Hewlett Packard Enterprise

HPE Security Fortify Standalone Report Generator

Developer Workbook

tasy-agent

FORTIFY

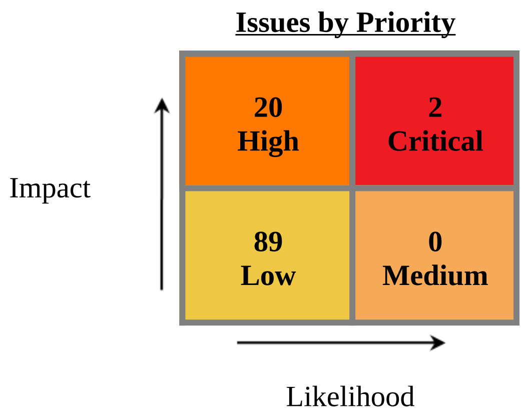
Table of Contents

Executive Summary

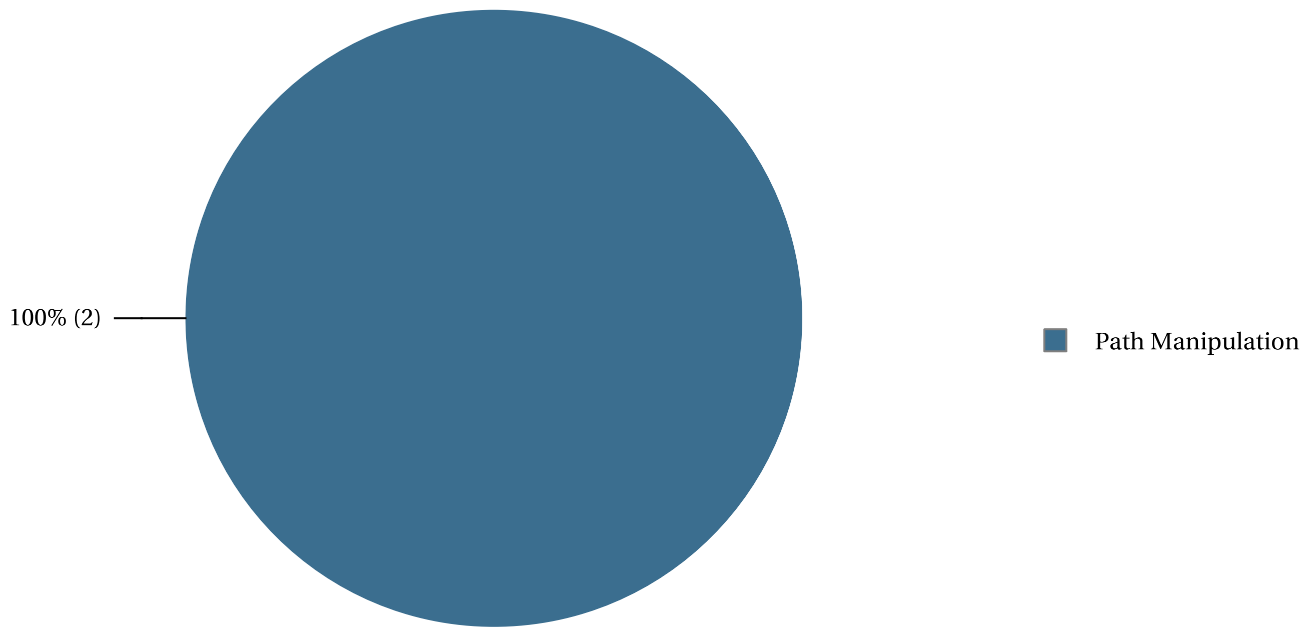
This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the tasy-agent project audit. The information contained in this workbook is targeted at project managers and developers.

This section provides an overview of the issues uncovered during analysis.

Project Name:	tasy-agent
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
SecurityScope:	Results Not Present
Other:	Results Not Present



Top Ten Critical Categories



Project Description

This section provides an overview of the HPE Security Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	Feb 1, 2018, 7:44 AM	**Engine Version:**	17.10.0156
Host Name:	srv-sec-protex.whebdc.com.br	**Certification:**	VALID
Number of Files:	76	**Lines of Code:**	3,524

Issue Breakdown by Fortify Categories

The following table depicts a summary of all issues grouped vertically by Fortify Category. For each category, the total number of issues is shown by Fortify Priority Order, including information about the number of audited issues.

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Access Specifier Manipulation	0	0 / 2	0	0	0 / 2
Code Correctness: Class Does Not Implement equals	0	0	0	0 / 1	0 / 1
Command Injection	0	0	0	0 / 1	0 / 1
Dead Code: Expression is Always false	0	0	0	0 / 1	0 / 1
Dead Code: Unused Field	0	0	0	0 / 7	0 / 7
Dead Code: Unused Method	0	0	0	0 / 8	0 / 8
Denial of Service	0	0	0	0 / 1	0 / 1
J2EE Bad Practices: JVM Termination	0	0	0	0 / 2	0 / 2
J2EE Bad Practices: Leftover Debug Code	0	0	0	0 / 2	0 / 2
J2EE Bad Practices: Sockets	0	0	0	0 / 2	0 / 2
J2EE Bad Practices: Threads	0	0	0	0 / 13	0 / 13
Missing Check against Null	0	0	0	0 / 1	0 / 1
Often Misused: Authentication	0	0 / 2	0	0	0 / 2
Path Manipulation	0 / 2	0 / 8	0	0	0 / 10
Poor Error Handling: Empty Catch Block	0	0	0	0 / 1	0 / 1
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 13	0 / 13
Poor Error Handling: Overly Broad Throws	0	0	0	0 / 27	0 / 27
Poor Error Handling: Throw Inside Finally	0	0	0	0 / 1	0 / 1
Poor Logging Practice: Use of a System Output Stream	0	0	0	0 / 5	0 / 5
Race Condition: Singleton Member Field	0	0 / 1	0	0	0 / 1
System Information Leak: Incomplete Servlet Error Handling	0	0	0	0 / 2	0 / 2
Unreleased Resource: Sockets	0	0 / 2	0	0	0 / 2
Unreleased Resource: Streams	0	0 / 5	0	0	0 / 5
Weak Cryptographic Hash: Missing Required Step	0	0	0	0 / 1	0 / 1

Results Outline

Access Specifier Manipulation (2 issues)

Abstract

The method call changes an access specifier.

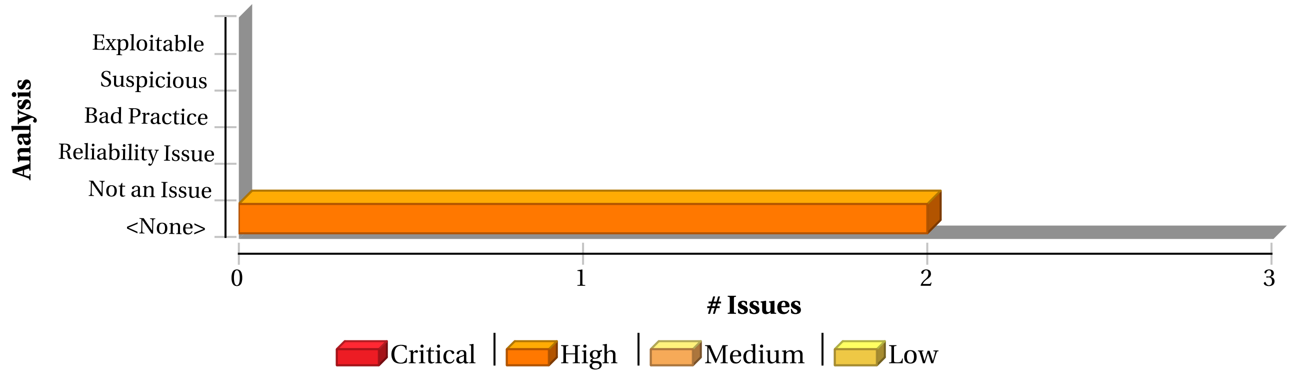
Explanation

The AccessibleObject API allows the programmer to get around the access control checks provided by Java access specifiers. In particular it enables the programmer to allow a reflected object to bypass Java access controls and in turn change the value of private fields or invoke private methods, behaviors that are normally disallowed.

Recommendation

Access specifiers should only be changed by a privileged class using arguments that an attacker cannot set. All occurrences should be examined carefully.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Access Specifier Manipulation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Access Specifier Manipulation	**High**
Package: com.philips.tasy.agent.client.core.server	
client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java, line 57 (Access Specifier Manipulation)	**High**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Access Specifier Manipulation — High

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java, line 57 (Access Specifier Manipulation) — High

Sink: setAccessible()
Enclosing Method: getValueFromField()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java:57
Taint Flags:

```
54 throws IllegalAccessException, NoSuchFieldException {
55 Objects.requireNonNull(fieldName);
56 final Field f = from.getDeclaredField(fieldName);
57 f.setAccessible(true);
58 return f.get(base);
59 }
60
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java, line 202 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: finalizeNativeLibs()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java:202
Taint Flags:

```
199 try {
200 final Method finalize = lib.getClass().getDeclaredMethod("finalize");
201
202 finalize.setAccessible(true);
203 finalize.invoke(lib);
204
205 } catch (IllegalAccessException | InvocationTargetException | NoSuchMethodException e) {
```

Code Correctness: Class Does Not Implement equals (1 issue)

Abstract

The `equals()` method is called on an object that does not implement `equals()`.

Explanation

When comparing objects, developers usually want to compare properties of objects. However, calling `equals()` on a class (or any super class/interface) that does not explicitly implement `equals()` results in a call to the `equals()` method inherited from `java.lang.Object`. Instead of comparing object member fields or other properties, `Object.equals()` compares two object instances to see if they are the same. Although there are legitimate uses of `Object.equals()`, it is often an indication of buggy code.

Example 1:

```
public class AccountGroup
{
   private int gid;

  public int getGid()
    {
      return gid;
    }

 public void setGid(int newGid)
 {
      gid = newGid;
  }
}
...
public class CompareGroup
{
    public boolean compareGroups(AccountGroup group1, AccountGroup group2)
 {
      return group1.equals(group2);   //equals() is not implemented in
AccountGroup
  }
}
```

Recommendation

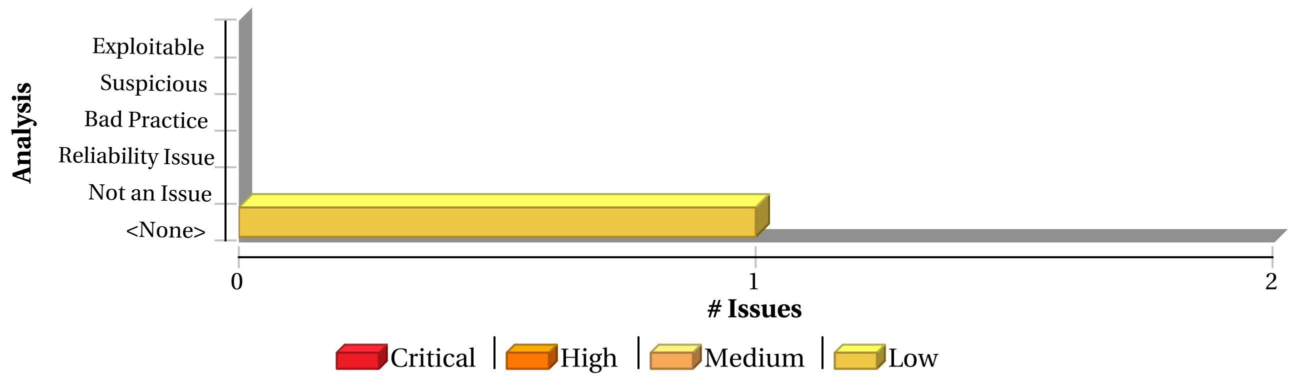
Verify that the use of `Object.equals()` is really the method you intend to call. If not, implement an `equals()` method or use a different method for comparing objects.

Example 2: The following code adds an `equals()` method to the example from the Explanation section.

```
public class AccountGroup
{
  private int gid;

  public int getGid()
    {
      return gid;
    }
```

```
 public void setGid(int newGid)
 {
      gid = newGid;
  }

 public boolean equals(Object o)
    {
      if (!(o instanceof AccountGroup))
          return false;
      AccountGroup other = (AccountGroup) o;
     return (gid == other.getGid());
    }
}
...
public class CompareGroup
{
    public static boolean compareGroups(AccountGroup group1, AccountGroup
group2)
  {
      return group1.equals(group2);
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Class Does Not Implement equals	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: Class Does Not Implement equals **Low**

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ OnlineApplicationContainer.java, line 186 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 186 (Code Correctness: Class Does Not Implement equals) — **Low**

Sink Details

Sink: FunctionCall: equals
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java:186
Taint Flags:

```
183 try {
184 final ClassLoader base;
185 if (TasyAgentConfiguration.getExtraClassPathLoader() != null
186 && webAppContext.getContextPath().equals("/admin")) {
187 base = TasyAgentConfiguration.getExtraClassPathLoader();
188 } else {
189 base = OnlineApplicationContainer.this.getClass().getClassLoader();
```

Command Injection (1 issue)

Abstract

Executing commands that include unvalidated user input can cause an application to execute malicious commands on behalf of an attacker.

Explanation

Command injection vulnerabilities take two forms:

- An attacker can change the command that the program executes: the attacker explicitly controls what the command is.

- An attacker can change the environment in which the command executes: the attacker implicitly controls what the command means.

In this case we are primarily concerned with the second scenario, the possibility that an attacker may be able to change the meaning of the command by changing an environment variable or by putting a malicious executable early in the search path. Command injection vulnerabilities of this type occur when:

1. An attacker modifies an application's environment.

2. The application executes a command without specifying an absolute path or verifying the binary being executed.

3. By executing the command, the application gives an attacker a privilege or capability that the attacker would not otherwise have.

Example: The following code is from a web application that allows users access to an interface through which they can update their password on the system. Part of the process for updating passwords in certain network environments is to run a `make` command in the `/var/yp` directory, the code for which is shown below.

```
...
System.Runtime.getRuntime().exec("make");
...
```

The problem here is that the program does not specify an absolute path for make and fails to clean its environment prior to executing the call to `Runtime.exec()`. If an attacker can modify the `$PATH` variable to point to a malicious binary called `make` and then execute the application in their environment, the malicious binary will be loaded instead of the one intended. Because of the nature of the application, it runs with the privileges necessary to perform system operations, which means the attacker's `make` will now be run with these privileges, possibly giving them complete control of the system.

