the class being executed.



Notes

Security policy is refreshed from file when a web application is reloaded.

•If you need to change your security policy for a web application, make the necessary changes to your policy file, then reload the web application.



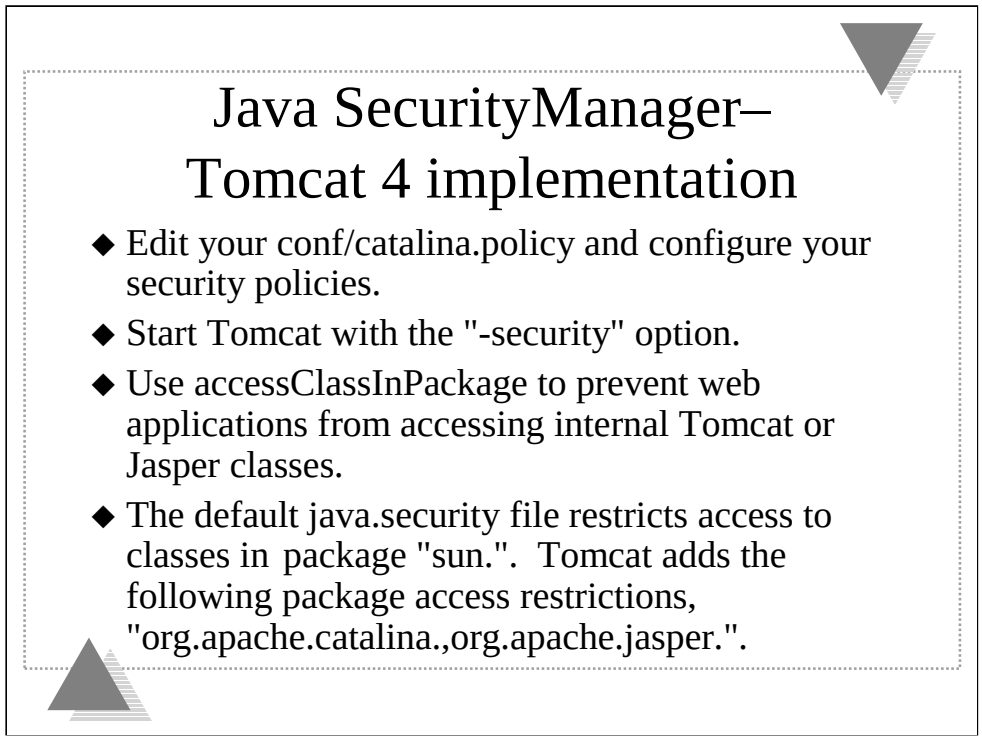
Notes

Does not use accessClassInPackage to protect internal Tomcat or Jasper classes.

•A web application could access any public data or methods in Tomcat or Jasper classes.

Does not use defineClassInPackage to protect you from a web application replacing a JVM or Tomcat package.

•Since Tomcat 3.2 class loading defers to the parent class loader first, this shouldn't be a problem.

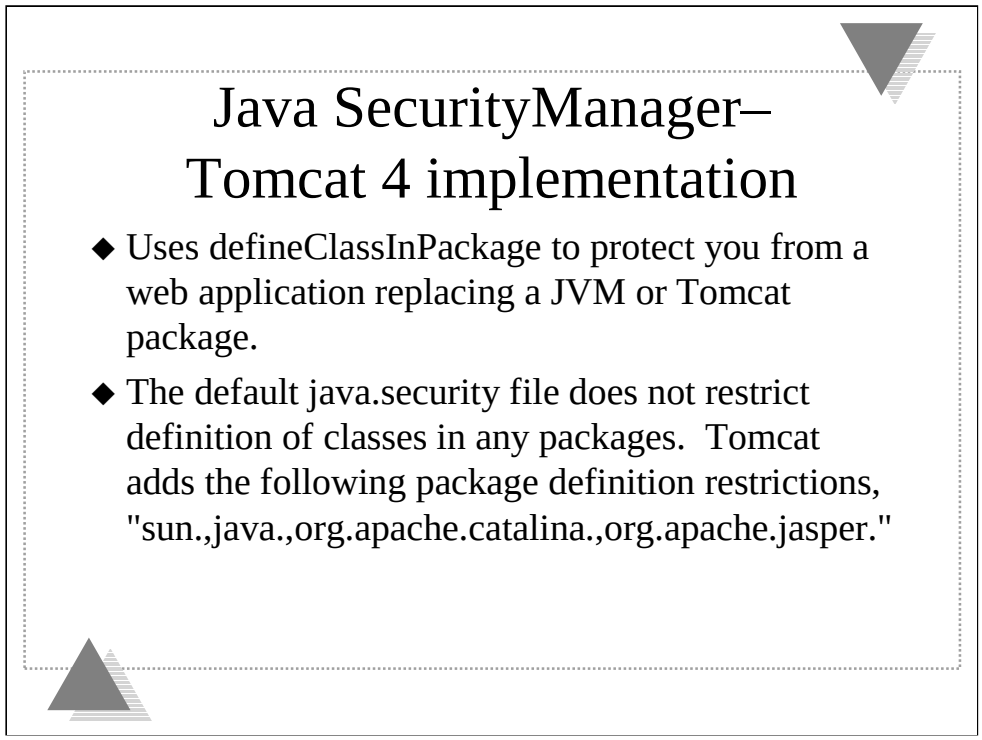


Notes

Note that unlike Tomcat 3.2, PolicyInterceptor does not exist in Tomcat 4.

accessClassInPackage

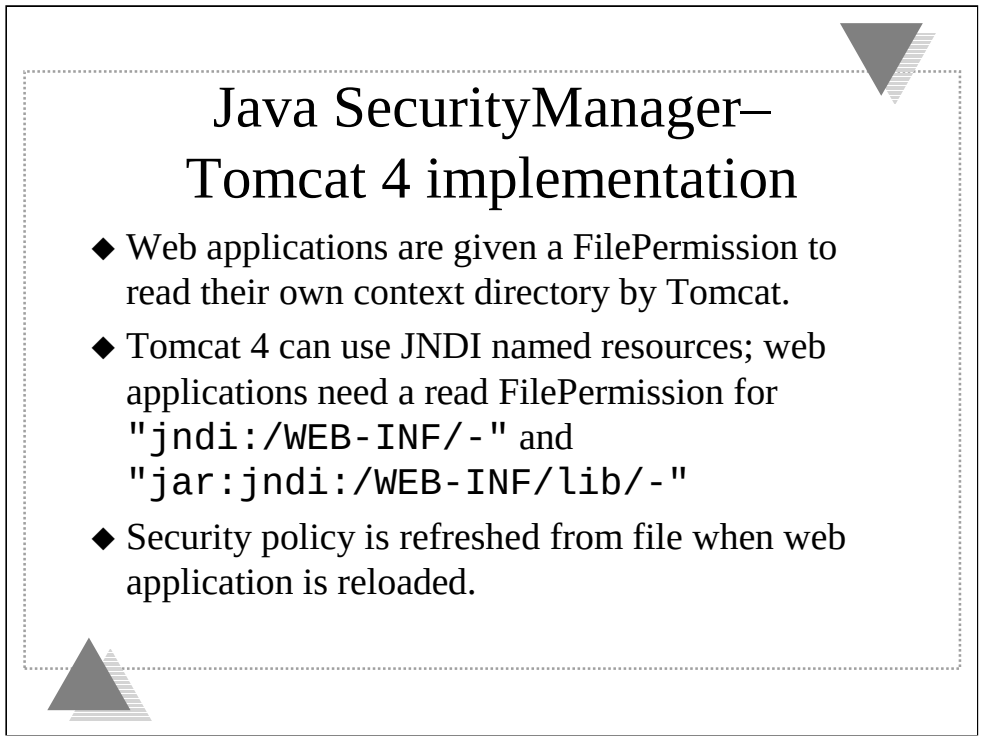
•Because Tomcat 4 implements accessClassInPackage, it is inherently more secure than Tomcat 3.2



Notes

Uses defineClassInPackage to protect you from a web application replacing a JVM or Tomcat package.

