Security

Permalink to this page: https://cwiki.apache.org/confluence/x/qyolBg

Preface

This FAQ section provides help with some security-related issues. If you hear of a vulnerability or its exploitation, please see the security page.

The Record

There have been no public cases of damage done to a company, organization, or individual due to a Tomcat security issue. There have been no documented cases of data loss or application crashes caused by an intruder. While there have been numerous analyses conducted on Tomcat, partially because this is easy to do with Tomcat's source code openly available, there have been only **theoretical** vulnerabilities found. All of those were addressed even though there were no documented cases of actual exploitation of these vulnerabilities.

That said,

- There have been several reports of a compromise done via guess of the password of a user of the Manager web application.

  There was once a bug that blindly clicking-trough the Windows installer configured a manager user with blank password (CVE-2009-3548). This was fixed by April 2010 (Tomcat 5.5.29, 6.0.24 and later are safe).

  Please see "Security considerations" pages in Tomcat documentation (linked below) for a reference on how access to Management Applications in Tomcat should be secured.

- There have been several reports of compromises via vulnerabilities in 3-rd party web applications deployed on Tomcat. E.g. vulnerabilities in Apache Struts framework were a popular attack target several times in years 2013-2017. E.g. Equifax breach in year 2017. It is unknown whether Equifax has run their application on Tomcat, but there have been a number of similar compromise reports from Tomcat users. Those are not caused by a vulnerability in Tomcat.

Role of Customization

We believe, and the evidence suggests, that Tomcat is more than secure enough for most use-cases. However, like all other components of Tomcat, you can customize any and all of the relevant parts of the server to achieve even higher security. For example, the session manager implementation is pluggable, and even the default implementation has support for pluggable random number generators. If you have a special need that you feel is not met by Tomcat out of the box, consider these customization options. At the same time, please bring up your requirements on the user mailing list, where we'll be glad to discuss it and assist in your approach/design/implementation as needed.

It is also possible to configure Tomcat insecurely. Please see "Security considerations" pages in Tomcat documentation (linked below) for the list of security-sensitive options.

Links

- Known vulnerabilities: https//tomcat.apache.org/security.html
- Security considerations (Tomcat documentation) — Tomcat 9, Tomcat 8.5, Tomcat 7.

Questions

Answers

How do I use OpenSSL to set up my own Certificate Authority (CA)?

Using OpenSSL to set up your own CA.

Oh no! Port 8005 is available for anyone on localhost to shutdown my tomcat!

See these 2 discussions.

- Possible to switch off tcp/ip server shutdown?
- Tomcat shutdown & security

What about Tomcat running as root?

See these threads:

- Tomcat as root and security issues

How do I force all my pages to run under HTTPS?

Use security-constraint in web.xml.

What is the default login for the manager and admin app?

The admin and manager application do not provide a default login. Doing so would be a security flaw. You need to edit $CATALINA_HOME/conf/tomcat-users.xml file if you are using the default install. See Configuring Manager Application Access for details.

Note that there exists malware that tries to guess the manager password.

There was once a bug that blindly clicking-trough the Windows installer configured a manager user with blank password (CVE-2009-3548). This was fixed by April 2010 (Tomcat 5.5.29, 6.0.24 and later are safe).

How do I restrict access by ip address or remote host?

By using the `RemoteHostValve` or `RemoteAddrValve`. Warning, these valves rely on accurate incoming ip addresses or hostnames. So they can fall victim to spoofing! See also `RemoteIpValve`. Valve Reference Link

How do I use jsvc/procrun to run Tomcat on port 80 securely?

Fairly easily 😉 See the Setup page in the docs for your tomcat release, and read this mailing list post for a complete setup example with permissions etc.

Has Tomcat's security been independently analyzed or audited?

Yes, by numerous organizations and individuals, many times. Try this Google search and you'll see many references, guides, and analyses.

How do I change the Server header in the response?

In `server.xml` file add a "server" attribute to the Connector element. https://tomcat.apache.org/tomcat-9.0-doc/config/http.html

Why are passwords in plain text?

We have a page dedicated to this topic. Password

How can I restrict the list of ciphers used for HTTPS?

See HowTo SSLCiphers.

Which cipher suites should I use?

See Security/Ciphers.

Is Tomcat affect by Log4Shell CVE-2021-44228?

Out of the box - No.

But that doesn't prevent an application deployed to Tomcat from using log4j2. In which case, please use general guidance on various remediations. As of this writing, they include any of these

- Upgrading log4j2 to 2.16.0 (or better)
  - This is the best fix (2.15.0 was also insufficient due to CVE-2021-45046)
- Use system property log4j2.formatMsgNoLookups=true to disable message formatting
  - This is a reasonable workaround - But due to CVE-2021-45046 you may still have other vulnerabilities
  - Setting a shell environment variable `LOG4J_FORMAT_MSG_NO_LOOKUPS=true` should work too
- Remove the following files from your jar file: log4j-core-2.XX.Y.jar (This is a hack. A simple one but effective. This also remediates CVE-2021-45046)
  - `org/apache/logging/log4j/core/net/JndiManager$1.class`

    - `org/apache/logging/log4j/core/util/JndiCloser.class`
    - `org/apache/logging/log4j/core/net/JndiManager$JndiManagerFactory.class`
    - `org/apache/logging/log4j/core/lookup/JndiLookup.class`
    - `org/apache/logging/log4j/core/net/JndiManager.class`
    - `org/apache/logging/log4j/core/selector/JndiContextSelector.class`
- Ensuring your JRE is 1.8.121 or better. (This version doesn't fix the issue, it just eliminates one very easy to exploit attack vector.)
    - Not recommended but better than nothing. When combined with the limiting of creating socket connections, this will make payloads harder (but not impossible) to deploy.
    - But these 2 system properties must also be set to false (or unset) com.sun.jndi.rmi.object.trustURLCodebase,com.sun.jndi.cosnaming.object.trustURLCodebase
    - Newer JVM's can still fall victim to this attack with the older log4j2. It is beyond the scope of this FAQ to explain how. But the TL;DR is it the exploit becomes more application or application server specific.

More details on these CVE's via the ASF blog

I found a vulnerability in JMXProxy

JMXProxy is a powerful servlet which has full access to all JMX capabilities. **By design**, enabling it opens you to a lot of security challenges. The equivalent of enabling generic remote JMX access at the JVM level.

With that in mind, if you enable it: You should **at a minimum** require an extremely strong password to protect this URL as well restrict the IP client list which may access it. (Ideally restricting it to localhost if possible)