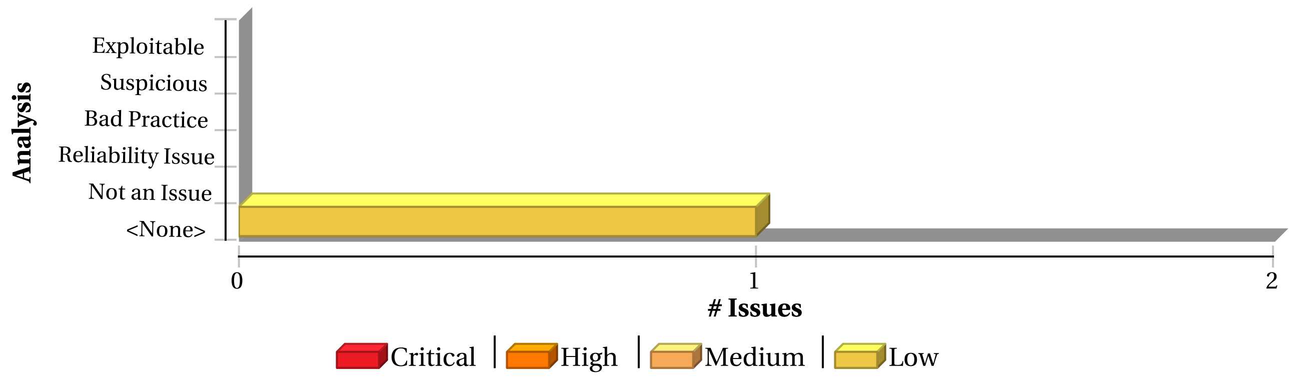
Recommendation

An attacker may indirectly control commands executed by a program by modifying the environment in which they are executed. The environment should not be trusted and precautions should be taken to prevent an attacker from using some manipulation of the environment to perform an attack. Whenever possible, commands should be controlled by the application and executed using an absolute path. In cases where the path is not known at compile time, such as for cross-platform applications, an absolute path should be constructed from trusted values during execution. Command values and paths read from configuration files or the environment should be sanity-checked against a set of invariants that define valid values.

Other checks can sometimes be performed to detect if these sources may have been tampered with. For example, if a configuration file is world-writable, the program might refuse to run. In cases where information about the binary to be executed is known in advance, the program may perform checks to verify the identity of the binary. If a binary should always be owned by a particular user or have a particular set of access permissions assigned to it, these properties can be verified programmatically before the binary is executed.

In the end it may be impossible for a program to fully protect itself from an imaginative attacker bent on controlling the commands it executes. You should strive to identify and protect against every conceivable manipulation of input values and the environment. The goal should be to shutdown as many attack vectors as possible.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Command Injection	1	0	0	1
Total	**1**	**0**	**0**	**1**

Command Injection — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 208 (Command Injection) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: ProcessBuilder(0)
Enclosing Method: updateClient()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:208
Taint Flags:

```
205 if (Files.isRegularFile(updatebat)) {
206 updatebat = Paths.get(System.getProperty(SysPropertiesNames.AGENT_BASE)).relativize(updatebat);
207 LOGGER.info("Runnning update.bat!");
208 final ProcessBuilder processBuilder = new ProcessBuilder("cmd", "/c", updatebat.toString());
209 processBuilder.environment().put("INSTDIR", System.getProperty(SysPropertiesNames.AGENT_BASE));
210 processBuilder.redirectError(ProcessBuilder.Redirect.INHERIT);
```

Command Injection	Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/ UpdateService.java, line 208 (Command Injection)	Low

```
211 processBuilder.redirectOutput(ProcessBuilder.Redirect.INHERIT);
```

Dead Code: Expression is Always false (1 issue)

Abstract

This expression (or part of it) will always evaluate to `false`.

Explanation

This expression (or part of it) will always evaluate to `false`; the program could be rewritten in a simpler form. The nearby code may be present for debugging purposes, or it may not have been maintained along with the rest of the program. The expression may also be indicative of a bug earlier in the method.

Example 1: The following method never sets the variable `secondCall` after initializing it to `false`. (The variable `firstCall` is mistakenly used twice.) The result is that the expression `firstCall && secondCall` will always evaluate to `false`, so `setUpDualCall()` will never be invoked.

```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  if (fCall > 0) {
    setUpFCall();
    firstCall = true;
  }
  if (sCall > 0) {
    setUpSCall();
    firstCall = true;
  }

  if (firstCall && secondCall) {
    setUpDualCall();
  }
}
```

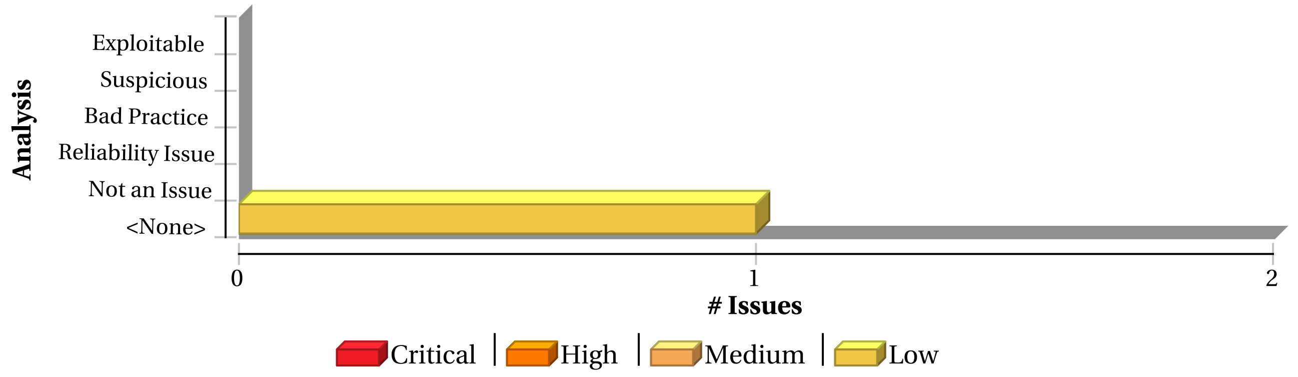
Example 2: The following method never sets the variable `firstCall` to `true`. (The variable `firstCall` is mistakenly set to `false` after the first conditional statement.) The result is that the first part of the expression `firstCall && secondCall` will always evaluate to `false`.

```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  if (fCall > 0) {
    setUpFCall();
    firstCall = false;
  }
  if (sCall > 0) {
    setUpSCall();
    secondCall = true;
  }

  if (firstCall || secondCall) {
    setUpForCall();
    }
}
```

Recommendation

In general, you should repair or remove unused code. It causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Expression is Always false	1	0	0	1
Total	**1**	**0**	**0**	**1**

Dead Code: Expression is Always false — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 243 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: clientPackVersions()
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:243
Taint Flags:

```
240 result = Stream.empty();
241 } else {
242 final File[] list = file.listFiles((d, name) -> CLIENT_PACK_PATTERN.asPredicate().test(name));
243 result = list == null ? Stream.empty() : Arrays.stream(list);
244 }
245 return result;
246 }
```

Dead Code: Unused Field (7 issues)

Abstract

This field is never used.

Explanation

This field is never accessed, except perhaps by dead code. Dead code is defined as code that is never directly or indirectly executed by a public method. It is likely that the field is simply vestigial, but it is also possible that the unused field points out a bug.

Example 1: The field named `glue` is not used in the following class. The author of the class has accidentally put quotes around the field name, transforming it into a string constant.

```
public class Dead {

  String glue;

  public String getGlue() {
    return "glue";
  }

}
```

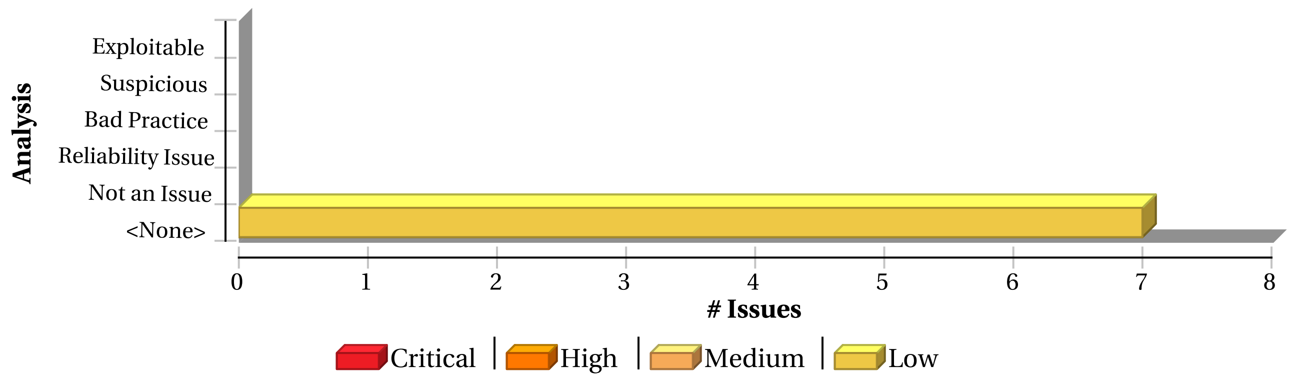
Example 2: The field named `glue` is used in the following class, but only from a method that is never called.

```
public class Dead {

  String glue;

  private String getGlue() {
    return glue;
  }

}
```

Recommendation

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Field	7	0	0	7
Total	**7**	**0**	**0**	**7**

Dead Code: Unused Field — Low

Package: .com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 96 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: obj
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:96
Taint Flags:

```
93 );
94 installedApps.installedModules()
95 .map(this::checkIfAppHasUpdate)
96 .filter(Optional::isPresent)
97 .map(Optional::get)
98 .forEach(response::addModule);
99
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 106 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: obj
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:106

Dead Code: Unused Field — Low

Package: .com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 106 (Dead Code: Unused Field) — **Low**

Taint Flags:

```
103 private Optional<App> isLatestClientModule(final App clientModule) {
104 return clientPackVersions()
105 .map(this::extractClientPackVersion)
106 .filter(Optional::isPresent)
107 .map(Optional::get)
108 .map(App::clientModule)
109 .max((l, r) -> l.getVersion().compareTo(r.getVersion()))
```

Package: com.philips.tasy.agent.server

server-boot/src/main/java/com.philips.tasy.agent.server/TasyAgentServer.java, line 17 (Dead Code: Unused Field) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: help
File: server-boot/src/main/java/com.philips.tasy.agent.server/TasyAgentServer.java:17
Taint Flags:

```
14 */
15 public class TasyAgentServer {
16 @Parameter(names = {"--help", "-h"}, help = true, description = "Display usage info")
17 private boolean help;
18
19 public static void main(final String... args) throws Exception {
20
```

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 57 (Dead Code: Unused Field) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: modulesProvider
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:57
Taint Flags:

```
54 @Inject
```

Dead Code: Unused Field — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 57 (Dead Code: Unused Field) — Low

```
55 private TasyAgentConfiguration configuration;
56 @Inject
57 private Provider<Modules> modulesProvider;
58 @Inject
59 private ObjectMapper mapper;
60
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 53 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: em
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:53
Taint Flags:

```
50 private static final Pattern CLIENT_PACK_PATTERN = Pattern.compile("client-pack-(\\d*)\\.(\\d*)\\.(\\d*)\\.zip");
51 private static final PhilipsLogger LOGGER = PhilipsLogger.getLogger(Update.class);
52 @Inject
53 private EntityManager em;
54 @Inject
55 private TasyAgentConfiguration configuration;
56 @Inject
```

server/src/main/java/com/philips/tasy/agent/server/services/Module.java, line 24 (Dead Code: Unused Field) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: libsProvider
File: server/src/main/java/com/philips/tasy/agent/server/services/Module.java:24
Taint Flags:

```
21 private final LibModule libModule;
22 private final EntityManager em;
23 @Inject
24 private Provider<Libs> libsProvider;
25
26 @Inject
```

Dead Code: Unused Field — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Module.java, line 24 (Dead Code: Unused Field) — **Low**

```
27 public Module(@Assisted final LibModule libModule, final EntityManager em) {
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 59 (Dead Code: Unused Field) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: mapper
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:59
Taint Flags:

```
56 @Inject
57 private Provider<Modules> modulesProvider;
58 @Inject
59 private ObjectMapper mapper;
60
61 @SuppressWarnings("PMD.UnnecessaryFullyQualifiedName")
62 private static FileSystem createZipFileSystem(final java.nio.file.Path zipPath) {
```

Dead Code: Unused Method (8 issues)

Abstract

This method is not reachable from any method outside the class.

Explanation

This method is never called or is only called from other dead code.

Example 1: In the following class, the method `doWork()` can never be called.

```
public class Dead {
  private void doWork() {
    System.out.println("doing work");
  }
  public static void main(String[] args) {
    System.out.println("running Dead");
  }
}
```

Example 2: In the following class, two private methods call each other, but since neither one is ever invoked from anywhere else, they are both dead code.

```
public class DoubleDead {
  private void doTweedledee() {
    doTweedledumb();
  }
  private void doTweedledumb() {
    doTweedledee();
  }
  public static void main(String[] args) {
    System.out.println("running DoubleDead");
  }
}
```

(In this case it is a good thing that the methods are dead: invoking either one would cause an infinite loop.)

Recommendation

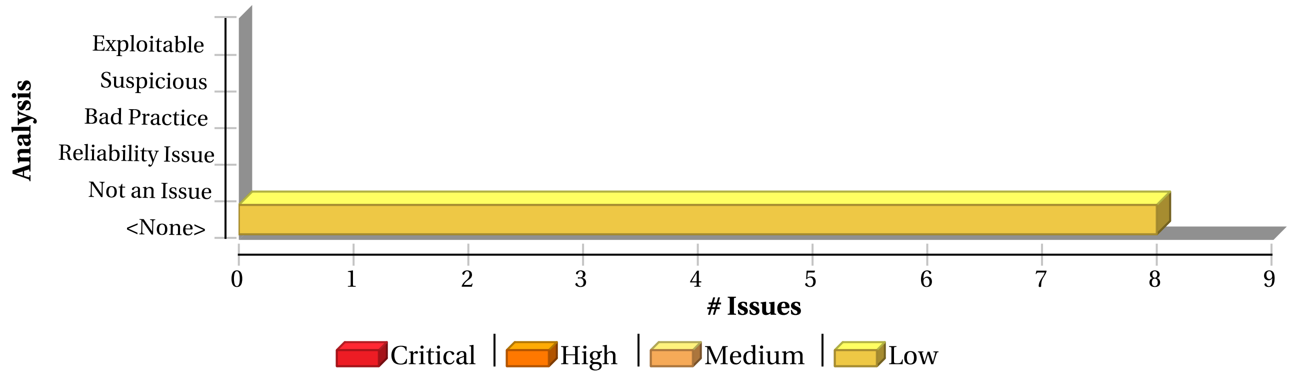
A dead method may indicate a bug in dispatch code.

Example 3: If method is flagged as dead named `getWitch()` in a class that also contains the following dispatch method, it may be because of a copy-and-paste error. The 'w' case should return `getWitch()` not `getMummy()`.