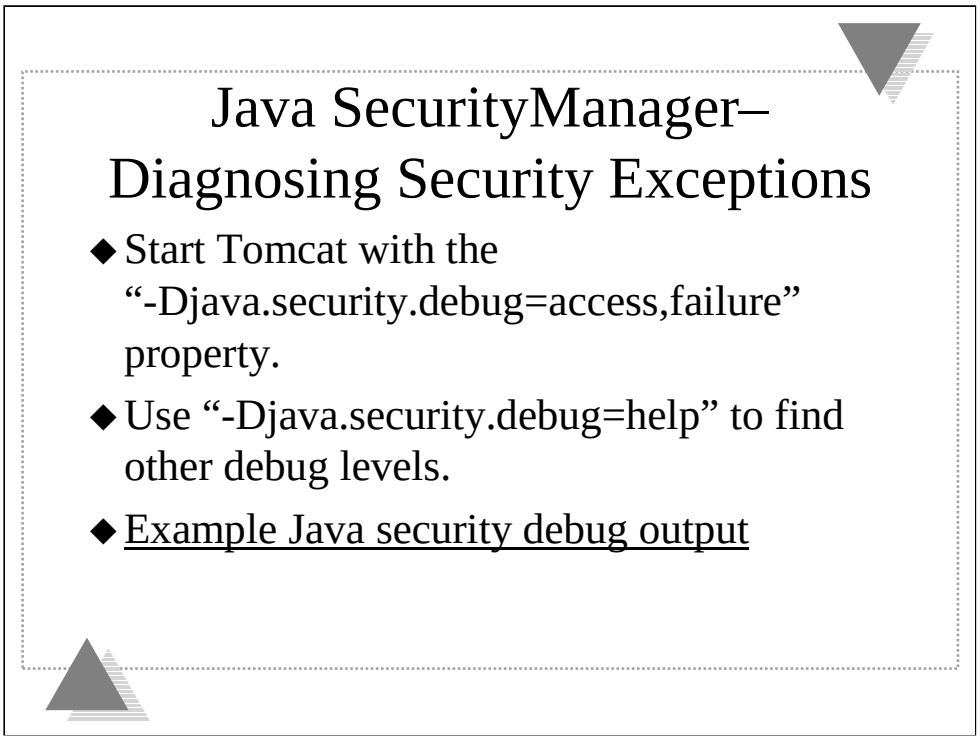
- Since Tomcat 4 web application class loading defers to the web application ClassLoader before trying its parent ClassLoader, defineClassInPackage had to be implemented.



Notes

Security policy is refreshed from file when web application is reloaded.

•If you need to change your security policy for a web application, make the necessary changes to your policy file, then reload the web application.



Notes

See the example Java security debug output in Appendix E.

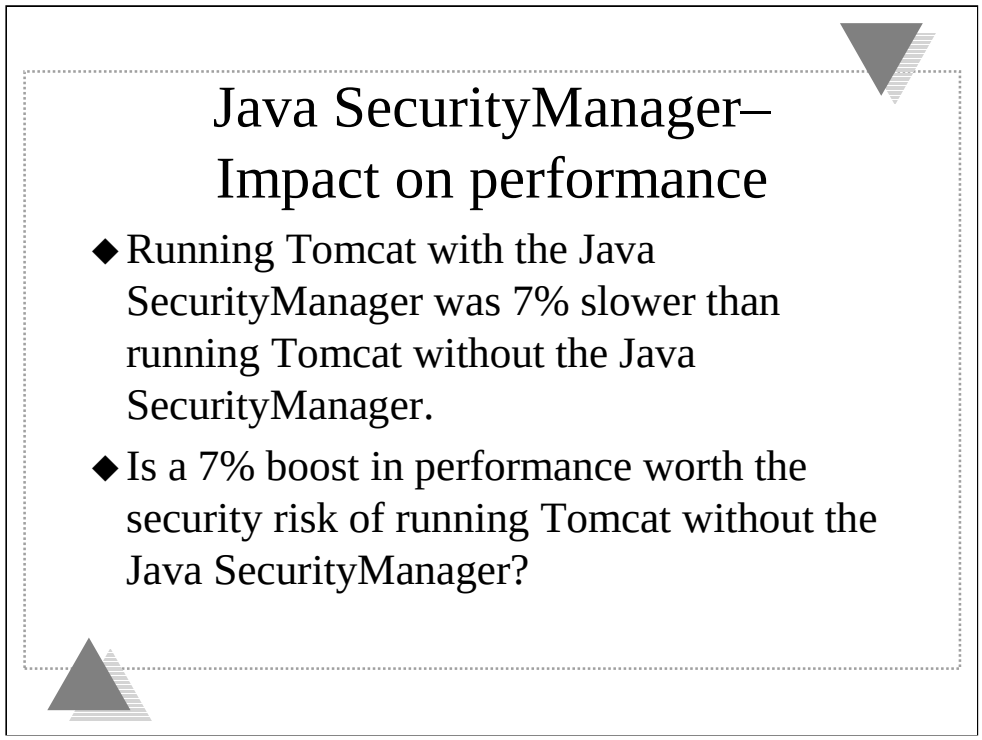
Using security debug can generate many MB's of output to stderr.

Search debug output for the string "denied".

Review the permission that was denied and the accompanying Tomcat execution stack trace.

Under the stack trace the ProtectionDomain which failed to have adequate permission will be listed.

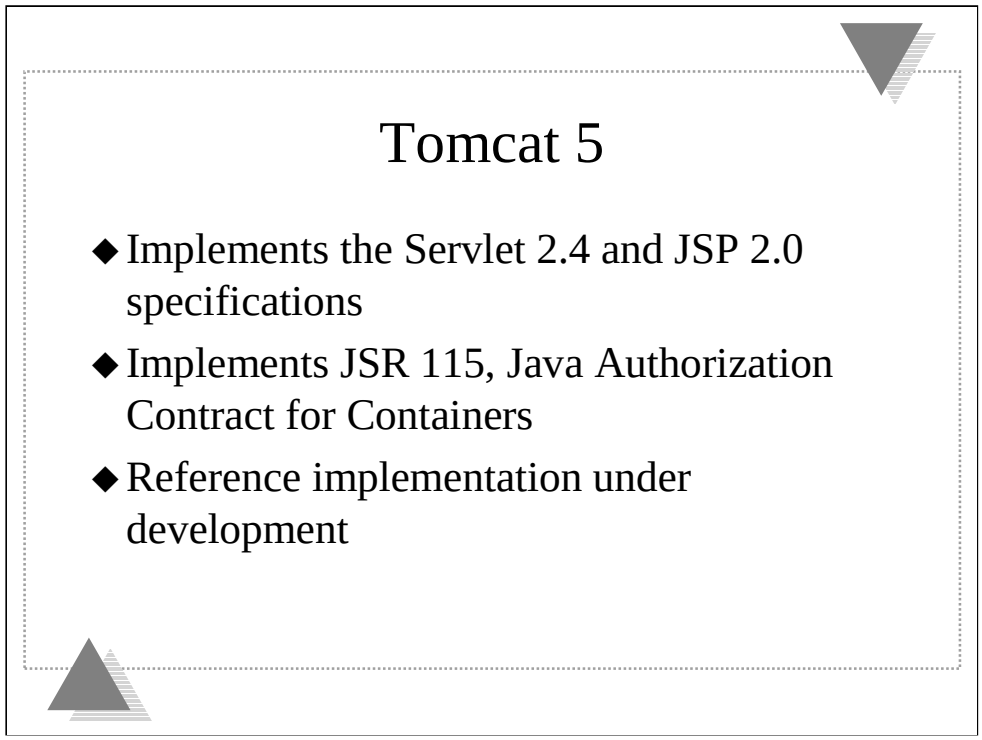
There will be times when the SecurityManager denies access during the normal operation of Tomcat. When looking at the debug output, you need to identify which one is causing your web application to fail.



Notes

Tests done using Java 2 J2SE 1.3 using HotSpot server on Solaris X86

Testing done using the Jakarta Tomcat 4 and the Tomcat 4 watchdog tests. The watchdog consists of hundreds of tests used to verify a servlet container implements the servlet and JSP specifications correctly.