```
public ScaryThing getScaryThing(char st) {
  switch(st) {
    case 'm':
      return getMummy();
    case 'w':
      return getMummy();
    default:
      return getBlob();
  }
}
```

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Method	8	0	0	8
Total	**8**	**0**	**0**	**8**

Dead Code: Unused Method — Low

Package: com.philips.tasy.agent.client

client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java, line 95 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getCommand
Enclosing Method: getCommand()
File: client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java:95
Taint Flags:

```
92
93 }
94
95 private SystemCLICommand getCommand(final String name) {
96 return commands.get(name);
97 }
98
```

Dead Code: Unused Method	Low

Package: com.philips.tasy.agent.client

client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java, line 95 (Dead Code: Unused Method)	Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java, line 43 (Dead Code: Unused Method)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getYmlObjectMapper
Enclosing Method: getYmlObjectMapper()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java:43
Taint Flags:

```
40 return getYmlObjectMapper().readValue(file, TasyAgentConfiguration.class);
41 }
42
43 private static ObjectMapper getYmlObjectMapper() {
44 return new ObjectMapper(new YAMLFactory());
45 }
46
```

Package: com.philips.tasy.agent.server.commands

server-boot/src/main/java/com.philips.tasy.agent.server/commands/RunCommand.java, line 59 (Dead Code: Unused Method)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: stopServiceManager
Enclosing Method: stopServiceManager()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/RunCommand.java:59
Taint Flags:

```
56 }
57 }
58
59 private void stopServiceManager() {
60 sm.stopAsync();
61 }
62 }
```

Dead Code: Unused Method — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 189 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: packModules
Enclosing Method: packModules()
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:189
Taint Flags:

```
186 });
187 }
188
189 private void packModules(final UpdateResult finalAppsToUpdate, final ZipOutputStream os) {
190 finalAppsToUpdate.getModules().stream()
191 .map(moduleToUpdate -> {
192 final Module module = modulesProvider.get()
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 62 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: createZipFileSystem
Enclosing Method: createZipFileSystem()
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:62
Taint Flags:

```
59 private ObjectMapper mapper;
60
61 @SuppressWarnings("PMD.UnnecessaryFullyQualifiedName")
62 private static FileSystem createZipFileSystem(final java.nio.file.Path zipPath) {
63 // convert the filename to a URI
64 final URI uri = URI.create("jar:file:" + zipPath.toUri().getPath());
65 final Map<String, String> env = new HashMap<>();
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 207 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Unused Method — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 207 (Dead Code: Unused Method) — Low

Sink Details

Sink: Function: serializeIndex
Enclosing Method: serializeIndex()
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:207
Taint Flags:

```
204 .forEach(zipConsumer(os, "modules"));
205 }
206
207 private void serializeIndex(final Object finalAppsToUpdate, final ZipOutputStream os) throws IOException {
208 final ZipEntry ze = new ZipEntry("index.json");
209 os.putNextEntry(ze);
210 /* could write directly, but ObjectMapper states that
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 216 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: zipConsumer
Enclosing Method: zipConsumer()
File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:216
Taint Flags:

```
213 os.closeEntry();
214 }
215
216 private Consumer<File> zipConsumer(final ZipOutputStream os, final String folderName) {
217 final Function<File, ZipEntry> zipEntryFunction;
218 if (Strings.isNullOrEmpty(folderName)) {
219 zipEntryFunction = (f) -> new ZipEntry(f.getName());
```

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 154 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: packClientUpdate
Enclosing Method: packClientUpdate()

Dead Code: Unused Method — Low

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Update.java, line 154 (Dead Code: Unused Method) — Low

File: server/src/main/java/com/philips/tasy/agent/server/services/Update.java:154
Taint Flags:

```
151 return rs;
152 }
153 @SuppressWarnings("PMD.UnnecessaryFullyQualifiedName")
154 private void packClientUpdate(final UpdateResult finalAppsToUpdate, final ZipOutputStream os) {
155 finalAppsToUpdate.mayGetNewClientVersion()
156 .map(newClientVersion ->
157 clientPackVersions()
```

Denial of Service (1 issue)

Abstract

An attacker could cause the program to crash or otherwise become unavailable to legitimate users.

Explanation

Attackers may be able to deny service to legitimate users by flooding the application with requests, but flooding attacks can often be defused at the network layer. More problematic are bugs that allow an attacker to overload the application using a small number of requests. Such bugs allow the attacker to specify the quantity of system resources their requests will consume or the duration for which they will use them.

Example 1: The following code allows a user to specify the amount of time for which a thread will sleep. By specifying a large number, an attacker may tie up the thread indefinitely. With a small number of requests, the attacker may deplete the application's thread pool.

```
  int usrSleepTime = Integer.parseInt(usrInput);
  Thread.sleep(usrSleepTime);
```

Example 2: The following code reads a String from a zip file. Because it uses the `readLine()` method, it will read an unbounded amount of input. An attacker may take advantage of this code to cause an `OutOfMemoryException` or to consume a large amount of memory so that the program spends more time performing garbage collection or runs out of memory during some subsequent operation.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);
  String line = br.readLine();
```

Recommendation

Validate user input to ensure that it will not cause inappropriate resource utilization.

Example 3: The following code allows a user to specify the amount of time for which a thread will sleep just as in Example 1, but only if the value is within reasonable bounds.

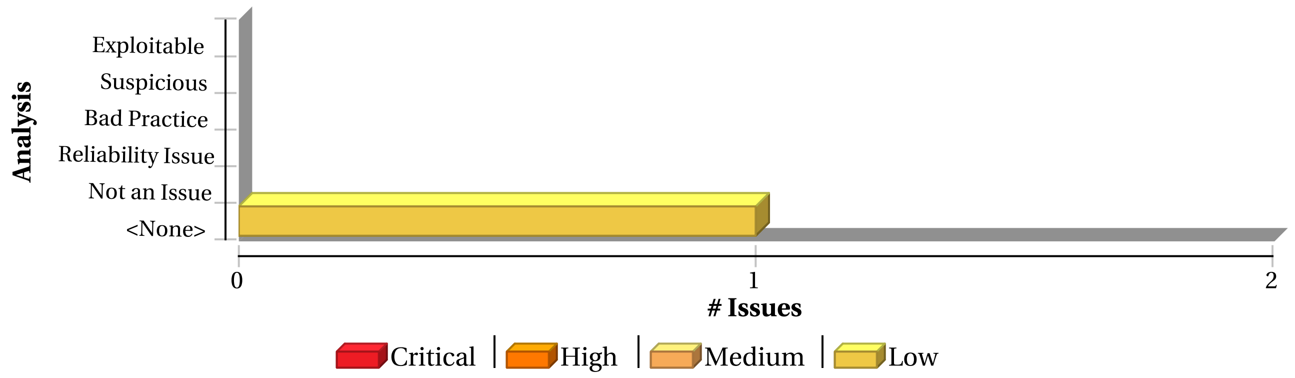
```
  int usrSleepTime = Integer.parseInt(usrInput);
  if (usrSleepTime >= SLEEP_MIN &&
      usrSleepTime <= SLEEP_MAX) {
    Thread.sleep(usrSleepTime);
  } else {
    throw new Exception("Invalid sleep duration");
  }
}
```

Example 4: The following code reads a String from a zip file just as in Example 2, but the maximum string length it will read is `MAX_STR_LEN` characters.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
```

```
Reader zipReader = new InputStreamReader(zipInput);
BufferedReader br = new BufferedReader(zipReader);
StringBuffer sb = new StringBuffer();
int intC;
while ((intC = br.read()) != -1) {
  char c = (char) intC;
  if (c == '\n') {
    break;
  }
  if (sb.length() >= MAX_STR_LEN) {
    throw new Exception("input too long");
  }
  sb.append(c);
}
String line = sb.toString();
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Denial of Service	1	0	0	1
Total	**1**	**0**	**0**	**1**

Denial of Service — Low

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 64 (Denial of Service) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: readLine()
Enclosing Method: run()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:64
Taint Flags:

```
61  final BufferedReader reader = new BufferedReader(
```

Denial of Service — Low

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 64 (Denial of Service) — Low

```
new InputStreamReader(accept.getInputStream(), CHARSET_NAME)
);
final String s = reader.readLine();
if ("STOP".equals(s)) {
try {
LOGGER.info("*** stopping jetty embedded server");
```

J2EE Bad Practices: JVM Termination (2 issues)

Abstract

A web application should not attempt to shut down its container.

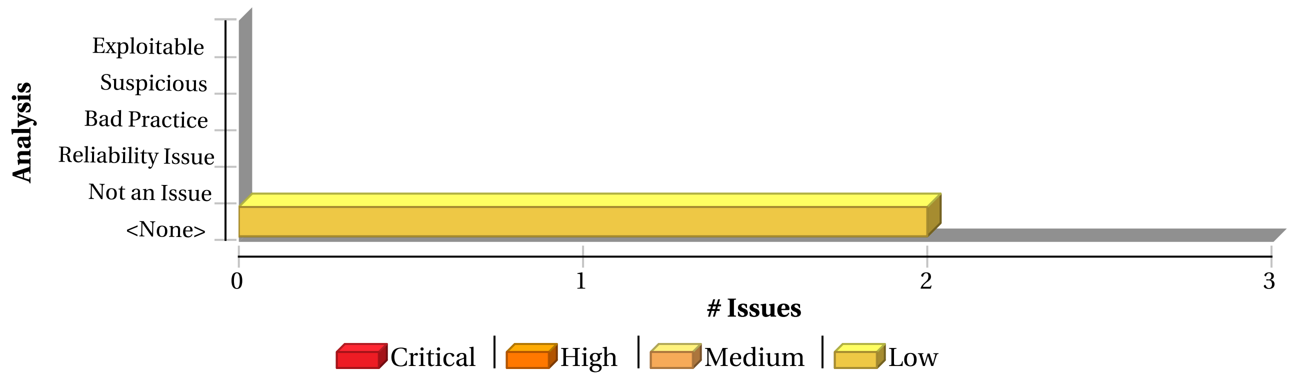
Explanation

It is never a good idea for a web application to attempt to shut down the application container. A call to a termination method is probably part of leftover debug code or code imported from a non-J2EE application.

Recommendation

Never call a termination method within a web application. Such method calls in a J2EE application indicates poor software hygiene and should be removed. Regardless of whether there is a perceived threat, it is unlikely that there is a legitimate reason for such code to remain in the application.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: JVM Termination	2	0	0	2
Total	**2**	**0**	**0**	**2**

J2EE Bad Practices: JVM Termination	**Low**
Package: .com.philips.tasy.agent.client	
client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 106 (J2EE Bad Practices: JVM Termination)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:106
Taint Flags:

J2EE Bad Practices: JVM Termination — Low

Package: .com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 106 (J2EE Bad Practices: JVM Termination) — Low

```
popup.add(creteMenuItem("About", this::showAbout));
popup.add(creteMenuItem("Exit", () -> {
runStop();
System.exit(0);
}));
trayIcon.setPopupMenu(popup);
trayIcon.setImageAutoSize(true);
```

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 76 (J2EE Bad Practices: JVM Termination) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: run()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:76
Taint Flags:

```
} finally {
try {
// when this happens, another instance took over
System.exit(0);
} catch (Exception e) {
LOGGER.warn("exception in exit", e);
}
```

J2EE Bad Practices: Leftover Debug Code (2 issues)

Abstract

Debug code can create unintended entry points in a deployed web application.

Explanation

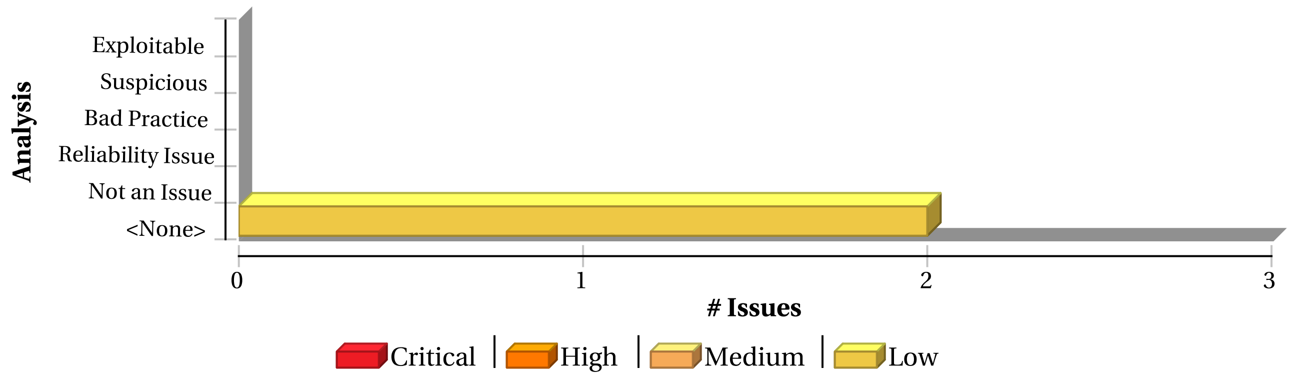
A common development practice is to add "back door" code specifically designed for debugging or testing purposes that is not intended to be shipped or deployed with the application. When this sort of debug code is accidentally left in the application, the application is open to unintended modes of interaction. These back door entry points create security risks because they are not considered during design or testing and fall outside of the expected operating conditions of the application.

The most common example of forgotten debug code is a `main()` method appearing in a web application. Although this is an acceptable practice during product development, classes that are part of a production J2EE application should not define a `main()`.

Recommendation

Remove debug code before deploying a production version of an application. Regardless of whether a direct security threat can be articulated, it is unlikely that there is a legitimate reason for such code to remain in the application after the early stages of development.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Leftover Debug Code	2	0	0	2
Total	**2**	**0**	**0**	**2**

J2EE Bad Practices: Leftover Debug Code	**Low**
Package: com.philips.tasy.agent.client	
client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java, line 34 (J2EE Bad Practices: Leftover Debug Code)	**Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

J2EE Bad Practices: Leftover Debug Code — Low

Package: com.philips.tasy.agent.client

client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java, line 34 (J2EE Bad Practices: Leftover Debug Code) — Low

Sink Details

Sink: Function: main
Enclosing Method: main()
File: client/src/main/java/com/philips/tasy/agent/client/TasyAgentClient.java:34
Taint Flags:

```
31 commands.forEach(jc::addCommand);
32 }
33
34 public static void main(final String... args) throws Exception {
35 //SystemCLICommand.register("tray", TrayCommand.class);
36 Register.register();
37 installExtra();
```

Package: com.philips.tasy.agent.server

server-boot/src/main/java/com.philips.tasy.agent.server/TasyAgentServer.java, line 19 (J2EE Bad Practices: Leftover Debug Code) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: main
Enclosing Method: main()
File: server-boot/src/main/java/com.philips.tasy.agent.server/TasyAgentServer.java:19
Taint Flags:

```
16 @Parameter(names = {"--help", "-h"}, help = true, description = "Display usage info")
17 private boolean help;
18
19 public static void main(final String... args) throws Exception {
20
21 final TasyAgentServer cm = new TasyAgentServer();
22 final JCommander jc = new JCommander(cm);
```

J2EE Bad Practices: Sockets (2 issues)

Abstract

Socket-based communication in web applications is prone to error.

Explanation

The J2EE standard permits the use of sockets only for the purpose of communication with legacy systems when no higher-level protocol is available. Authoring your own communication protocol requires wrestling with difficult security issues, including:

- In-band versus out-of-band signaling

- Compatibility between protocol versions

- Channel security

- Error handling

- Network constraints (firewalls)

- Session management

Without significant scrutiny by a security expert, chances are good that a custom communication protocol will suffer from security problems.

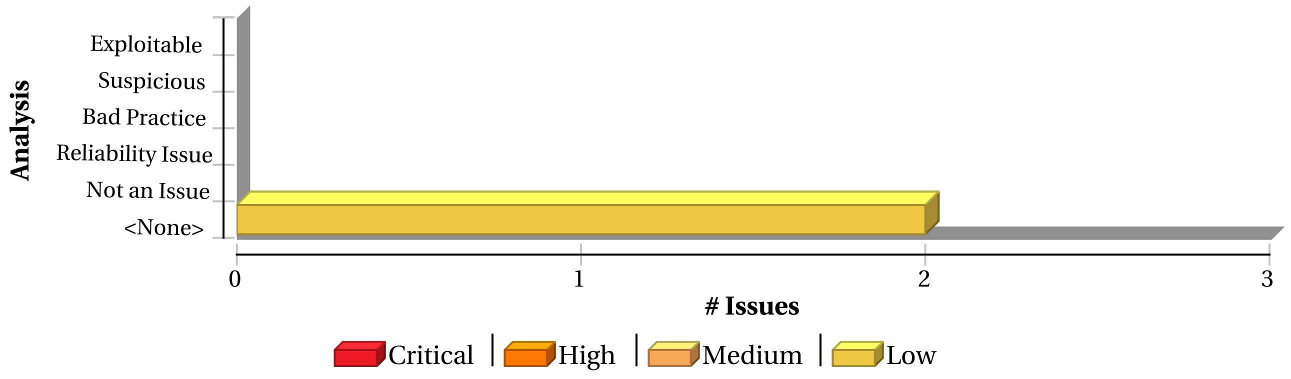
Many of the same issues apply to a custom implementation of a standard protocol. While there are usually more resources available that address security concerns related to implementing a standard protocol, these resources are also available to attackers.

Recommendation

Replace a custom communication protocol with an industry standard protocol or framework. Consider whether you can use a protocol such as HTTP, FTP, SMTP, CORBA, RMI/IIOP, EJB, or SOAP.

Consider the security track record of the protocol implementation you choose.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Sockets	2	0	0	2
Total	**2**	**0**	**0**	**2**

J2EE Bad Practices: Sockets — Low

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 53 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: ServerSocket()
Enclosing Method: startUp()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:53
Taint Flags:

```
50
51  @Override
52  protected void startUp() throws Exception {
53  socket = new ServerSocket(port, 1, InetAddress.getByName(LOCALHOST_ADDRESS));
54  }
55
56  @Override
```

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 35 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: Socket()
Enclosing Method: stopServer()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:35
Taint Flags:

```
32
33  public static void stopServer(final int port) throws IOException {
34  try {
35  final Socket s = new Socket(InetAddress.getByName(LOCALHOST_ADDRESS), port);
36  final OutputStream out = s.getOutputStream();
37  LOGGER.info("*** sending jetty stop request");
38  out.write("STOP\r\n".getBytes(CHARSET_NAME));
```

J2EE Bad Practices: Threads (13 issues)

Abstract

Thread management in a web application is forbidden in some circumstances and is always highly error prone.

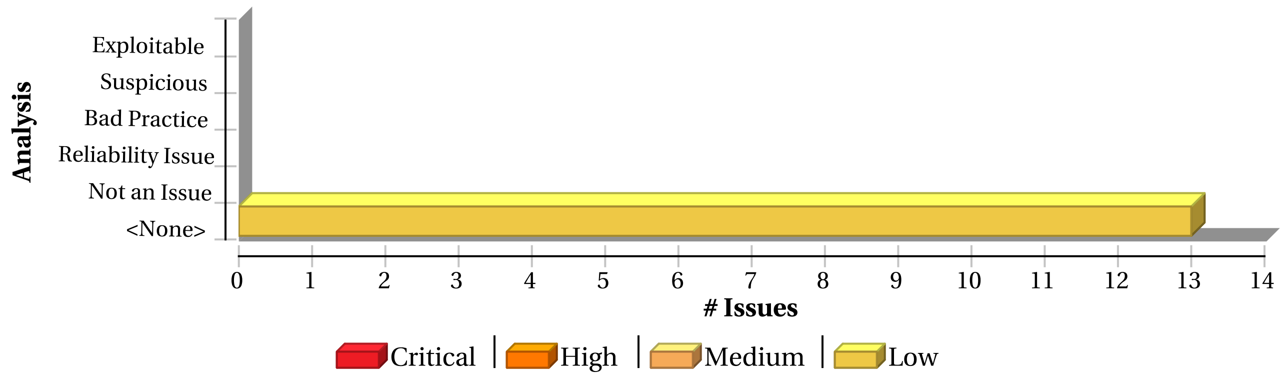
Explanation

Thread management in a web application is forbidden by the J2EE standard in some circumstances and is always highly error prone. Managing threads is difficult and is likely to interfere in unpredictable ways with the behavior of the application container. Even without interfering with the container, thread management usually leads to bugs that are hard to detect and diagnose like deadlock, race conditions, and other synchronization errors.

Recommendation

Avoid managing threads directly from within the web application. Instead use standards such as message driven beans and the EJB timer service that are provided by the application container.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Threads	13	0	0	13
Total	**13**	**0**	**0**	**13**

J2EE Bad Practices: Threads	Low
Package: .com.philips.tasy.agent.client	
client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 152 (J2EE Bad Practices: Threads)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: actionPerformed()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:152

J2EE Bad Practices: Threads — Low

Package: .com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 152 (J2EE Bad Practices: Threads) — Low

Taint Flags:

```
149
150 private MenuItem creteMenuItem(final String description, final Runnable handler) {
151 final MenuItem menuItem = new MenuItem(description);
152 menuItem.addActionListener((e) -> handler.run());
153 return menuItem;
154 }
155
```

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 83 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: forceOtherInstancesStop()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:83
Taint Flags:

```
80 final StopCommand stopCommand = new StopCommand();
81 stopCommand.setConfigFile(this.configFile);
82 stopCommand.run();
83 Thread.sleep(500);
84 } catch (Exception e) {
85 LOGGER.warn("exception stopping other instances at startup", e);
86 }
```

client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java, line 40 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: join()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java:40
Taint Flags:

J2EE Bad Practices: Threads — Low

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java, line 40 (J2EE Bad Practices: Threads) — Low

```
37  t.setContextClassLoader(ConfigScreenCommand.class.getClassLoader());
38  t.start();
39  try {
40  t.join();
41  } catch (InterruptedException e) {
42  LOGGER.warn("run command interrupted", e);
43  }
```

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 146 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: runStart()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:146
Taint Flags:

```
143  if (!this.runCommand.isPresent()) {
144  final RunCommand r = new RunCommand(configFile);
145  this.runCommand = Optional.of(r);
146  r.run();
147  }
148  }
149
```

client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java, line 36 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: Thread()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java:36
Taint Flags:

```
33
34  @Override
35  public void run() {
```

J2EE Bad Practices: Threads — Low

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java, line 36 (J2EE Bad Practices: Threads) — Low

```
final Thread t = new Thread(this::execute);
t.setContextClassLoader(ConfigScreenCommand.class.getClassLoader());
t.start();
try {
```

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 82 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: forceOtherInstancesStop()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:82
Taint Flags:

```
try {
final StopCommand stopCommand = new StopCommand();
stopCommand.setConfigFile(this.configFile);
stopCommand.run();
Thread.sleep(500);
} catch (Exception e) {
LOGGER.warn("exception stopping other instances at startup", e);
```

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 46 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: Thread()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:46
Taint Flags:

```

@Override
public void run() {
final Thread t = new Thread(this::execute);
t.setContextClassLoader(TrayCommand.class.getClassLoader());
t.start();
```

J2EE Bad Practices: Threads — Low

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 46 (J2EE Bad Practices: Threads) — Low

```
}
```

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 128 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: config()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:128
Taint Flags:

```
125 try {
126 final SystemCLICommand config = SystemCLICommand.getCommand("config");
127 config.setConfigFile(configFile);
128 config.run();
129 } catch (IllegalAccessException | InstantiationException e) {
130 LOGGER.error("exception starting config", e);
131 throw new IllegalArgumentException(e);
```

client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java, line 38 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/ConfigScreenCommand.java:38
Taint Flags:

```
35 public void run() {
36 final Thread t = new Thread(this::execute);
37 t.setContextClassLoader(ConfigScreenCommand.class.getClassLoader());
38 t.start();
39 try {
40 t.join();
41 } catch (InterruptedException e) {
```

J2EE Bad Practices: Threads — Low

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 48 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: run()
File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:48
Taint Flags:

```
45 public void run() {
46 final Thread t = new Thread(this::execute);
47 t.setContextClassLoader(TrayCommand.class.getClassLoader());
48 t.start();
49 }
50
51 public void execute() {
```

Package: com.philips.tasy.agent.client.core.commands

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java, line 55 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java:55
Taint Flags:

```
52 public void run() {
53 final Thread t = new Thread(this::execute);
54 t.setContextClassLoader(RunCommand.class.getClassLoader());
55 t.start();
56 if (!daemon) {
57
58 try {
```

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java, line 59 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State

J2EE Bad Practices: Threads — Low

Package: com.philips.tasy.agent.client.core.commands

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java, line 59 (J2EE Bad Practices: Threads) — Low

Scan Engine: SCA (Semantic)

Sink Details

Sink: join()
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java:59
Taint Flags:

```
56 if (!daemon) {
57
58 try {
59 t.join();
60 } catch (InterruptedException e) {
61 LOGGER.warn("Interrupted", e);
62 }
```

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java, line 53 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: Thread()
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java:53
Taint Flags:

```
50
51 @Override
52 public void run() {
53 final Thread t = new Thread(this::execute);
54 t.setContextClassLoader(RunCommand.class.getClassLoader());
55 t.start();
56 if (!daemon) {
```

Missing Check against Null (1 issue)

Abstract

The program can dereference a null pointer because it does not check the return value of a function that might return null.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break.

Two dubious assumptions that are easy to spot in code are "this function call can never fail" and "it doesn't matter if this function call fails". When a programmer ignores the return value from a function, they implicitly state that they are operating under one of these assumptions.

Example 1: The following code does not check to see if the string returned by `getParameter()` is `null` before calling the member function `compareTo()`, potentially causing a null dereference.

```
String itemName = request.getParameter(ITEM_NAME);
   if (itemName.compareTo(IMPORTANT_ITEM)) {
      ...
    }
  ...
```

Example 2:. The following code shows a system property that is set to `null` and later dereferenced by a programmer who mistakenly assumes it will always be defined.

```
System.clearProperty("os.name");
...
String os = System.getProperty("os.name");
if (os.equalsIgnoreCase("Windows 95") )
 System.out.println("Not supported");
```

The traditional defense of this coding error is:

"I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a null value."

But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

Recommendation

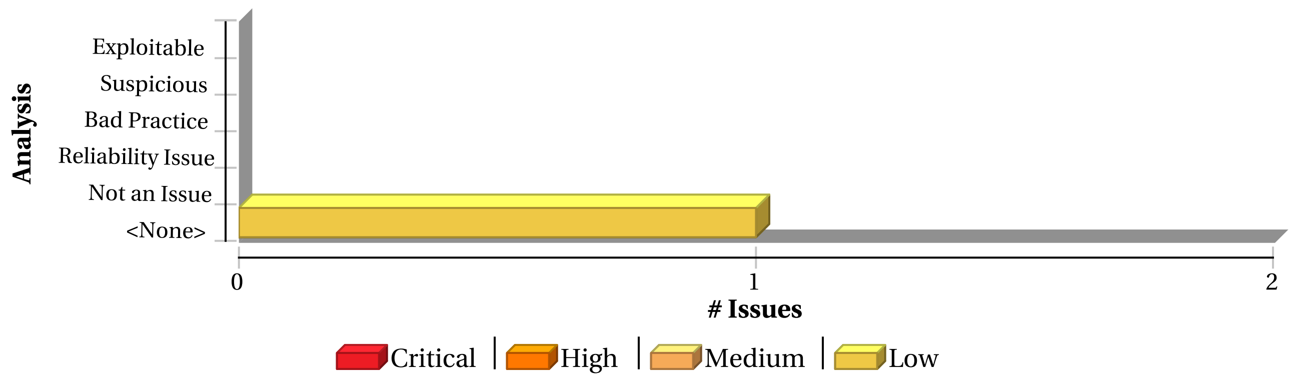
If a function can return an error code or any other evidence of its success or failure, always check for the error condition, even if there is no obvious way for it to occur. In addition to preventing security errors, many initially mysterious bugs have eventually led back to a failed method call with an unchecked return value.

Create an easy to use and standard way for dealing with failure in your application. If error handling is straightforward, programmers will be less inclined to omit it. One approach to standardized error handling is to write wrappers around commonly-used functions that check and handle error conditions without additional programmer intervention. When wrappers are implemented and adopted, the use of non-wrapped equivalents can be prohibited and enforced by using custom rules.

Example 3: The following code implements a wrapper around `getParameter()` that checks the return value of `getParameter()` against `null` and uses a default value if the requested parameter is not defined.