Notes

Servlet 2.4 specification

•http://jcp.org/aboutJava/communityprocess/first/jsr154/

JSP 2.0 specification

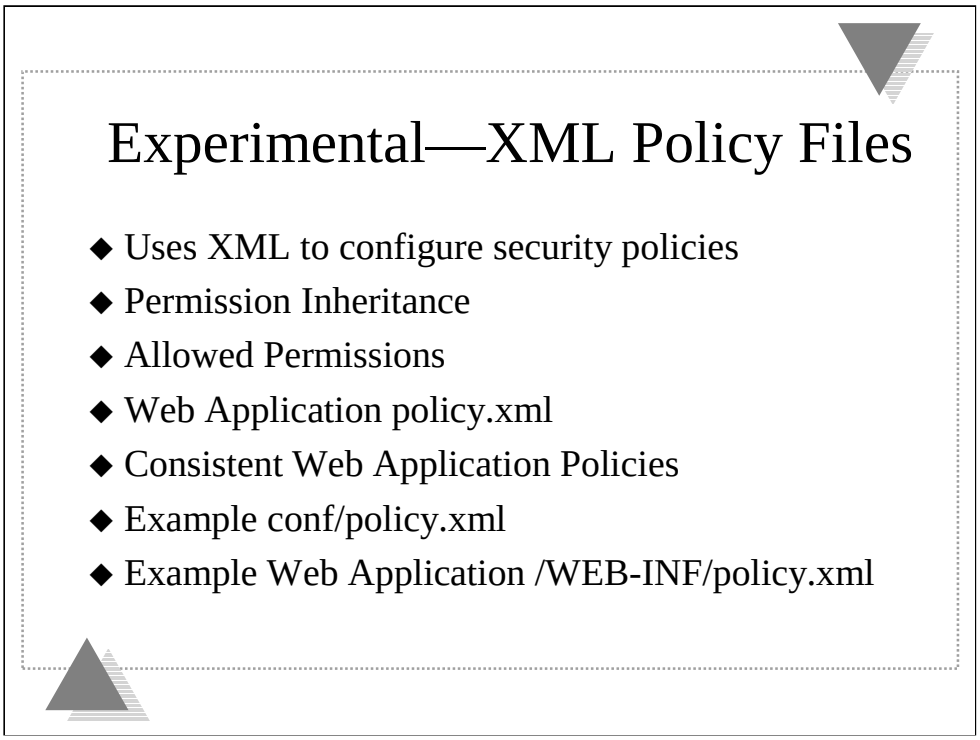
•http://jcp.org/aboutJava/communityprocess/first/jsr152/index.html

Java Specification Request JSR 115, Java Authorization Contract for Containers

•http://www.jcp.org/jsr/detail/115.jsp

•Role based access control using Java SecurityManager built on top of Java Authentication and Authorization Service (JAAS)

http://java.sun.com/products/jaas/



Notes

Permission Inheritance

Policies for different code bases can be nested with the parent inheriting the permissions configured for a child. This makes configuring your security policy easier. See Appendix F for example.

Allowed Permissions

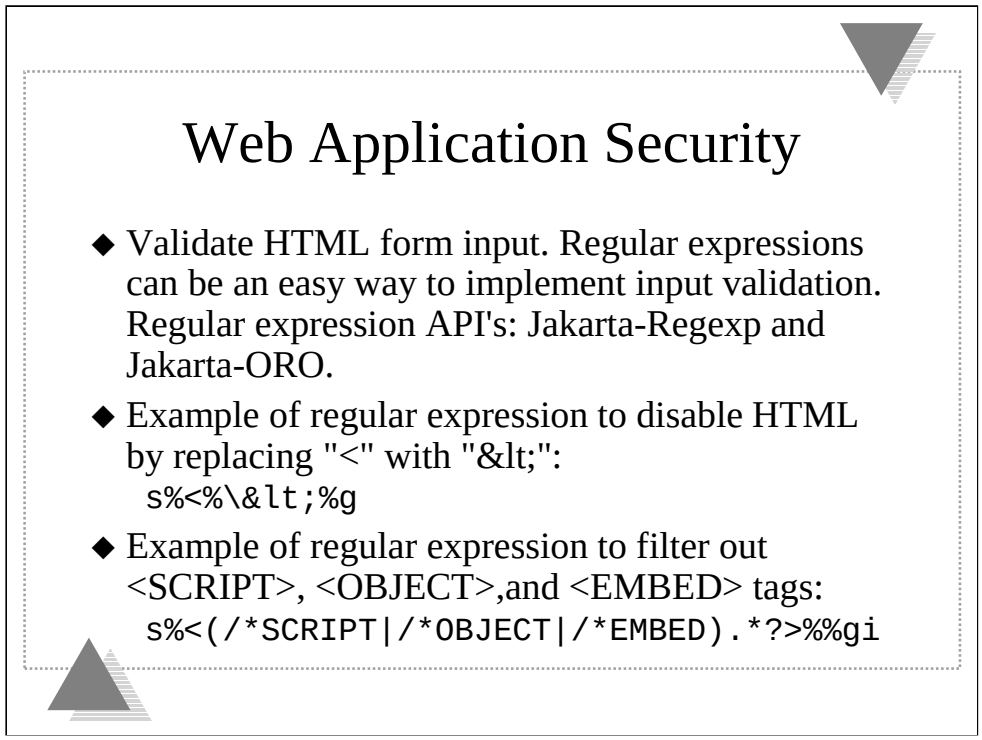
What permissions can be configured for a child code base such as a Host or Web Application can be restricted to those Allowed by the parent.

Web Application policy.xml

Permissions for a web application can be configured in its own /WEB-INF/policy.xml file. This allows the security policy and permissions for a web application to be bundled with it. See Appendix G for example.

Consistent Web Application Policies

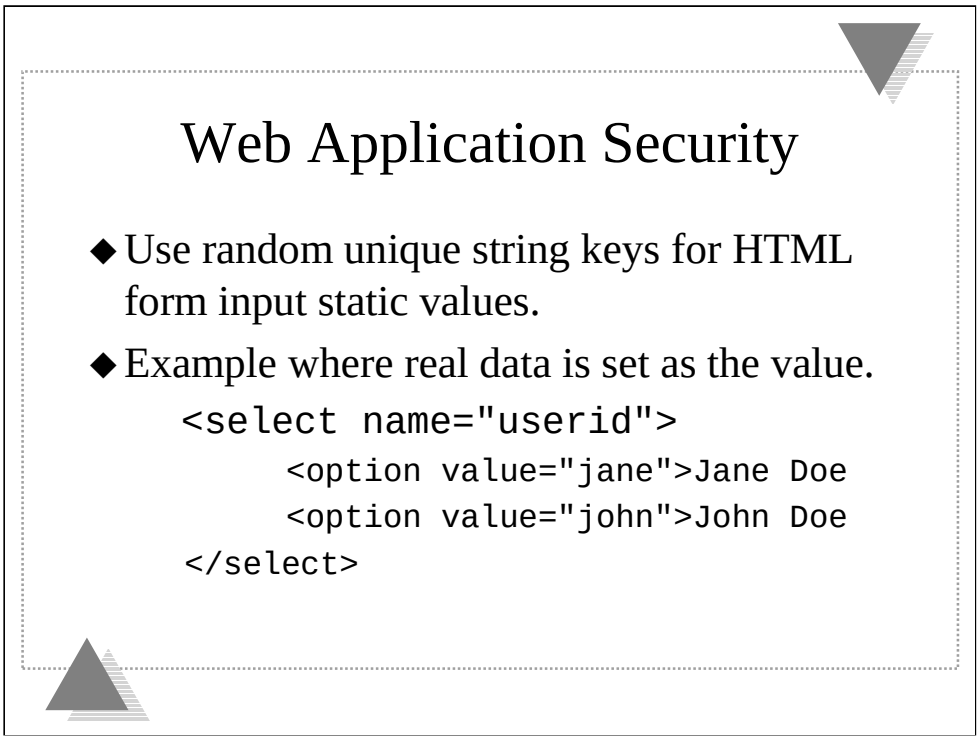
Web application security policy configuration is the same whether the web application is running from a directory or from a WAR.



Notes

Regular Expression API's:

- Jakarta-Regexp
  - http://jakarta.apache.org/regexp/
- Jakarta-ORO
  - http://jakarta.apache.org/oro/



Notes

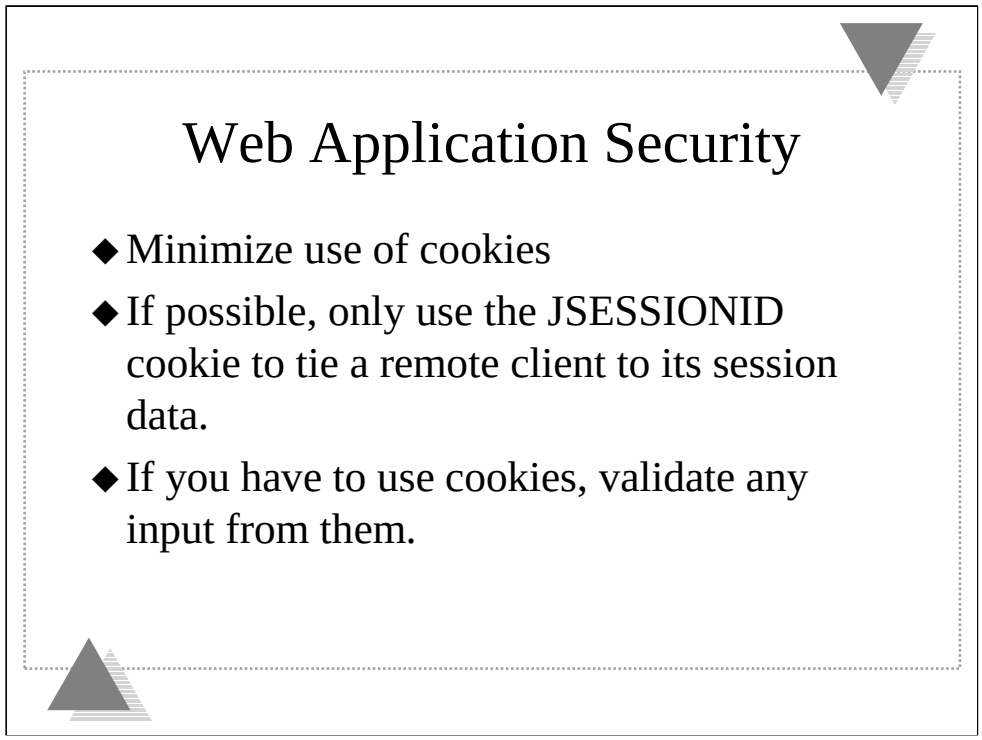
What if the remote client submitted the form with "userid=root"?