```
String safeGetParameter (HttpRequest request, String name)
{
    String value = request.getParameter(name);
    if (value == null) {
        return getDefaultValue(name)
    }
    return value;
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing Check against Null	1	0	0	1
Total	**1**	**0**	**0**	**1**

Missing Check against Null **Low**

Package: com.philips.tasy.agent.client.core

client-core/src/main/java/com/philips/tasy/agent/client/core/About.java, line 30 (Missing Check against Null) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Control Flow)

Sink Details

Sink: getClassLoader() : Class.getClassLoader may return NULL
Enclosing Method: getVersion()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/About.java:30

Missing Check against Null	Low
Package: com.philips.tasy.agent.client.core	
client-core/src/main/java/com/philips/tasy/agent/client/core/About.java, line 30 (Missing Check against Null)	**Low**

Taint Flags:

```
synchronized (About.class) {
if (version == null) {
try {
final Enumeration<URL> resources = About.class.getClassLoader()
.getResources("META-INF/MANIFEST.MF");

while (resources.hasMoreElements()) {
```

Often Misused: Authentication (2 issues)

Abstract

Attackers may spoof DNS entries. Do not rely on DNS names for security.

Explanation

Many DNS servers are susceptible to spoofing attacks, so you should assume that your software will someday run in an environment with a compromised DNS server. If attackers are allowed to make DNS updates (sometimes called DNS cache poisoning), they can route your network traffic through their machines or make it appear as if their IP addresses are part of your domain. Do not base the security of your system on DNS names.

Example: The following code uses a DNS lookup to determine whether an inbound request is from a trusted host. If an attacker can poison the DNS cache, they can gain trusted status.

```
String ip = request.getRemoteAddr();
InetAddress addr = InetAddress.getByName(ip);
if (addr.getCanonicalHostName().endsWith("trustme.com")) {
trusted = true;
}
```

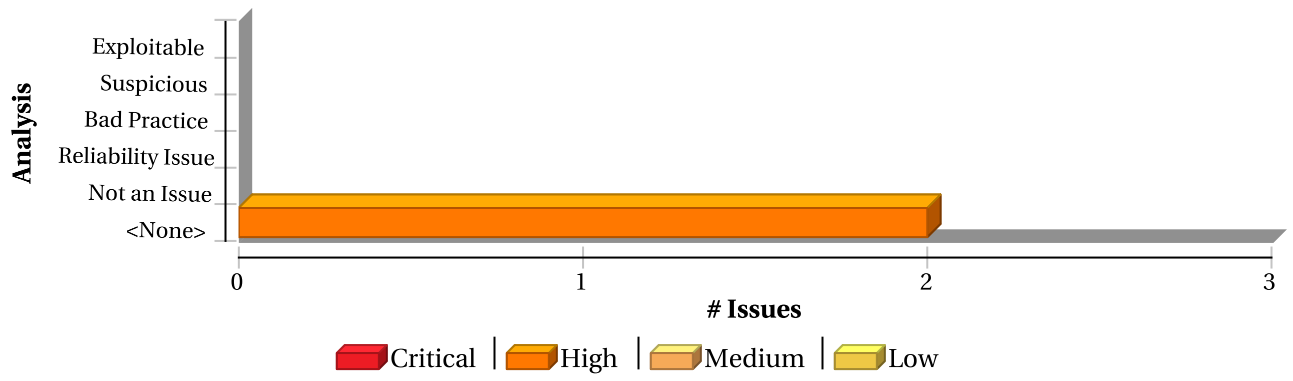
IP addresses are more reliable than DNS names, but they can also be spoofed. Attackers may easily forge the source IP address of the packets they send, but response packets will return to the forged IP address. To see the response packets, the attacker has to sniff the traffic between the victim machine and the forged IP address. In order to accomplish the required sniffing, attackers typically attempt to locate themselves on the same subnet as the victim machine. Attackers may be able to circumvent this requirement by using source routing, but source routing is disabled across much of the Internet today. In summary, IP address verification can be a useful part of an authentication scheme, but it should not be the single factor required for authentication.

Recommendation

You can increase confidence in a domain name lookup if you check to make sure that the host's forward and backward DNS entries match. Attackers will not be able to spoof both the forward and the reverse DNS entries without controlling the nameservers for the target domain. This is not a foolproof approach however: attackers may be able to convince the domain registrar to turn over the domain to a malicious nameserver. Basing authentication on DNS entries is simply a risky proposition.

While no authentication mechanism is foolproof, there are better alternatives than host-based authentication. Password systems offer decent security, but are susceptible to bad password choices, insecure password transmission, and bad password management. A cryptographic scheme like SSL is worth considering, but such schemes are often so complex that they bring with them the risk of significant implementation errors, and key material can always be stolen. In many situations, multi-factor authentication including a physical token offers the most security available at a reasonable price.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: Authentication	2	0	0	2
Total	**2**	**0**	**0**	**2**

Often Misused: Authentication — High

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 35 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByName()
Enclosing Method: stopServer()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:35
Taint Flags:

```
32
33 public static void stopServer(final int port) throws IOException {
34 try {
35 final Socket s = new Socket(InetAddress.getByName(LOCALHOST_ADDRESS), port);
36 final OutputStream out = s.getOutputStream();
37 LOGGER.info("*** sending jetty stop request");
38 out.write("STOP\r\n".getBytes(CHARSET_NAME));
```

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 53 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByName()

Often Misused: Authentication	High

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 53 (Often Misused: Authentication)	High

Enclosing Method: startUp()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:53
Taint Flags:

```

@Override
protected void startUp() throws Exception {
socket = new ServerSocket(port, 1, InetAddress.getByName(LOCALHOST_ADDRESS));
}

@Override
```

Path Manipulation (10 issues)

Abstract

Allowing user input to control paths used in file system operations could enable an attacker to access or modify otherwise protected system resources.

Explanation

Path manipulation errors occur when the following two conditions are met:

1. An attacker is able to specify a path used in an operation on the file system.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program may give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker.

Example 1: The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "`../../tomcat/conf/server.xml`", which causes the application to delete one of its own configuration files.

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2: The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension `.txt`.

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

Some think that in the mobile world, classic vulnerabilities, such as path manipulation, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code adapts Example 1 to the Android platform.

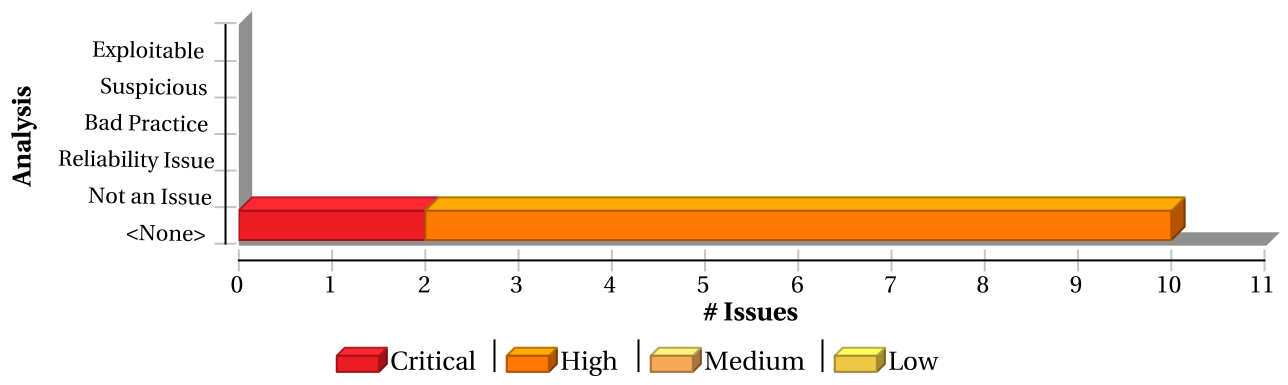
```
...
        String rName = this.getIntent().getExtras().getString("reportName");
        File rFile = getBaseContext().getFileStreamPath(rName);
...
        rFile.delete();
...
```

Recommendation

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to keep track of. Programmers often resort to blacklisting in these situations. Blacklisting selectively rejects or escapes potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a whitelist of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Path Manipulation	10	0	0	10
Total	**10**	**0**	**0**	**10**

Path Manipulation **Critical**

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 33 (Path Manipulation) **Critical**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: javax.naming.InitialContext.lookup()
From: com.philips.tasy.agent.commons.HomeDirLocator.checkJndi
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:31

```
28 result = new FileAndDescription(new File(value.trim()), "JNDI/java:comp/env/" + name);
29 break;
30 }
```

Path Manipulation — Critical

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 33 (Path Manipulation) — Critical

```
31  value = (String) iniCtxt.lookup(name);
32  if (!isStringEmpy(value)) {
33  result = new FileAndDescription(new File(value.trim()), "JNDI/" + name);
34  break;
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: checkJndi()
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:33
Taint Flags: NAMING, XSS

```
30  }
31  value = (String) iniCtxt.lookup(name);
32  if (!isStringEmpy(value)) {
33  result = new FileAndDescription(new File(value.trim()), "JNDI/" + name);
34  break;
35  }
36  } catch (NamingException e) {
```

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 28 (Path Manipulation) — Critical

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: javax.naming.Context.lookup()
From: com.philips.tasy.agent.commons.HomeDirLocator.checkJndi
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:26

```
23  try {
24  final InitialContext iniCtxt = new InitialContext();
25  final Context env = (Context) iniCtxt.lookup("java:comp/env");
26  String value = (String) env.lookup(name);
27  if (!isStringEmpy(value)) {
28  result = new FileAndDescription(new File(value.trim()), "JNDI/java:comp/env/" + name);
29  break;
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: checkJndi()
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:28

Path Manipulation — Critical

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 28 (Path Manipulation) — Critical

Taint Flags: NAMING, XSS

```
25 final Context env = (Context) iniCtxt.lookup("java:comp/env");
26 String value = (String) env.lookup(name);
27 if (!isStringEmpy(value)) {
28 result = new FileAndDescription(new File(value.trim()), "JNDI/java:comp/env/" + name);
29 break;
30 }
31 value = (String) iniCtxt.lookup(name);
```

Path Manipulation — High

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 206 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.adminservices.UpdateService.updateClient
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:206

```
203 });
204 Path updatebat = ClientPaths.updateFolder().toPath().resolve("client").resolve("update.bat");
205 if (Files.isRegularFile(updatebat)) {
206 updatebat = Paths.get(System.getProperty(SysPropertiesNames.AGENT_BASE)).relativize(updatebat);
207 LOGGER.info("Runnning update.bat!");
208 final ProcessBuilder processBuilder = new ProcessBuilder("cmd", "/c", updatebat.toString());
209 processBuilder.environment().put("INSTDIR", System.getProperty(SysPropertiesNames.AGENT_BASE));
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: updateClient()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:206
Taint Flags: PROPERTY

```
203 });
204 Path updatebat = ClientPaths.updateFolder().toPath().resolve("client").resolve("update.bat");
205 if (Files.isRegularFile(updatebat)) {
206 updatebat = Paths.get(System.getProperty(SysPropertiesNames.AGENT_BASE)).relativize(updatebat);
207 LOGGER.info("Runnning update.bat!");
```

Path Manipulation — High

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 206 (Path Manipulation) — High

```
final ProcessBuilder processBuilder = new ProcessBuilder("cmd", "/c", updatebat.toString());
processBuilder.environment().put("INSTDIR", System.getProperty(SysPropertiesNames.AGENT_BASE));
```

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java, line 60 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.core.server.ClientPaths.getHomeDir
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPat
hs.java:51

```
48 if (System.getenv().containsKey(agentHomeEnvVarName)) {
49 result = System.getenv().get(agentHomeEnvVarName);
50 } else if (System.getProperties().containsKey(SysPropertiesNames.AGENT_HOME)) {
51 result = System.getProperty(SysPropertiesNames.AGENT_HOME);
52 } else {
53 result = new File(".").getAbsolutePath();
54 }
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: getConfigFile()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java:60
Taint Flags: PROPERTY

```
57 } else {
58 cfg = new File(configFile);
59 if (!cfg.exists() && !cfg.isAbsolute()) {
60 cfg = new File(ClientPaths.getHomeDir(), configFile);
61 }
62 }
63 return cfg;
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java, line 21 (Path Manipulation) — High

Issue Details

Path Manipulation — High

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java, line 21 (Path Manipulation) — High

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.core.server.ClientPaths.baseTempFolder
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPat
hs.java:59

```
56  }
57
58  public static String baseTempFolder() {
59  return System.getProperty("java.io.tmpdir");
60  }
61 }
62
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: TasyAgentConfiguration()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java:21
Taint Flags: PROPERTY

```
18 public class TasyAgentConfiguration {
19  private static URLClassLoader extraClassPathLoader;
20  @JsonIgnore
21  private String baseTempDir = new File(ClientPaths.baseTempFolder(), "tasy-agent").getAbsolutePath();
22  private String baseDir = new File(ClientPaths.getHomeDir()).getAbsolutePath();
23  private String repositoryUrl;
24  private int stopPort = 8072;
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java, line 22 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.core.server.ClientPaths.getHomeDir
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPat
hs.java:51

Path Manipulation — High

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java, line 22 (Path Manipulation) — **High**

```
if (System.getenv().containsKey(agentHomeEnvVarName)) {
result = System.getenv().get(agentHomeEnvVarName);
} else if (System.getProperties().containsKey(SysPropertiesNames.AGENT_HOME)) {
result = System.getProperty(SysPropertiesNames.AGENT_HOME);
} else {
result = new File(".").getAbsolutePath();
}
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: TasyAgentConfiguration()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/TasyAgentConfiguration.java:22
Taint Flags: PROPERTY

```
private static URLClassLoader extraClassPathLoader;
@JsonIgnore
private String baseTempDir = new File(ClientPaths.baseTempFolder(), "tasy-agent").getAbsolutePath();
private String baseDir = new File(ClientPaths.getHomeDir()).getAbsolutePath();
private String repositoryUrl;
private int stopPort = 8072;
private int listenPort = 4565;
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPaths.java, line 31 (Path Manipulation) — **High**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.core.server.ClientPaths.getHomeDir
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPat
hs.java:51

```
if (System.getenv().containsKey(agentHomeEnvVarName)) {
result = System.getenv().get(agentHomeEnvVarName);
} else if (System.getProperties().containsKey(SysPropertiesNames.AGENT_HOME)) {
result = System.getProperty(SysPropertiesNames.AGENT_HOME);
} else {
result = new File(".").getAbsolutePath();
}
```

Path Manipulation — High

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPaths.java, line 31 (Path Manipulation) — High

Sink Details

Sink: java.io.File.File()
Enclosing Method: updateFolder()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPaths.java:31
Taint Flags: PROPERTY

```
28 }
29
30 public static File updateFolder() {
31 return ensureDir(new File(getHomeDir(), "upd"));
32 }
33
34 private static File ensureDir(final File f) {
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java, line 56 (Path Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.client.core.server.ClientPaths.getHomeDir
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClientPat
hs.java:51

```
48 if (System.getenv().containsKey(agentHomeEnvVarName)) {
49 result = System.getenv().get(agentHomeEnvVarName);
50 } else if (System.getProperties().containsKey(SysPropertiesNames.AGENT_HOME)) {
51 result = System.getProperty(SysPropertiesNames.AGENT_HOME);
52 } else {
53 result = new File(".").getAbsolutePath();
54 }
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: getConfigFile()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java:56
Taint Flags: PROPERTY