Notes

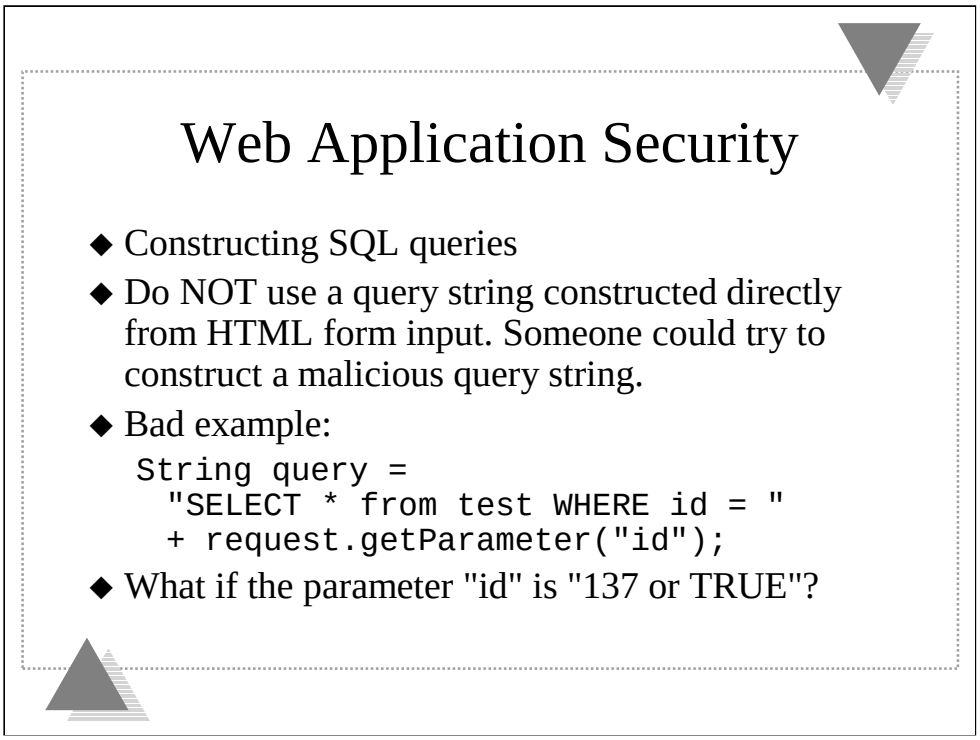
Use random unique string keys for static HTML form input values. This prevents spoofing of form input values.

Map the unique keys to the real value on the server side and store in the users session.

The random keys should be unique for each HTTP request.


Web Application Security
Minimize use of cookies
If possible, only use the JSESSIONID cookie to tie a remote client to its session data.
If you have to use cookies, validate any input from them.

Notes


Web Application Security
Constructing SQL queries
Do NOT use a query string constructed directly from HTML form input. Someone could try to construct a malicious query string.
Bad example:
String query =
"SELECT * from test WHERE id = "
+ request.getParameter("id");
What if the parameter "id" is "137 or TRUE"?

Notes

Notes

Advantages of PreparedStatement

- Will escape special characters in strings.
- Strongly typed
- Prevents spoofing of the SQL query string.

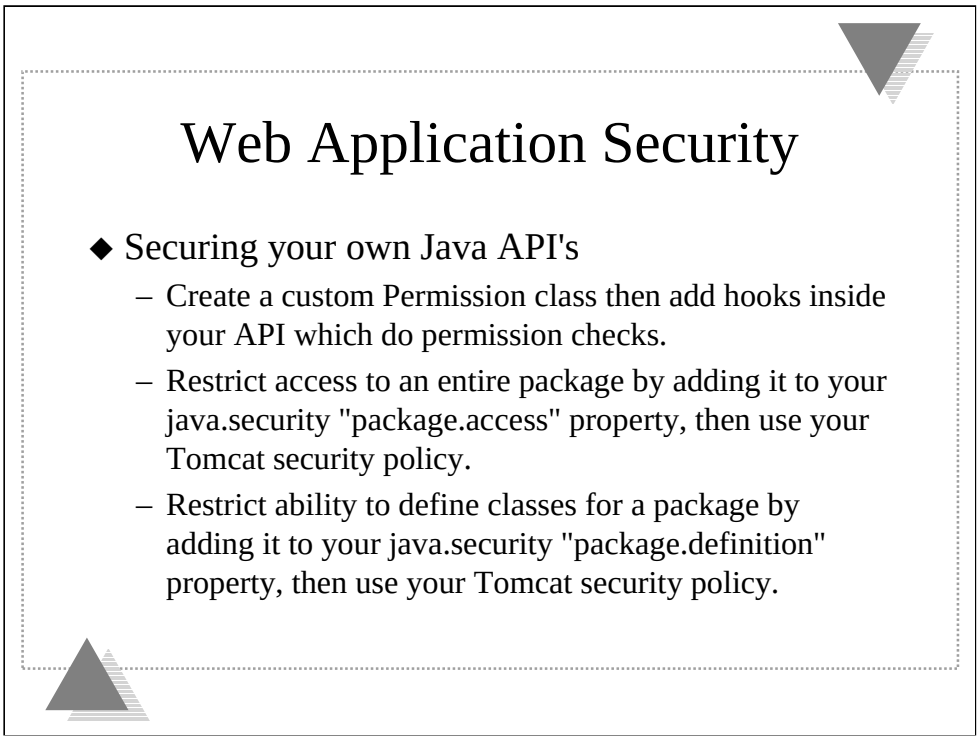
Web Application Security

- Host web applications which handle sensitive data on a secure server which uses SSL (https).
  - Verify that the web application is running on a secure server.

```
if( !request.isSecure() ||
  request.getServerPort() != 443 ) {
   // Redirect user to an error page
}
```

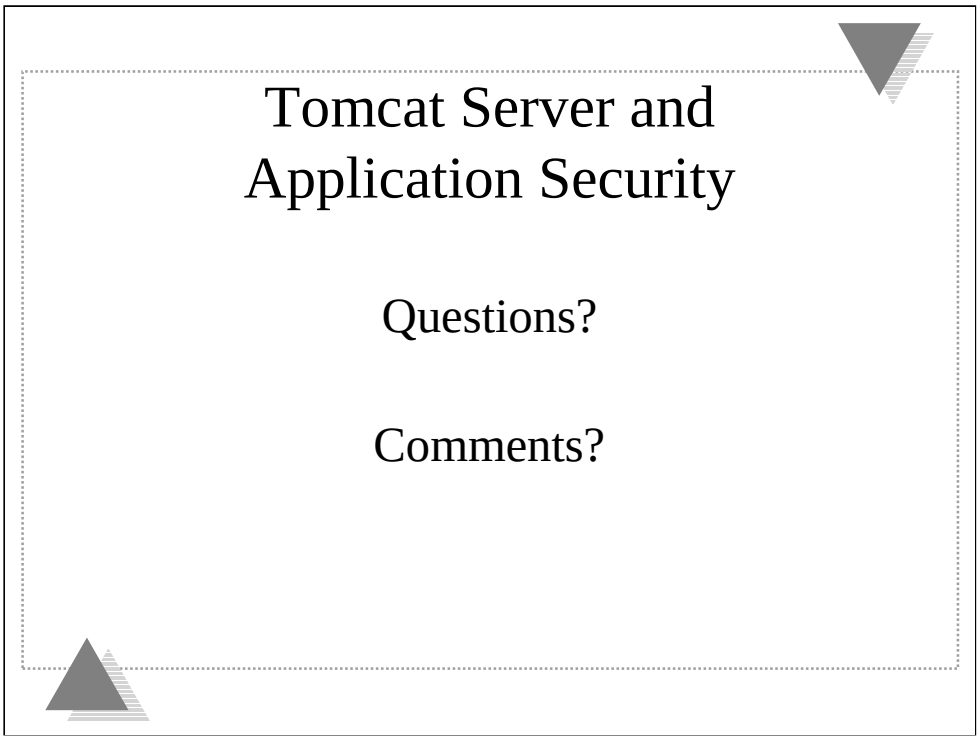
- Use the Java SecurityManager!