```
53 private static File getConfigFile() {
54 File cfg;
55 if (configFile == null) {
56 cfg = new File(ClientPaths.getHomeDir(), "config.yml");
```

Path Manipulation High

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ConfigurationProvider.java, line 56 (Path Manipulation) High

```
} else {
cfg = new File(configFile);
if (!cfg.exists() && !cfg.isAbsolute()) {
```

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 85 (Path Manipulation) High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.commons.HomeDirLocator.fromUserHome
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:85

```
}

private static FileAndDescription fromUserHome() {
final File legacyHome = new File(new File(System.getProperty("user.home")), ".agent");
return new FileAndDescription(legacyHome, "$user.home/.agent");
}

```

Sink Details

Sink: java.io.File.File()
Enclosing Method: fromUserHome()
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:85
Taint Flags: PROPERTY

```
}

private static FileAndDescription fromUserHome() {
final File legacyHome = new File(new File(System.getProperty("user.home")), ".agent");
return new FileAndDescription(legacyHome, "$user.home/.agent");
}

```

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 48 (Path Manipulation) High

Issue Details

Path Manipulation — High

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 48 (Path Manipulation) — **High**

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.System.getProperty()
From: com.philips.tasy.agent.commons.HomeDirLocator.checkSysProps
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:46

```
43 private static Optional<FileAndDescription> checkSysProps() {
44 FileAndDescription result = null;
45 for (String name : HOME_NAMES) {
46 final String sysProp = System.getProperty(name);
47 if (sysProp != null) {
48 result = new FileAndDescription(new File(sysProp.trim()),
49 String.format("System.getProperty(\"%s\")", name));
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: checkSysProps()
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:48
Taint Flags: PROPERTY

```
45 for (String name : HOME_NAMES) {
46 final String sysProp = System.getProperty(name);
47 if (sysProp != null) {
48 result = new FileAndDescription(new File(sysProp.trim()),
49 String.format("System.getProperty(\"%s\")", name));
50 break;
51 }
```

Poor Error Handling: Empty Catch Block (1 issue)

Abstract

Ignoring an exception can cause the program to overlook unexpected states and conditions.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break.

Two dubious assumptions that are easy to spot in code are "this method call can never fail" and "it doesn't matter if this call fails". When a programmer ignores an exception, they implicitly state that they are operating under one of these assumptions.

Example 1: The following code excerpt ignores a rarely-thrown exception from `doExchange()`.

```
try {
  doExchange();
}
catch (RareException e) {
  // this can never happen
}
```

If a `RareException` were to ever be thrown, the program would continue to execute as though nothing unusual had occurred. The program records no evidence indicating the special situation, potentially frustrating any later attempt to explain the program's behavior.

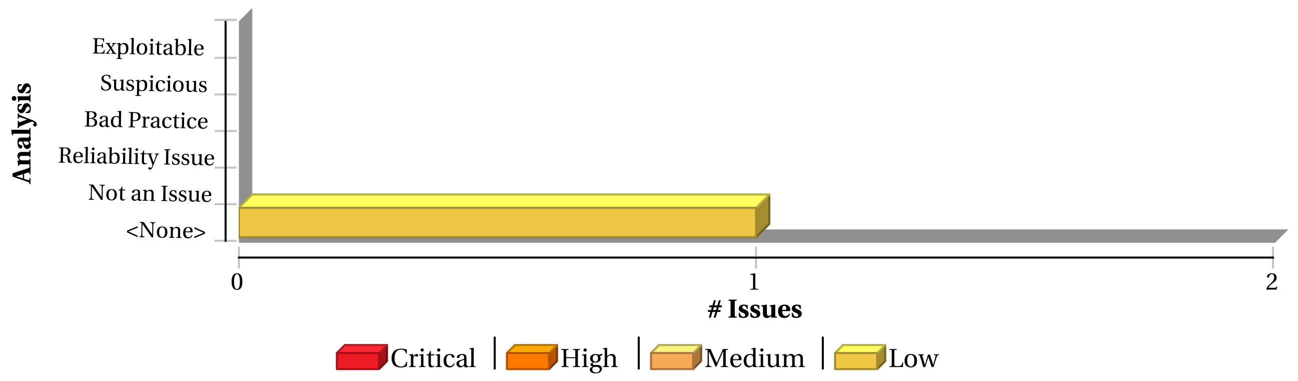
Recommendation

At a minimum, log the fact that the exception was thrown so that it will be possible to come back later and make sense of the resulting program behavior. Better yet, abort the current operation. If the exception is being ignored because the caller cannot properly handle it but the context makes it inconvenient or impossible for the caller to declare that it throws the exception itself, consider throwing a `RuntimeException` or an `Error`, both of which are unchecked exceptions. As of JDK 1.4, `RuntimeException` has a constructor that makes it easy to wrap another exception.

Example 2: The code in Example 1 could be rewritten in the following way:

```
try {
  doExchange();
}
catch (RareException e) {
  throw new RuntimeException("This can never happen", e);
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Empty Catch Block	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Error Handling: Empty Catch Block — Low

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java, line 36 (Poor Error Handling: Empty Catch Block) — **Low**

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: checkJndi()
File: commons/src/main/java/com/philips/tasy/agent/commons/HomeDirLocator.java:36
Taint Flags:

```
33 result = new FileAndDescription(new File(value.trim()), "JNDI/" + name);
34 break;
35 }
36 } catch (NamingException e) {
37 // ignore
38 }
39 }
```

Poor Error Handling: Overly Broad Catch (13 issues)

Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

Explanation

Multiple catch blocks can get ugly and repetitive, but "condensing" catch blocks by catching a high-level class like `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.

Example: The following code excerpt handles three types of exceptions in an identical fashion.

```
try {
  doExchange();
}
catch (IOException e) {
  logger.error("doExchange failed", e);
}
catch (InvocationTargetException e) {
  logger.error("doExchange failed", e);
}
catch (SQLException e) {
  logger.error("doExchange failed", e);
}
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

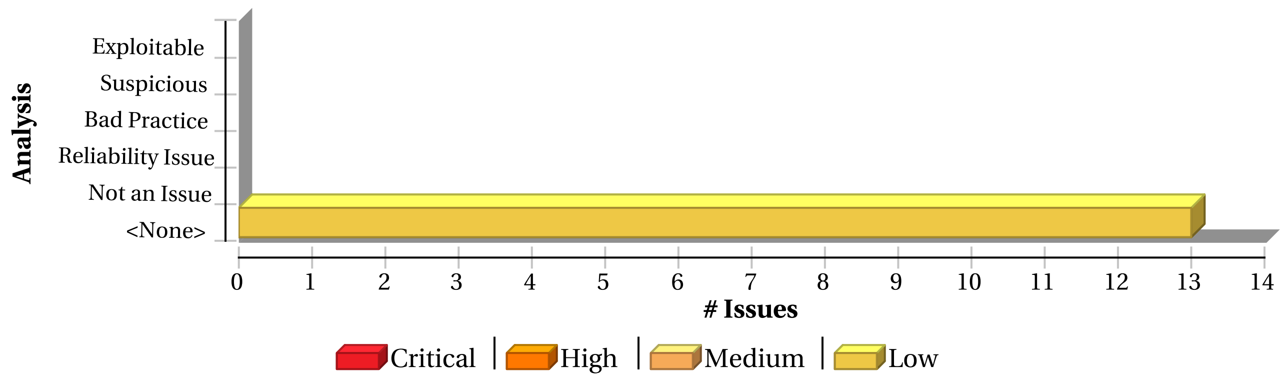
```
try {
  doExchange();
}
catch (Exception e) {
  logger.error("doExchange failed", e);
}
```

However, if `doExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from `RuntimeException` such as `ClassCastException`, and `NullPointerException`, which is not the programmer's intent.

Recommendation

Do not catch broad exception classes like `Exception`, `Throwable`, `Error`, or except at the very top level of the program or thread.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Catch	13	0	0	13
Total	**13**	**0**	**0**	**13**

Poor Error Handling: Overly Broad Catch — Low

Package: .com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 242 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: accept()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:242
Taint Flags:

```
239 .invoke();
240 }
241 }
242 } catch (Exception e) {
243 LOGGER.error("Exception updating module " + module.getNewApp().getModuleName(), e);
244 }
245
```

Package: .com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 90 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Poor Error Handling: Overly Broad Catch — Low

Package: .com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 90 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: accept()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java:90
Taint Flags:

```
87 .forEach(wa -> {
88 try {
89 deploy(wa);
90 } catch (Exception e) {
91 LOGGER.error("Exception deploing inner modules " + wa.getContextPath(), e);
92 }
93 });
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 108 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: accept()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java:108
Taint Flags:

```
105 .forEach((webAppContext) -> {
106 try {
107 deploy(webAppContext);
108 } catch (Exception e) {
109 LOGGER.error("Exception deploying configured service to module "
110 + webAppContext.getContextPath(), e);
111 }
```

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 84 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: forceOtherInstancesStop()

Poor Error Handling: Overly Broad Catch — Low

Package: com.philips.tasy.agent.client

client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java, line 84 (Poor Error Handling: Overly Broad Catch) — Low

File: client-services/src/main/java/com/philips/tasy/agent/client/TrayCommand.java:84
Taint Flags:

```
81 stopCommand.setConfigFile(this.configFile);
82 stopCommand.run();
83 Thread.sleep(500);
84 } catch (Exception e) {
85 LOGGER.warn("exception stopping other instances at startup", e);
86 }
87 }
```

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 106 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: updateToLatest()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:106
Taint Flags:

```
103 .request()
104 .post(Entity.entity(objectMapper.writeValueAsString(apps), MediaType.APPLICATION_JSON_TYPE));
105 processUpdateResponse(response);
106 } catch (Exception e) {
107 LOGGER.warn(EXCEPTION_ON_UPDATER_SERVICE_WILL_RETRY_SOON, e);
108 }
109 }
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 120 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: updateApp()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:120
Taint Flags:

Poor Error Handling: Overly Broad Catch — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 120 (Poor Error Handling: Overly Broad Catch) — Low

```
.request()
.post(Entity.entity(objectMapper.writeValueAsString(apps), MediaType.APPLICATION_JSON_TYPE));
processUpdateResponse(response);
} catch (Exception e) {
LOGGER.warn(EXCEPTION_ON_UPDATER_SERVICE_WILL_RETRY_SOON, e);
}
}
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 140 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: checkForUpdates()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:140
Taint Flags:

```
final String s = response.readEntity(String.class);
result = objectMapper.readValue(s, UpdateResult.class);
}
} catch (Exception e) {
LOGGER.warn(EXCEPTION_ON_UPDATER_SERVICE_WILL_RETRY_SOON, e);
}
return result;
```

Package: com.philips.tasy.agent.client.core.commands

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunSingleCommand.java, line 50 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: afterStart()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunSingleCommand.java:50
Taint Flags:

```
.get()
.invoke();
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.philips.tasy.agent.client.core.commands

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunSingleCommand.java, line 50 (Poor Error Handling: Overly Broad Catch) — Low

```
49
50  } catch (Exception e) {
51  LOGGER.error("Exception deploing single module", e);
52  }
53  }
```

client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java, line 102 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: execute()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/commands/RunCommand.java:102
Taint Flags:

```
99  if (!daemon) {
100  sm.awaitStopped();
101  }
102  } catch (Exception e) {
103  LOGGER.error("Exception on application bootstrap", e);
104  }
105  }
```

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java, line 32 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: classForName()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java:32
Taint Flags:

```
29  if (contextClassLoader != null) {
30  clzz = contextClassLoader.loadClass(name);
31  }
32  } catch (Throwable e) {
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ClassLoaderCleaner.java, line 32 (Poor Error Handling: Overly Broad Catch) — Low

```
LOGGER.debug("classForName Exception", e);
}
return clzz != null ? clzz : Class.forName(name);
```

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/Hash.java, line 50 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: checksum()
File: commons/src/main/java/com/philips/tasy/agent/commons/Hash.java:50
Taint Flags:

```
47 digest.update(block, 0, length);
48 }
49 result = digest.digest();
50 } catch (Exception e) {
51 LOGGER.warn("Exception digesting file", e);
52 }
53 return result;
```

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: run()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:71
Taint Flags:

```
68 server.stop();
69 accept.close();
70 socket.close();
71 } catch (Error e) {
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 71 (Poor Error Handling: Overly Broad Catch) — Low

```
LOGGER.warn("error", e);
} finally {
try {
```

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 77 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: run()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:77
Taint Flags:

```
try {
// when this happens, another instance took over
System.exit(0);
} catch (Exception e) {
LOGGER.warn("exception in exit", e);
}
}
```

Poor Error Handling: Overly Broad Throws (27 issues)

Abstract

The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.

Explanation

Declaring a method to throw `Exception` or `Throwable` makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system.

Example: The following method throws three types of exceptions.

```
public void doExchange()
  throws IOException, InvocationTargetException,
         SQLException {
  ...
}
```

While it might seem tidier to write

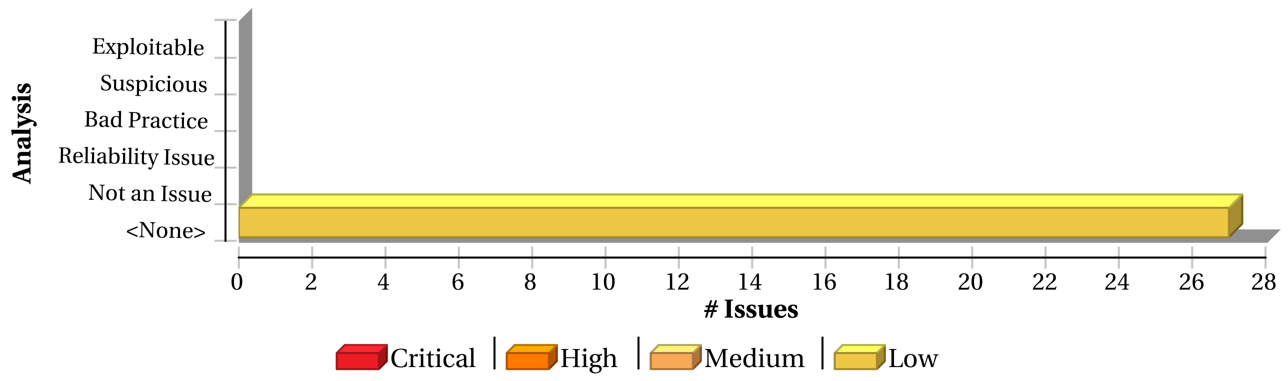
```
public void doExchange()
  throws Exception {
  ...
}
```

doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of `doExchange()` introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.