Notes



Notes

You may want to make your own Java API's more secure if they perform sensitive operations like account administration.


Tomcat Server and
Application Security
Questions?
Comments?

Notes

Appendix A--Sample UNIX Start Script

```
#!/bin/sh

unset CLASSPATH
CATALINA_HOME=/usr/local/tomcat4
BINDIR=${CATALINA_HOME}/bin

JAVA_HOME=/usr/local/java

PATH=${JAVA_HOME}/bin:${PATH}
CATALINA_OPTS="-server -Xms160m -Xmx160m -Xss128k -Xincgc"
#CATALINA_OPTS="-server -Djava.security.debug=access,failure"
#CATALINA_OPTS="-server -Djava.security.debug=all"

export CATALINA_HOME JAVA_HOME PATH CLASSPATH CATALINA_OPTS

CATALINA_USER=tomcat
CATALINA_OPTIONS=-security

case $1 in
'start')
        echo "Starting Tomcat with ulimit -n 1024"
        ulimit -n 1024
        echo "CD to catalina home so that turbine/torque is happy"
        cd ${CATALINA_HOME}
        echo "Starting Tomcat with the following options: ${CATALINA_OPTIONS}"
        su ${CATALINA_USER} -c "${BINDIR}/startup.sh ${CATALINA_OPTIONS}"
        ;;
'stop')
        su ${CATALINA_USER} -c ${BINDIR}/shutdown.sh
        ;;
*)
        echo "usage: ${CATALINA_HOME}/tomcat {start|stop} {CATALINA_OPTIONS}"
        ;;
esac
```

Appendix B--Example Java Security Properties File

```
# This is the "master security properties file".
#
# In this file, various security properties are
# set for use by # java.security classes. This
# is where users can statically register
# Cryptography Package Providers
# ("providers" for short). The term "provider"
# refers to a package or set of packages that
# supply a concrete implementation of a subset
# of the cryptography aspects of the Java Security
# API. A provider may, for example, implement one or
# more digital signature algorithms or message
# digest algorithms.
#
# Each provider must implement a subclass of the
# Provider class. To register a provider in this
# master security properties file, specify the
# Provider subclass name and priority in the format
#
#    security.provider.=
#
# This declares a provider, and specifies its
# preference order n. The preference order is
# the order in which providers are searched for
# requested algorithms (when no specific provider
# is requested). The order is 1-based; 1 is the
# most preferred, followed by 2, and so on.
#
# <className> must specify the subclass of the
# Provider class whose constructor sets the values
# of various properties that are required for the
# Java Security API to look up the algorithms or
# other facilities implemented by the provider.
#
# There must be at least one provider specification
# in java.security. There is a default provider that
# comes standard with the JDK. It is called the "SUN"
# provider, and its Provider subclass named Sun
# appears in the sun.security.provider package.
# Thus, the "SUN" provider is registered via the
# following:
#
#    security.provider.1=sun.security.provider.Sun
#
# (The number 1 is used for the default provider.)
#
# Note: Statically registered Provider subclasses
# are instantiated when the system is initialized.
# Providers can be dynamically registered instead
# by calls to either the addProvider or
# insertProviderAt method in the Security class.
```

```
# List of providers and their preference orders
# (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider

# Class to instantiate as the system Policy.
# This is the name of the class that will be
# used as the Policy object.
#
policy.provider=sun.security.provider.PolicyFile

# The default is to have a single system-wide
# policy file, and a policy file in the user's
# home directory.
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy

# whether or not we expand properties in the
# policy file if this is set to false, properties
#(${...}) will not be expanded in policy files.
policy.expandProperties=true

# whether or not we allow an extra policy to
# be passed on the command line with
# -Djava.security.policy=somefile. Comment out
# this line to disable this feature.
policy.allowSystemProperty=true

# whether or not we look into the IdentityScope
# for trusted Identities when encountering a 1.1
# signed JAR file. If the identity is found
# and is trusted, we grant it AllPermission.
policy.ignoreIdentityScope=false

# Default keystore type.
#
keystore.type=jks

# Class to instantiate as the system scope:
#
system.scope=sun.security.provider.IdentityDatabase

# List of comma-separated packages that start
# with or equal this string will cause a security
# exception to be thrown when passed to
# checkPackageAccess unless the corresponding
# RuntimePermission ("accessClassInPackage."+package)
# has been granted.
package.access=sun.
```

```
# List of comma-separated packages that start with
# or equal this string will cause a security exception
# to be thrown when passed to checkPackageDefinition
# unless the corresponding RuntimePermission
# ("defineClassInPackage."+package) has been granted.
#
# by default, no packages are restricted for definition,
# and none of the class loaders supplied with the JDK
# call checkPackageDefinition.
#
#package.definition=
```

Appendix C -- Example Tomcat 4 catalina.policy file

```
// ========== SYSTEM CODE PERMISSIONS =========================================

// These permissions apply to javac
grant codeBase "file:${java.home}/lib/-" {
  permission java.security.AllPermission;
};

// These permissions apply to all shared system extensions
grant codeBase "file:${java.home}/jre/lib/ext/-" {
  permission java.security.AllPermission;
};

// These permissions apply to javac when ${java.home] points at $JAVA_HOME/jre
grant codeBase "file:${java.home}/../lib/-" {
  permission java.security.AllPermission;
};

// These permissions apply to all shared system extensions when
// ${java.home} points at $JAVA_HOME/jre
grant codeBase "file:${java.home}/lib/ext/-" {
  permission java.security.AllPermission;
};

// ========== CATALINA CODE PERMISSIONS =======================================

// These permissions apply to the server startup code
grant codeBase "file:${catalina.home}/bin/bootstrap.jar" {
  permission java.security.AllPermission;
};

// These permissions apply to the servlet API classes
// and those that are shared across all class loaders
// located in the "common" directory
grant codeBase "file:${catalina.home}/common/-" {
  permission java.security.AllPermission;
};

// These permissions apply to the container's core code, plus any additional
// libraries installed in the "server" directory
grant codeBase "file:${catalina.home}/server/-" {
  permission java.security.AllPermission;
};

// These permissions apply to the jasper page compiler.
grant codeBase "file:${catalina.home}/shared/lib/jasper-compiler.jar" {
  permission java.security.AllPermission;
};

// These permissions apply to the jasper JSP runtime
grant codeBase "file:${catalina.home}/shared/lib/jasper-runtime.jar" {
  permission java.security.AllPermission;
};
```

```
// These permissions apply to the privileged admin and manager web applications
grant codeBase "file:${catalina.home}/server/webapps/admin/WEB-INF/classes/-" {
  permission java.security.AllPermission;
};

grant codeBase "file:${catalina.home}/server/webapps/admin/WEB-INF/lib/struts.jar" {
  permission java.security.AllPermission;
};

// These permissions are granted by default to all web applications
// In addition, a web application will be given a read FilePermission
// and JndiPermission for all files and directories in its document root.
grant {
  // Required for JNDI lookup of named JDBC DataSource's and
  // javamail named MimePart DataSource used to send mail
  permission java.util.PropertyPermission "java.home", "read";
  permission java.util.PropertyPermission "java.naming.*", "read";
  permission java.util.PropertyPermission "javax.sql.*", "read";

  // OS Specific properties to allow read access
  permission java.util.PropertyPermission "os.name", "read";
  permission java.util.PropertyPermission "os.version", "read";
  permission java.util.PropertyPermission "os.arch", "read";
  permission java.util.PropertyPermission "file.separator", "read";
  permission java.util.PropertyPermission "path.separator", "read";
  permission java.util.PropertyPermission "line.separator", "read";

  // JVM properties to allow read access
  permission java.util.PropertyPermission "java.version", "read";
  permission java.util.PropertyPermission "java.vendor", "read";
  permission java.util.PropertyPermission "java.vendor.url", "read";
  permission java.util.PropertyPermission "java.class.version", "read";
  permission java.util.PropertyPermission "java.specification.version", "read";
  permission java.util.PropertyPermission "java.specification.vendor", "read";
  permission java.util.PropertyPermission "java.specification.name", "read";

  permission java.util.PropertyPermission "java.vm.specification.version", "read";
  permission java.util.PropertyPermission "java.vm.specification.vendor", "read";
  permission java.util.PropertyPermission "java.vm.specification.name", "read";
  permission java.util.PropertyPermission "java.vm.version", "read";
  permission java.util.PropertyPermission "java.vm.vendor", "read";
  permission java.util.PropertyPermission "java.vm.name", "read";

  // Required for getting BeanInfo
  permission java.lang.RuntimePermission "accessClassInPackage.sun.beans.*";

  // Required for running servlets generated by JSPC
  permission java.lang.RuntimePermission
    "accessClassInPackage.org.apache.jasper.runtime.*";

  // Required for OpenJMX
  permission java.lang.RuntimePermission "getAttribute";

  // Allow read of JAXP compliant XML parser debug
  permission java.util.PropertyPermission "jaxp.debug", "read";
};
```

```
// You can assign additional permissions to particular web applications by
// adding additional "grant" entries here, based on the code base for that
// application, /WEB-INF/classes/, or /WEB-INF/lib/ jar files.
//
// Different permissions can be granted to JSP pages, classes loaded from
// the /WEB-INF/classes/ directory, all jar files in the /WEB-INF/lib/
// directory, or even to individual jar files in the /WEB-INF/lib/ directory.
//
// For instance, assume that the standard "examples" application
// included a JDBC driver that needed to establish a network connection to the
// corresponding database and used the scrape taglib to get the weather from
// the NOAA web server.  You might create a "grant" entries like this:
//
// The permissions granted to the context root directory apply to JSP pages.
// grant codeBase "file:${catalina.home}/webapps/examples/-" {
//   permission java.net.SocketPermission "dbhost.mycompany.com:5432", "connect";
//   permission java.net.SocketPermission "*.noaa.gov:80", "connect";
//   permission java.io.FilePermission
//     "${catalina.home}${/}webapps${/}examples${/}WEB-INF${/}logs${/}-", "read,write,
// };
//
// The permissions granted to the context WEB-INF/classes directory
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-INF/classes/-" {
//   permission java.io.FilePermission
//     "${catalina.home}${/}webapps${/}examples${/}WEB-INF${/}logs${/}-", "read,write,
// };
//
// The permission granted to your JDBC driver
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-INF/lib/driver.jar" {
//   permission java.net.SocketPermission "dbhost.mycompany.com:5432", "connect";
// };
// The permission granted to the scrape taglib
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-INF/lib/scrape.jar" {
//   permission java.net.SocketPermission "*.noaa.gov:80", "connect";
// };
```

Appendix D -- Example Security Exception Stack Trace

```
java.security.AccessControlException:
  access denied (java.lang.RuntimePermission exitVM)
```

The Java SecurityManager denies permission for the examples web application security.jsp to use System.exit().

```
        at java.security.AccessControlContext.checkPermission
           (AccessControlContext.java:267)
        at java.security.AccessController.checkPermission(AccessController.java:394)
        at java.lang.SecurityManager.checkPermission(SecurityManager.java:540)
        at java.lang.SecurityManager.checkExit(SecurityManager.java:760)
        at java.lang.Runtime.exit(Runtime.java:86)
        at java.lang.System.exit(System.java:696)
```

The security.jsp page is performing something restricted by the Java SecurityManager.

```
        at org.apache.jsp.security_jsp._jspService(security_jsp.java:51)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), common/lib/servlet.jar, common/lib/jasper-runtime.jar, and the examples web application.

```
        at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:136)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.jasper.servlet.JspServletWrapper.service
           (JspServletWrapper.java:204)
        at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:289)
        at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:240)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), common/lib/servlet.jar, and common/lib/jasper-runtime.jar.

```
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), and common/lib/servlet.jar,

```
        at org.apache.catalina.core.ApplicationDispatcher.invoke
           (ApplicationDispatcher.java:684)
        at org.apache.catalina.core.ApplicationDispatcher.doForward
           (ApplicationDispatcher.java:432)
        at org.apache.catalina.core.ApplicationDispatcher.access$0
           (ApplicationDispatcher.java:360)
        at org.apache.catalina.core.ApplicationDispatcher$PrivilegedForward.run
           (ApplicationDispatcher.java:131)
        at java.security.AccessController.doPrivileged(Native Method)
```

AccessController.doPrivileged() resets permissions on the stack to those granted to server/lib/*.jar (catalina) so that the forward can be done with those permissions.

```
at org.apache.catalina.core.ApplicationDispatcher.forward
   (ApplicationDispatcher.java:348)
at org.apache.jasper.runtime.PageContextImpl.forward
   (PageContextImpl.java:427)
```

The forward.jsp page is triggering a forward to security.jsp.

```
at org.apache.jsp.forward_jsp._jspService(forward_jsp.java:50)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), common/lib/servlet.jar, common/lib/jasper-runtime.jar, and the examples web application.

```
at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:136)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
at org.apache.jasper.servlet.JspServletWrapper.service
   (JspServletWrapper.java:204)
at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:289)
at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:240)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), common/lib/servlet.jar, and common/lib/jasper-runtime.jar.

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
```

Permissions are the intersection of those granted to JVM, server/lib/*.jar (catalina), and common/lib/servlet.jar,

```
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter
   (ApplicationFilterChain.java:247)
at org.apache.catalina.core.ApplicationFilterChain.access$0
   (ApplicationFilterChain.java:197)
at org.apache.catalina.core.ApplicationFilterChain$1.run
   (ApplicationFilterChain.java:176)
at java.security.AccessController.doPrivileged(Native Method)
at org.apache.catalina.core.ApplicationFilterChain.doFilter
   (ApplicationFilterChain.java:172)
at org.apache.catalina.core.StandardWrapperValve.invoke
   (StandardWrapperValve.java:260)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:643)
at org.apache.catalina.core.StandardPipeline.invoke
   (StandardPipeline.java:480)
at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995)
at org.apache.catalina.core.StandardContextValve.invoke
   (StandardContextValve.java:191)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:643)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke
   (AuthenticatorBase.java:471)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:641)
```

```
at org.apache.catalina.core.StandardPipeline.invoke
   (StandardPipeline.java:480)
at org.apache.catalina.core.ContainerBase.invoke
   (ContainerBase.java:995)
at org.apache.catalina.core.StandardContext.invoke
   (StandardContext.java:2388)
at org.apache.catalina.core.StandardHostValve.invoke
   (StandardHostValve.java:180)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:643)
at org.apache.catalina.valves.ErrorDispatcherValve.invoke
   (ErrorDispatcherValve.java:170)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:641)
at org.apache.catalina.valves.ErrorReportValve.invoke
   (ErrorReportValve.java:172)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:641)
at org.apache.catalina.valves.AccessLogValve.invoke
   (AccessLogValve.java:469)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:641)
at org.apache.catalina.core.StandardPipeline.invoke
   (StandardPipeline.java:480)
at org.apache.catalina.core.ContainerBase.invoke
   (ContainerBase.java:995)
at org.apache.catalina.core.StandardEngineValve.invoke
   (StandardEngineValve.java:174)
at org.apache.catalina.core.
   StandardPipeline$StandardPipelineValveContext.invokeNext
   (StandardPipeline.java:643)
at org.apache.catalina.core.StandardPipeline.invoke
   (StandardPipeline.java:480)
at org.apache.catalina.core.ContainerBase.invoke
   (ContainerBase.java:995)
at org.apache.coyote.tomcat4.CoyoteAdapter.service
   (CoyoteAdapter.java:223)
at org.apache.coyote.http11.Http11Processor.process
   (Http11Processor.java:405)
at org.apache.coyote.http11.
   Http11Protocol$Http11ConnectionHandler.processConnection
   (Http11Protocol.java:380)
at org.apache.tomcat.util.net.TcpWorkerThread.runIt
   (PoolTcpEndpoint.java:508)
at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run
   (ThreadPool.java:533)
```

Permissions are the intersection of those granted to JVM and server/lib/*.jar (catalina).

```
at java.lang.Thread.run(Thread.java:479)
```

Permissions as granted to JVM.

Appendix E -- Example Security Debug Output

```
access: access denied (java.lang.RuntimePermission exitVM)
java.lang.Exception: Stack trace
        at java.lang.Thread.dumpStack(Thread.java:992)
        at java.security.AccessControlContext.checkPermission
           (AccessControlContext.java:256)
        at java.security.AccessController.checkPermission(AccessController.java:394)
        at java.lang.SecurityManager.checkPermission(SecurityManager.java:540)
        at java.lang.SecurityManager.checkExit(SecurityManager.java:760)
        at java.lang.Runtime.exit(Runtime.java:86)
        at java.lang.System.exit(System.java:696)
        at org.apache.jsp.security_jsp._jspService(security_jsp.java:51)
        at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:136)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.jasper.servlet.JspServletWrapper.service
           (JspServletWrapper.java:204)
        at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:289)
        at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:240)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter
           (ApplicationFilterChain.java:247)
        at org.apache.catalina.core.ApplicationFilterChain.access$0
           (ApplicationFilterChain.java:197)
        at org.apache.catalina.core.ApplicationFilterChain$1.run
           (ApplicationFilterChain.java:176)
        at java.security.AccessController.doPrivileged(Native Method)
        at org.apache.catalina.core.ApplicationFilterChain.doFilter
           (ApplicationFilterChain.java:172)
        at org.apache.catalina.core.StandardWrapperValve.invoke
           (StandardWrapperValve.java:260)
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:643)
        at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480)
        at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995)
        at org.apache.catalina.core.StandardContextValve.invoke
           (StandardContextValve.java:191)
           at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:643)
        at org.apache.catalina.authenticator.AuthenticatorBase.invoke
           (AuthenticatorBase.java:471)
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:641)
        at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480)
        at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995)
        at org.apache.catalina.core.StandardContext.invoke(StandardContext.java:2388)
        at org.apache.catalina.core.StandardHostValve.invoke
           (StandardHostValve.java:180)
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:643)
        at org.apache.catalina.valves.ErrorDispatcherValve.invoke
           (ErrorDispatcherValve.java:170)
```

```
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:641)
        at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:17
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:641)
        at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:469)
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:641)
        at org.apache.catalina.core.
           StandardPipeline.invoke(StandardPipeline.java:480)
        at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995)
        at org.apache.catalina.core.StandardEngineValve.invoke
           (StandardEngineValve.java:174)
        at org.apache.catalina.core.
           StandardPipeline$StandardPipelineValveContext.invokeNext
           (StandardPipeline.java:643)
        at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480)
        at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995)
        at org.apache.coyote.tomcat4.CoyoteAdapter.service(CoyoteAdapter.java:223)
        at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:405)
        at org.apache.coyote.
           http11.Http11Protocol$Http11ConnectionHandler.processConnection
           (Http11Protocol.java:380)
        at org.apache.tomcat.util.net.TcpWorkerThread.runIt(PoolTcpEndpoint.java:508)
        at org.apache.tomcat.util.
           threads.ThreadPool$ControlRunnable.run(ThreadPool.java:533)
        at java.lang.Thread.run(Thread.java:479)
access: domain that failed ProtectionDomain
  (file:/usr/local/tomcat4/webapps/examples/ )
java.security.Permissions@58d39f (
 (java.lang.RuntimePermission accessClassInPackage.sun.*)
 (java.lang.RuntimePermission accessClassInPackage.org.apache.jasper.runtime)
 (java.lang.RuntimePermission accessClassInPackage.org.apache.jasper.resources.*)
 (java.util.PropertyPermission org.apache.xml.* read)
 (java.util.PropertyPermission javax.sql.* read)
 (java.util.PropertyPermission java.version read)
 (java.util.PropertyPermission true read)
 (java.util.PropertyPermission path.separator read)
 (java.util.PropertyPermission java.naming.* read)
 (java.util.PropertyPermission java.vendor read)
 (java.util.PropertyPermission user.dir read)
 (java.util.PropertyPermission line.separator read)
 (java.util.PropertyPermission jaxp.debug read)
 (java.util.PropertyPermission file.separator read)
 (java.util.PropertyPermission javax.xml.* read)
 (java.util.PropertyPermission java.home read)
 (java.util.PropertyPermission java.vendor.url read)
 (java.util.PropertyPermission org.apache.xerces.* read)
 (java.util.PropertyPermission false read)
 (java.io.FilePermission /usr/local/kinetic/java/jre/lib/jaxp.properties read)
 (java.io.FilePermission /usr/local/kinetic/java/jre/lib/castor.properties read)
 (java.io.FilePermission /usr/local/tomcat4/common/endorsed/xalan.jar read)
 (java.io.FilePermission /usr/local/tomcat4/common/endorsed/xercesImpl.jar read)
 (java.io.FilePermission /usr/local/kinetic/java/jre/lib/ext/mail.jar read)
 (java.io.FilePermission /usr/local/tomcat4/webapps/examples/- read)
```

```
(java.io.FilePermission /usr/local/tomcat4/work/Standalone/localhost/examples/- read)
 (java.net.SocketPermission kinetic.more.net:80 connect,resolve)
 (java.net.SocketPermission vortex.more.net:25 connect,resolve)
 (java.net.SocketPermission * resolve)
)
```

Appendix F -- Example experimental XML Security Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Server>

<!-- Default Tomcat Catalina XML Security Policy Configuraton File -->
<!-- Used when Tomcat is started with the -security-xml arg        -->

<Server debug="false">
  <SecurityPolicy name="Server">
    <!-- catalina.jar gets pretty broad permissions, only those which are
         not inherited from child SecurityPolicy elements need to be defined. -->
    <CodeSource codeBase="file:${catalina.home}/server/lib/catalina.jar"/>
    <PropertyPermission name="*" action="read,write"/>
    <RuntimePermission name="*"/>
    <ReflectPermission name="*"/>
    <SecurityPermission name="*"/>
    <NetPermission name="*"/>
    <!-- Permission to create and listen to the default shutdown port -->
    <SocketPermission name="localhost:8005" action="listen"/>
    <!-- Allow catalina to perform file IO in its own directory -->
    <FilePermission name="${catalina.home}${/}-" action="read,write,delete"/>
    <SecurityPolicy name="Connector">
      <!-- Policy for default HTTP connector on port 8080.  If you are using
           mod_webapp or mod_jk you will need to add the appropriate jar files
           and Permissions here. -->
      <CodeSource codeBase="file:${catalina.home}/server/lib/tomcat-coyote.jar"/>
      <CodeSource codeBase="file:${catalina.home}/server/lib/tomcat-util.jar"/>
      <CodeSource codeBase="file:${catalina.home}/server/lib/tomcat-http11.jar"/>
      <!-- Permission to create and listen at the HTTP port 8080 -->
      <SocketPermission name="localhost:8080" action="listen"/>
      <!-- Only HTTP requests from the localhost are accepted.  Change localhost
           below to * to allow HTTP requests to be accepted from anywhere. -->
      <SocketPermission name="localhost:1024-" action="accept"/>
      <PropertyPermission name="catalina.base" action="read"/>
      <RuntimePermission name="accessClassInPackage.org.apache.catalina.*"/>
      <RuntimePermission name="defineClassInPackage.org.apache.catalina.*"/>
      <RuntimePermission name="setContextClassLoader"/>
    </SecurityPolicy>
    <SecurityPolicy name="Castor">
      <!-- Castor is used for reading the XML Policy files -->
      <CodeSource
          codeBase="file:${catalina.home}/server/lib/castor-0.9.3.9-xml.jar"/>
      <RuntimePermission name="accessClassInPackage.org.apache.catalina.*"/>
      <RuntimePermission name="defineClassInPackage.org.apache.catalina.*"/>
    </SecurityPolicy>
    <SecurityPolicy name="Configuration">
      <!-- These jar files and Permissions are required for Tomcat startup
           configuration.  -->
      <CodeSource codeBase="file:${catalina.home}/server/lib/commons-digester.jar"/>
      <CodeSource codeBase="file:${catalina.home}/server/lib/commons-beanutils.jar"/>
      <CodeSource codeBase="file:${catalina.home}/server/lib/commons-modeler.jar"/>
      <CodeSource codeBase="file:${catalina.home}/server/lib/mx4j.jar"/>
      <CodeSource codeBase="file:${catalina.home}/common/lib/commons-logging.jar"/>
      <CodeSource
          codeBase="file:${catalina.home}/common/lib/commons-logging-api.jar"/>
```

```
    <CodeSource codeBase="file:${catalina.home}/common/endorsed/xercesImpl.jar"/>
    <CodeSource codeBase="file:${catalina.home}/common/endorsed/xml-apis.jar"/>
    <PropertyPermission name="*" action="read,write"/>
    <RuntimePermission name="accessDeclaredMembers"/>
    <RuntimePermission name="createMBeanServer"/>
    <RuntimePermission name="defineClassInPackage.org.apache.catalina.*"/>
    <RuntimePermission name="accessClassInPackage.org.apache.catalina.*"/>
    <RuntimePermission name="defineClassInPackage.sun.*"/>
    <FilePermission name="${java.home}${/}..${/}-" action="read"/>
    <FilePermission name="${catalina.home}${/}-" action="read"/>
  </SecurityPolicy>
  <SecurityPolicy name="JNDI">
    <!-- These jar files and Permissions are needed for the Java Naming
         and Directory Interface (JNDI). -->
    <CodeSource codeBase="file:${catalina.home}/common/lib/naming-factory.jar"/>
    <CodeSource codeBase="file:${catalina.home}/common/lib/naming-common.jar"/>
    <CodeSource codeBase="file:${catalina.home}/common/lib/naming-resources.jar"/>
    <RuntimePermission name="defineClassInPackage.org.apache.catalina.*"/>
    <RuntimePermission name="accessClassInPackage.org.apache.catalina.*"/>
    <PropertyPermission name="catalina.base" action="read"/>
    <FilePermission name="${catalina.home}${/}-" action="read"/>
    <FilePermission
        name="${catalina.home}${/}conf${/}tomcat-users.xml" action="write,delete"/>
    <FilePermission name="${catalina.home}${/}conf${/}tomcat-users.xml.new"
          action="write,delete"/>
    <FilePermission name="${catalina.home}${/}conf${/}tomcat-users.xml.old"
          action="write,delete"/>
    <JndiPermission name="*"/>
  </SecurityPolicy>
</SecurityPolicy>
<SecurityPolicy name="Database">
  <!-- Example configuration for database connection pooling using mysql. -->
  <CodeSource codeBase="file:${catalina.home}/common/lib/mm.mysql.jar"/>
  <CodeSource codeBase="file:${catalina.home}/common/lib/commons-dbcp.jar"/>
  <CodeSource codeBase="file:${catalina.home}/common/lib/commons-pool.jar"/>
  <SocketPermission name="localhost:3306" action="connect"/>
</SecurityPolicy>
<Service name="Tomcat-Standalone">
  <!-- The SecurityPolicy's within the service element are for servlets and
       JSP pages.  -->
  <SecurityPolicy name="Servlet">
    <CodeSource codeBase="file:${catalina.home}/common/lib/servlet.jar"/>
    <CodeSource codeBase="file:${catalina.home}/server/lib/servlets-default.jar"/>
    <CodeSource codeBase="file:${catalina.home}/server/lib/servlets-invoker.jar"/>
    <CodeSource codeBase="file:${catalina.home}/server/lib/servlets-manager.jar"/>
    <!-- Needed for parsing HTTP request parameters -->
    <RuntimePermission name="defineClassInPackage.org.apache.catalina.util"/>
    <RuntimePermission name="setContextClassLoader"/>
    <RuntimePermission name="getClassLoader"/>
    <SecurityPolicy name="Jasper">
      <!-- These jar files and Permissions are needed for compiling and
           executing JSP pages. -->
      <CodeSource codeBase="file:${catalina.home}/common/lib/jasper-runtime.jar"/>
      <CodeSource codeBase="file:${catalina.home}/common/lib/jasper-compiler.jar"/>
      <CodeSource codeBase="file:${catalina.home}/common/lib/ant.jar"/>
      <CodeSource codeBase="file:${catalina.home}/common/lib/jasper-compiler.jar"/>
```

```
      <PropertyPermission name="*" action="read,write"/>
      <RuntimePermission name="defineClassInPackage.org.apache.jasper.*"/>
      <RuntimePermission name="defineClassInPackage.org.apache.catalina.*"/>
      <RuntimePermission name="accessClassInPackage.org.apache.jasper.*"/>
      <RuntimePermission name="accessClassInPackage.org.apache.catalina.*"/>
      <RuntimePermission name="setIO"/>
      <RuntimePermission name="createClassLoader"/>
      <SecurityPermission name="getPolicy"/>
      <FilePermission name="${java.home}${/}..${/}-" action="read"/>
      <FilePermission name="${catalina.home}${/}-" action="read"/>
      <FilePermission
          name="${catalina.home}${/}webapps${/}-" action="write,delete"/>
      <FilePermission name="${catalina.home}${/}logs${/}-" action="write"/>
      <FilePermission name="${catalina.home}${/}work${/}-" action="write,delete"/>
      <Permission javaClass="org.apache.naming.JndiPermission" name="*"/>
      <!-- Needed for our example usage of a mysql database -->
      <SocketPermission name="localhost:3306" action="connect"/>
    </SecurityPolicy>
  </SecurityPolicy>
  <Engine name="Standalone">
    <Host name="localhost">
      <Context name="/examples" debug="false">
        <!-- Example security configuration for the examples web application.
             Since this is configured in the global policy.xml this overrides
             any /WEB-INF/policy.xml within the examples web application. -->
        <SecurityPolicy name="Examples">
          <CodeSource codeBase="file:${catalina.home}/webapps/examples/-"/>
          <FilePermission name="${webapp.home}${/}-" action="write"/>
          <FilePermission name="${webapp.work}${/}-" action="write"/>
        </SecurityPolicy>
      </Context>
      <!-- These are the Permissions Context's for this Host are allowed to set.-->
      <Allowed>
        <FilePermission name="${webapp.home}${/}-" action="write"/>
        <FilePermission name="${webapp.work}${/}-" action="write"/>
        <!-- Needed for our example usage of a mysql database -->
        <SocketPermission name="localhost:3306" action="connect"/>
      </Allowed>
    </Host>
  </Engine>
</Service>
<!-- Default permissions granted to all code -->
<Default>
  <!-- Required for JNDI lookup of named JDBC DataSource's and
       javamail named MimePart DataSource used to send mail -->
  <PropertyPermission name="java.home" action="read"/>
  <PropertyPermission name="java.naming.*" action="read"/>
  <PropertyPermission name="javax.sql.*" action="read"/>
  <PropertyPermission name="javax.xml.*" action="read"/>
  <!-- Generic properties we allow to be read -->
  <PropertyPermission name="java.version" action="read"/>
  <PropertyPermission name="java.vendor" action="read"/>
  <PropertyPermission name="java.vendor.url" action="read"/>
  <PropertyPermission name="user.dir" action="read"/>
  <!-- OS Specific properties we allow to be read -->
  <PropertyPermission name="file.separator" action="read"/>
  <PropertyPermission name="path.separator" action="read"/>
  <PropertyPermission name="line.separator" action="read"/>
```

```
    <!-- Xerces and Xalan need these properties -->
    <PropertyPermission name="org.apache.xerces.*" action="read"/>
    <PropertyPermission name="org.apache.xml.*" action="read"/>
    <!-- Needed for using Xerces and Xalan -->
    <FilePermission name="${java.home}${/}lib${/}jaxp.properties" action="read"/>
    <FilePermission name="${catalina.home}${/}common${/}endorsed${/}xercesImpl.jar"
          action="read"/>
    <FilePermission name="${catalina.home}${/}common${/}endorsed${/}xalan.jar"
          action="read"/>
    <!-- Needed by Xalan and Jasper -->
    <RuntimePermission name="accessClassInPackage.sun.*"/>
    <!-- XML parsers and jasper need permission to access jsp and webapp dtd's -->
    <RuntimePermission name="accessClassInPackage.org.apache.jasper.resources.*"/>
    <!-- Needed for parsing HTTP request parameters -->
    <RuntimePermission name="accessClassInPackage.org.apache.catalina.util"/>
    <!-- Needed for Castor -->
    <FilePermission name="${java.home}${/}lib${/}castor.properties" action="read"/>
    <!-- Needed by struts -->
    <PropertyPermission name="jaxp.debug" action="read"/>
    <!-- Let all hostnames be resolved by DNS -->
    <SocketPermission name="*" action="resolve"/>
  </Default>
</Server>
```

Appendix G -- Example experimental web application XML Security Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Webapp>

<Webapp>
  <SecurityPolicy name="DBTags-Examples">
    <!-- The following CodeSource grants the following permissions
         to JSP pages only -->
    <CodeSource codeBase="file:${webapp.home}/*"/>
    <SecurityPolicy name="dbtags.jar">
      <!-- The following CodeSource grants the following permissions
         to the dbtags.jar and the JSP pages by inheritance-->
      <CodeSource codeBase="file:${webapp.home}/WEB-INF/lib/dbtags.jar"/>
      <SocketPermission name="localhost:3306" action="connect"/>
    </SecurityPolicy>
  </SecurityPolicy>
</Webapp>
```