Recommendation

Do not declare methods to throw `Exception` or `Throwable`. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, consider throwing unchecked exceptions rather than checked exceptions. This can be accomplished by implementing exception classes that extend `RuntimeException` or `Error` instead of `Exception`, or add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Throws	27	0	0	27
Total	**27**	**0**	**0**	**27**

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.admin.services

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 50 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: uninstallPackage
Enclosing Method: uninstallPackage()
File: admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java:50
Taint Flags:

```
47 @POST
48 @Path("uninstall")
49 @Produces(MediaType.TEXT_PLAIN)
50 public Boolean uninstallPackage(@QueryParam("package") final String name) throws Exception {
51 return requireService.uninstallPackage(name);
52 }
53
```

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hasPackage

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.admin.services

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

Enclosing Method: hasPackage()
File: admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java:28
Taint Flags:

```
@GET
@Path("has")
@Produces(MediaType.TEXT_PLAIN)
public Boolean hasPackage(@QueryParam("package") final String name) throws Exception {
return requireService.hasPackage(name);
}

```

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 35 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: installPackage
Enclosing Method: installPackage()
File: admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java:35
Taint Flags:

```
@POST
@Path("install/{module}")
@Produces(MediaType.TEXT_PLAIN)
public void installPackage(@PathParam("module") final String name,
@QueryParam("v") final String version) throws Exception {
requireService.installPackage(name, version);
}
```

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: listInstalledPackages
Enclosing Method: listInstalledPackages()
File: admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java:57
Taint Flags:

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.admin.services

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

```
54  @GET
55  @Path("list")
56  @Produces(MediaType.APPLICATION_JSON)
57  public List<Object> listInstalledPackages() throws Exception {
58  return requireService.listInstalledPackages();
59  }
60 }
```

admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java, line 43 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: installPackage
Enclosing Method: installPackage()
File: admin-console/src/main/java/com/philips/tasy/agent/admin/services/RequireServiceImpl.java:43
Taint Flags:

```
40  @POST
41  @Path("install/id:{module}")
42  @Produces(MediaType.TEXT_PLAIN)
43  public void installPackage(@PathParam("module") final String moduleId) throws Exception {
44  requireService.installPackage(moduleId);
45  }
46
```

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 32 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: installPackage
Enclosing Method: installPackage()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:32
Taint Flags:

```
29  return AgentCommand.has().moduleName(name).get().invoke();
30  }
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 32 (Poor Error Handling: Overly Broad Throws) — Low

```
public void installPackage(final String name, final String version) throws Exception {
final Response response = ClientBuilder.newClient()
.target(configuration.getRepositoryUrl())
.path(configuration.getRepositoryContextPath())
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: processResponse
Enclosing Method: processResponse()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:57
Taint Flags:

```
processResponse(response);
}

private void processResponse(final Response response) throws Exception {
if (response.getStatus() == Response.Status.NOT_FOUND.getStatusCode()) {
throw new NotFoundException();
}
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 88 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: listInstalledPackages
Enclosing Method: listInstalledPackages()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:88
Taint Flags:

```
return result;
}

public List<Object> listInstalledPackages() throws Exception {
return AgentCommand.list().invoke();
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 88 (Poor Error Handling: Overly Broad Throws) — Low

```
90 }
91
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 45 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: installPackage
Enclosing Method: installPackage()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:45
Taint Flags:

```
42 processResponse(response);
43 }
44
45 public void installPackage(final String moduleId) throws Exception {
46 final Response response = ClientBuilder.newClient()
47 .target(configuration.getRepositoryUrl())
48 .path(configuration.getRepositoryContextPath())
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 79 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: uninstallPackage
Enclosing Method: uninstallPackage()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:79
Taint Flags:

```
76 .invoke();
77 }
78
79 public Boolean uninstallPackage(final String name) throws Exception {
80 boolean result = false;
81 if (hasPackage(name)) {
82 AgentCommand.stop().moduleName(name).get().invoke();
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateServiceImpl.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: runOneIteration
Enclosing Method: runOneIteration()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateServiceImpl.java:72
Taint Flags:

```
69 }
70
71 @Override
72 protected void runOneIteration() throws Exception {
73 updateService.updateToLatest();
74 }
75
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java, line 28 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hasPackage
Enclosing Method: hasPackage()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/RequireService.java:28
Taint Flags:

```
25 this.configuration = configuration;
26 }
27
28 public Boolean hasPackage(final String name) throws Exception {
29 return AgentCommand.has().moduleName(name).get().invoke();
30 }
31
```

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateServiceImpl.java, line 67 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateServiceImpl.java, line 67 (Poor Error Handling: Overly Broad Throws) — Low

Sink Details

Sink: Function: startUp
Enclosing Method: startUp()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateServiceImpl.java:67
Taint Flags:

```
64 }
65
66 @Override
67 protected void startUp() throws Exception {
68 updateService.startUp();
69 }
70
```

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 119 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shutDown
Enclosing Method: shutDown()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java:119
Taint Flags:

```
116 * signal the web container to stop.
117 */
118 @Override
119 protected void shutDown() throws Exception {
120 webServer.shutDown();
121 AgentCommandBuilder.setContainer(null);
122
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: startUp

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java, line 57 (Poor Error Handling: Overly Broad Throws) — Low

Enclosing Method: startUp()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/OnlineApplicationContainer.java:57
Taint Flags:

```
54  * @see Server#start()
55  */
56  @Override
57  protected void startUp() throws Exception {
58  clearTempDir();
59  webServer.setup(debug);
60  deployInnerModules();
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ApplicationContainer.java, line 12 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: deploy
Enclosing Method: deploy()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ApplicationContainer.java:12
Taint Flags:

```
9  * Created by vhkrausser on 04/08/2016.
10  */
11 public interface ApplicationContainer {
12  void deploy(final WebAppContext webAppContext) throws Exception;
13
14  void undeploy(final WebAppContext webAppContext) throws Exception;
15
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/AgentCommand.java, line 29 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: invoke
Enclosing Method: invoke()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/AgentCommand.java:29
Taint Flags:

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/AgentCommand.java, line 29 (Poor Error Handling: Overly Broad Throws) — Low

```
 }


 T invoke() throws Exception;
}

undefined
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java, line 138 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shutDown
Enclosing Method: shutDown()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java:138
Taint Flags:

```
/**
 * signal the web container to stop.
 */
protected void shutDown() throws Exception {
Preconditions.checkState(server != null, "Server not started");
for (Handler h : contexts.getHandlers()) {
h.stop();
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java, line 41 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: setup
Enclosing Method: setup()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java:41
Taint Flags:

```
 *
 * @see Server#start()
 */
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java, line 41 (Poor Error Handling: Overly Broad Throws) — Low

```
41  protected void setup(final boolean debug) throws Exception {
42  LOGGER.info("WEBSERVER - Starting Up");
43  Preconditions.checkState(server == null, "Server already started");
44  System.setProperty("jetty.home", configuration.getBaseDir());
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/ApplicationContainer.java, line 14 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: undeploy
Enclosing Method: undeploy()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/ApplicationContainer.java:14
Taint Flags:

```
11 public interface ApplicationContainer {
12  void deploy(final WebAppContext webAppContext) throws Exception;
13
14  void undeploy(final WebAppContext webAppContext) throws Exception;
15
16  List<String> getInstalledModules();
17
```

client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java, line 80 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: start
Enclosing Method: start()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java:80
Taint Flags:

```
77  server.start();
78  }
79
80  protected void start() throws Exception {
81
82  this.contexts.start();
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/WebServer.java, line 80 (Poor Error Handling: Overly Broad Throws) — **Low**

```
83  LOGGER.info("WEBSERVER - UP");
```

Package: com.philips.tasy.agent.commons

commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java, line 7 (Poor Error Handling: Overly Broad Throws) — **Low**

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: compareTo
Enclosing Method: compareTo()
File: commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java:7
Taint Flags:

```
4
5 public class VersionTest {
6  @org.junit.Test
7  public void compareTo() throws Exception {
8  Version minor = new Version(1,0,0);
9  Version major = new Version(2,0,0);
10  assertEquals(-1, minor.compareTo(major) );
```

commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java, line 20 (Poor Error Handling: Overly Broad Throws) — **Low**

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: compareTo3
Enclosing Method: compareTo3()
File: commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java:20
Taint Flags:

```
17  }
18
19  @org.junit.Test
20  public void compareTo3() throws Exception {
21  Version minor = new Version(2,0,0);
22  Version major = new Version(1,0,0);
23  assertEquals(1,minor.compareTo(major) );
```

Poor Error Handling: Overly Broad Throws

Low

Package: com.philips.tasy.agent.commons

commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java, line 13 (Poor Error Handling: Overly Broad Throws)

Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: compareTo2
Enclosing Method: compareTo2()
File: commons/src/test/java/com/philips/tasy/agent/commons/VersionTest.java:13
Taint Flags:

```
10  assertEquals(-1, minor.compareTo(major) );
11  }
12  @org.junit.Test
13  public void compareTo2() throws Exception {
14  Version minor = new Version(1,0,0);
15  Version major = new Version(1,0,0);
16  assertEquals(0, minor.compareTo(major));
```

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 52 (Poor Error Handling: Overly Broad Throws)

Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: startUp
Enclosing Method: startUp()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:52
Taint Flags:

```
49  }
50
51  @Override
52  protected void startUp() throws Exception {
53  socket = new ServerSocket(port, 1, InetAddress.getByName(LOCALHOST_ADDRESS));
54  }
55
```

Package: com.philips.tasy.agent.server.http

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 120 (Poor Error Handling: Overly Broad Throws)

Low

Issue Details

Poor Error Handling: Overly Broad Throws — Low

Package: com.philips.tasy.agent.server.http

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 120 (Poor Error Handling: Overly Broad Throws) — Low

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: shutDown
Enclosing Method: shutDown()
File: server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java:120
Taint Flags:

```
117 * signal the web server to stop.
118 */
119 @Override
120 public void shutDown() throws Exception {
121 Preconditions.checkState(server != null, "Server not started");
122 server.stop();
123 }
```

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 47 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: startUp
Enclosing Method: startUp()
File: server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java:47
Taint Flags:

```
44 * @see Server#start()
45 */
46 @Override
47 public void startUp() throws Exception {
48 Preconditions.checkState(server == null, "Server already started");
49
50 // Create the server.
```

Poor Error Handling: Throw Inside Finally (1 issue)

Abstract

Using a `throw` statement inside a `finally` block breaks the logical progression through the `try-catch-finally`.

Explanation

In Java, `finally` blocks are always executed after their corresponding `try-catch` blocks and are often used to free allocated resources, such as file handles or database cursors. Throwing an exception in a `finally` block can bypass critical cleanup code since normal program execution will be disrupted.

Example 1: In the following code, the call to `stmt.close()` is bypassed when the `FileNotFoundException` is thrown.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis = null;
    Statement stmt = null;
    try
    {
        stmt = conn.createStatement();
        fis = new FileInputStream("badFile.txt");
        ...
    }
    catch (FileNotFoundException fe)
    {
        log("File not found.");
    }
    catch (SQLException se)
    {
        //handle error
    }
    finally
    {
        if (fis == null)
        {
            throw new FileNotFoundException();
        }

        if (stmt != null)
        {
            try
            {
                stmt.close();
            }
            catch (SQLException e)
            {
                log(e);
            }
        }
    }
}
```

This category is from the Cigital Java Rulepack. http://www.cigital.com/

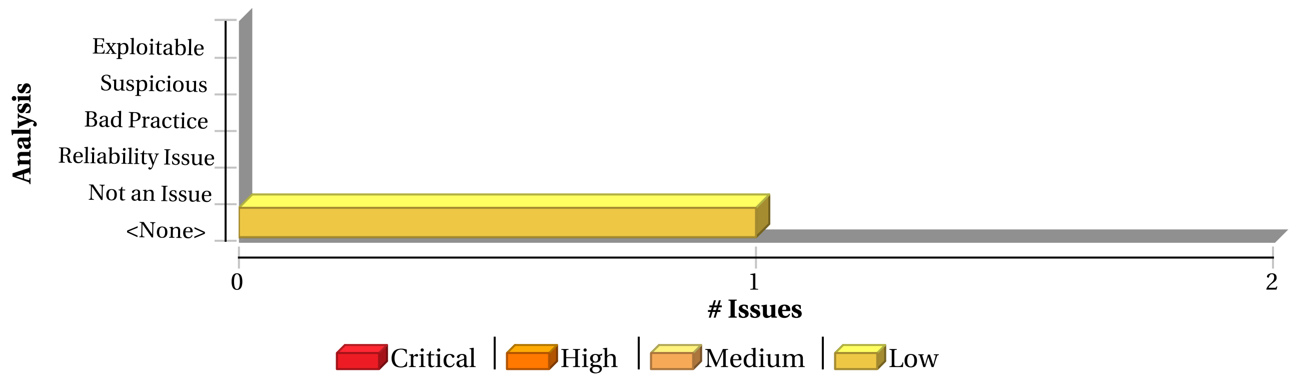
Recommendation

Never throw exceptions from within `finally` blocks. If you must re-throw an exception, do it inside a `catch` block so as not to interrupt the normal execution of the `finally` block.

Example 2: The following code re-throws the `FileNotFoundException` in the `catch` block.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis = null;
    Statement stmt = null;
    try
    {
        stmt = conn.createStatement();
        fis = new FileInputStream("badFile.txt");
        ...
    }
    catch (FileNotFoundException fe)
    {
        log("File not found.");
        throw fe;
    }
    catch (SQLException se)
    {
        //handle error
    }
    finally
    {
        if (fis != null)
        {
            try
            {
                fis.close();
            }
            catch (IOException ie)
            {
                log(ie);
            }
        }

        if (stmt != null)
        {
            try
            {
                stmt.close();
            }
            catch (SQLException e)
            {
                log(e);
            }
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Throw Inside Finally	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Error Handling: Throw Inside Finally — Low

Package: com.philips.tasy.agent.client.adminservices

client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java, line 176 (Poor Error Handling: Throw Inside Finally) — **Low**

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: FinallyBlock
Enclosing Method: executeUpdateWithStream()
File: client-services/src/main/java/com/philips/tasy/agent/client/adminservices/UpdateService.java:176
Taint Flags:

```
173 }
174
175 }
176 }
177
178 private void updateClient(final FileSystem zipFileSystem) throws IOException {
179
```

Poor Logging Practice: Use of a System Output Stream (5 issues)

Abstract

Using `System.out` or `System.err` rather than a dedicated logging facility makes it difficult to monitor the behavior of the program.

Explanation

Example 1: The first Java program that a developer learns to write often looks like this:

```
public class MyClass
  public static void main(String[] args) {
    System.out.println("hello world");
  }
}
```

While most programmers go on to learn many nuances and subtleties about Java, a surprising number hang on to this first lesson and never give up on writing messages to standard output using `System.out.println()`.

The problem is that writing directly to standard output or standard error is often used as an unstructured form of logging. Structured logging facilities provide features like logging levels, uniform formatting, a logger identifier, timestamps, and, perhaps most critically, the ability to direct the log messages to the right place. When the use of system output streams is jumbled together with the code that uses loggers properly, the result is often a well-kept log that is missing critical information.

Developers widely accept the need for structured logging, but many continue to use system output streams in their "pre-production" development. If the code you are reviewing is past the initial phases of development, use of `System.out` or `System.err` may indicate an oversight in the move to a structured logging system.

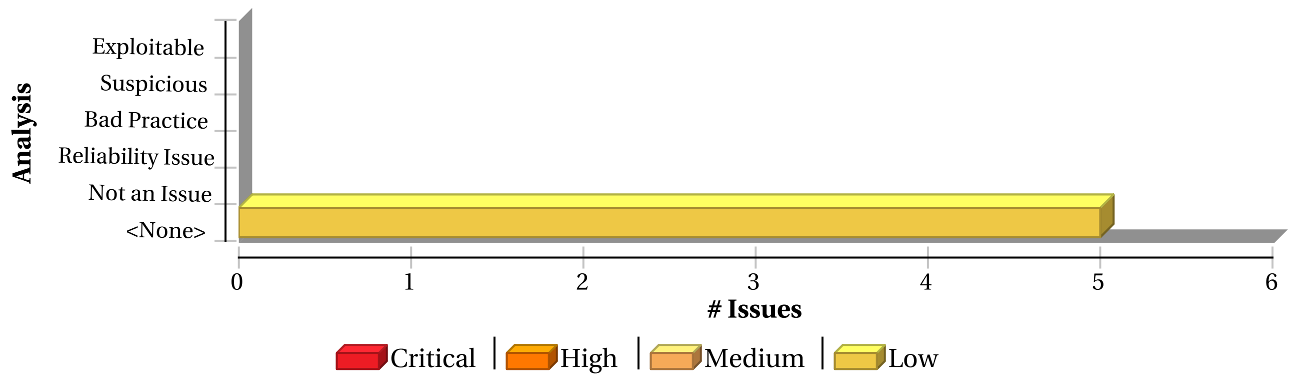
Recommendation

Use a Java logging facility rather than `System.out` or `System.err`.

Example 2: For example, the "hello world" program above can be re-written using log4j like this:

```
import org.apache.log4j.Logger;
import org.apache.log4j.BasicConfigurator;

public class MyClass {
  private final static Logger logger =
            Logger.getLogger(MyClass.class);

  public static void main(String[] args) {
    BasicConfigurator.configure();
    logger.info("hello world");
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Logging Practice: Use of a System Output Stream	5	0	0	5
Total	**5**	**0**	**0**	**5**

Poor Logging Practice: Use of a System Output Stream — Low

Package: com.philips.tasy.agent.server.commands

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 46 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: read()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java:46
Taint Flags:

```
43 System.out.print(question);
44 System.out.print(" [default=");
45 System.out.print(def);
46 System.out.println("]: ");
47 String ret = scanner.nextLine();
48 if (Strings.isNullOrEmpty(ret)) {
49 ret = def;
```

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 44 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print

Poor Logging Practice: Use of a System Output Stream — Low

Package: com.philips.tasy.agent.server.commands

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 44 (Poor Logging Practice: Use of a System Output Stream) — Low

Enclosing Method: read()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java:44
Taint Flags:

```
41 @SuppressWarnings("PMD.SystemPrintln")
42 public String read(final String question, final String def) {
43 System.out.print(question);
44 System.out.print(" [default=");
45 System.out.print(def);
46 System.out.println("]: ");
47 String ret = scanner.nextLine();
```

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 43 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: read()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java:43
Taint Flags:

```
40
41 @SuppressWarnings("PMD.SystemPrintln")
42 public String read(final String question, final String def) {
43 System.out.print(question);
44 System.out.print(" [default=");
45 System.out.print(def);
46 System.out.println("]: ");
```

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 45 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: read()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java:45
Taint Flags:

Poor Logging Practice: Use of a System Output Stream — Low

Package: com.philips.tasy.agent.server.commands

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 45 (Poor Logging Practice: Use of a System Output Stream) — Low

```
public String read(final String question, final String def) {
System.out.print(question);
System.out.print(" [default=");
System.out.print(def);
System.out.println("]: ");
String ret = scanner.nextLine();
if (Strings.isNullOrEmpty(ret)) {
```

server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java, line 56 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: readPassword()
File: server-boot/src/main/java/com.philips.tasy.agent.server/commands/ConfigureCommand.java:56
Taint Flags:

```

@SuppressWarnings("PMD.SystemPrintln")
public Password readPassword(final String msg, final String realm) {
System.out.print(msg);
return Password.getPassword(realm, null, null);
 }
}
```

Race Condition: Singleton Member Field (1 issue)

Abstract

Servlet member fields might allow one user to see another user's data.

Explanation

Many Servlet developers do not understand that a Servlet is a singleton. There is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads.

A common result of this misunderstanding is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.

Example 1: The following Servlet stores the value of a request parameter in a member field and then later echoes the parameter value to the response output stream.

```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     name = req.getParameter("name");
     ...
     out.println(name + ", thanks for visiting!");
   }
}
```

While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way:

Thread 1: assign "`Dick`" to `name`
Thread 2: assign "`Jane`" to `name`
Thread 1: print "`Jane, thanks for visiting!`"
Thread 2: print "`Jane, thanks for visiting!`"

Thereby showing the first user the second user's name.

Recommendation

Do not use Servlet member fields for anything but constants. (i.e. make all member fields `static final`).

Developers are often tempted to use Servlet member fields for user data when they need to transport data from one region of code to another. If this is your aim, consider declaring a separate class and using the Servlet only to "wrap" this new class.

Example 2: The bug in the example above can be corrected in the following way:

```
public class GuestBook extends HttpServlet {
   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
 GBRequestHandler handler = new GBRequestHandler();
 handler.handle(req, res);
   }
}

public class GBRequestHandler {

   String name;

   public void handle(HttpServletRequest req, HttpServletResponse res) {
     name = req.getParameter("name");
     ...
     out.println(name + ", thanks for visiting!");
   }

}
```

Alternatively, a Servlet can utilize synchronized blocks to access servlet instance variables but using synchronized blocks may cause significant performance problems.

Please notice that wrapping the field access within a synchronized block will only prevent the issue if all read and write operations on that member are performed within the same synchronized block or method.

Example 3: Wrapping the Example 1 write operation (assignment) in a synchronized block will not fix the problem since the threads will have to get a lock to modify `name` field, but they will release the lock afterwards, allowing a second thread to change the value again. If, after changing the `name` value, the first thread resumes execution, the value printed will be the one assigned by the second thread:

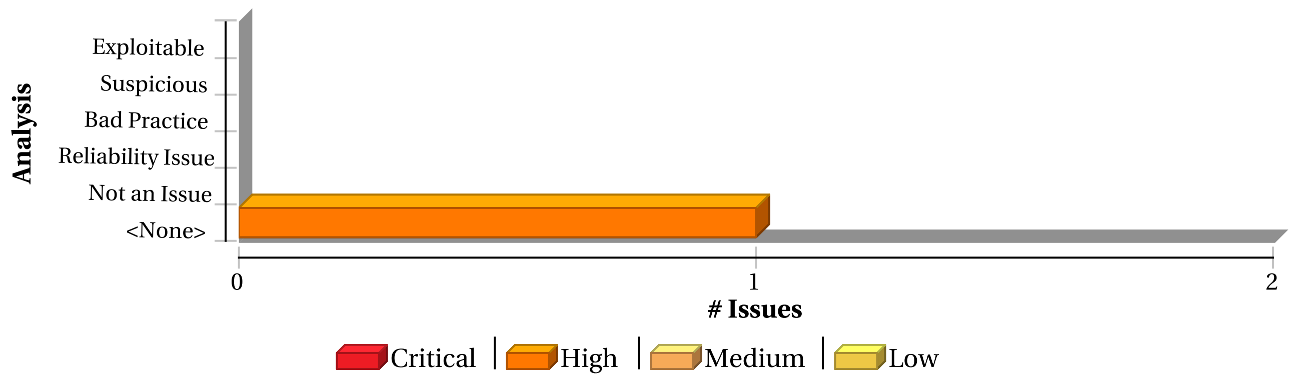
```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     synchronized(name) {
        name = req.getParameter("name");
     }
     ...
     out.println(name + ", thanks for visiting!");
   }
}
```

In order to fix the race condition, all the write and read operations on the shared member field should be run atomically within the same synchronized block:

```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     synchronized(name) {
        name = req.getParameter("name");
        ...
```

```
            out.println(name + ", thanks for visiting!");
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Race Condition: Singleton Member Field	1	0	0	1
Total	**1**	**0**	**0**	**1**

Race Condition: Singleton Member Field — High

Package: com.philips.tasy.agent.server.http

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 87 (Race Condition: Singleton Member Field) — **High**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

Sink Details

Sink: AssignmentStatement
Enclosing Method: innerinit^()
File: server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java:87
Taint Flags:

```
84
85 private void makeRootServlet(final HandlerCollection handlers) {
86 final ServletContextHandler sch = new ServletContextHandler(handlers, ROOT_CONTEXT);
87 final ServletHolder rootContextHolder = new ServletHolder(new HttpServlet() {
88 @Override
89 protected void doPost(final HttpServletRequest req, final HttpServletResponse resp)
90 throws ServletException, IOException {
```

System Information Leak: Incomplete Servlet Error Handling (2 issues)

Abstract

If a Servlet fails to catch all exceptions, it might reveal debugging information that will help an adversary form a plan of attack.

Explanation

When a Servlet throws an exception, the default error response the Servlet container sends back to the user typically includes debugging information. This information is of great value to an attacker. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used, and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.

Example 1: In the following method a DNS lookup failure will cause the Servlet to throw an exception.

```
protected void doPost (HttpServletRequest req,
                    HttpServletResponse res)
              throws IOException {
    String ip = req.getRemoteAddr();
    InetAddress addr = InetAddress.getByName(ip);
    ...
    out.println("hello " + addr.getHostName());
}
```

Example 2: The following method will throw a `NullPointerException` if the parameter "name" is not part of the request.

```
protected void doPost (HttpServletRequest req,
                    HttpServletResponse res)
              throws IOException {
    String name = getParameter("name");
    ...
    out.println("hello " + name.trim());
}
```

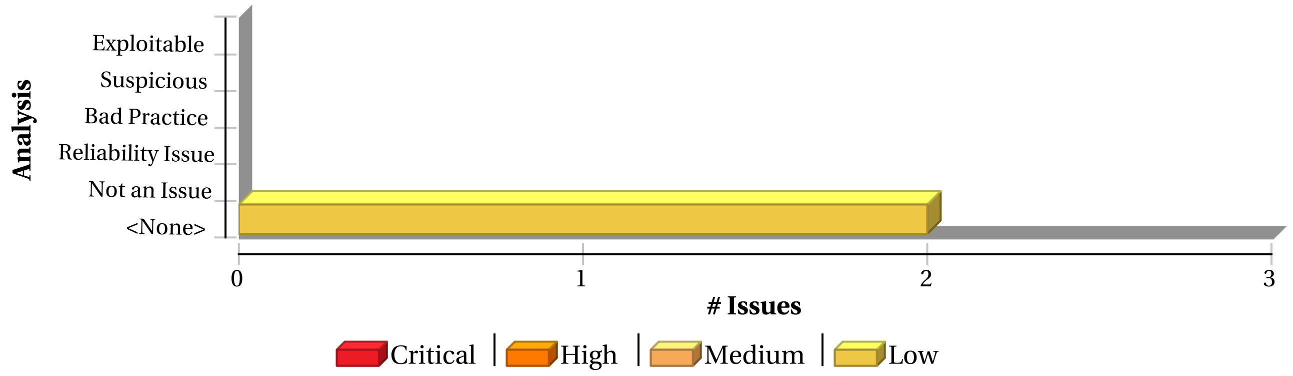
Recommendation

All top-level Servlet methods should catch `Throwable`, thereby minimizing the chance that the Servlet's error response mechanism is invoked.

Example 3: The method from Example 1 should be rewritten as follows:

```
protected void doPost (HttpServletRequest req,
                    HttpServletResponse res) {
      try {
          String ip = req.getRemoteAddr();
          InetAddress addr = InetAddress.getByName(ip);
      ...
          out.println("hello " + addr.getHostName());
      }catch (Throwable t) {
          logger.error("caught throwable at top level", t);
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: Incomplete Servlet Error Handling	2	0	0	2
Total	**2**	**0**	**0**	**2**

System Information Leak: Incomplete Servlet Error Handling — Low

Package: com.philips.tasy.agent.server.http

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 89 (System Information Leak: Incomplete Servlet Error Handling) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: doPost
Enclosing Method: doPost()
File: server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java:89
Taint Flags:

```
86 final ServletContextHandler sch = new ServletContextHandler(handlers, ROOT_CONTEXT);
87 final ServletHolder rootContextHolder = new ServletHolder(new HttpServlet() {
88 @Override
89 protected void doPost(final HttpServletRequest req, final HttpServletResponse resp)
90 throws ServletException, IOException {
91 resp.sendRedirect(CONTEXT_PATH);
92 }
```

System Information Leak: Incomplete Servlet Error Handling — Low

Package: com.philips.tasy.agent.server.http

server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java, line 95 (System Information Leak: Incomplete Servlet Error Handling) — **Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: doGet
Enclosing Method: doGet()
File: server-boot/src/main/java/com.philips.tasy.agent.server/http/WebServer.java:95
Taint Flags:

```
92 }
93
94 @Override
95 protected void doGet(final HttpServletRequest req, final HttpServletResponse resp)
96 throws ServletException, IOException {
97 resp.sendRedirect(CONTEXT_PATH);
98 }
```

Unreleased Resource: Sockets (2 issues)

Abstract

The program can potentially fail to release a socket.

Explanation

The program can potentially fail to release a socket.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool.

Example 1: The following method never closes the socket it opens. In a busy environment, this can result in the JVM using up all of its sockets.

```
private void echoSocket(String host, int port) throws UnknownHostException,
SocketException, IOException
{
  Socket sock = new Socket(host, port);
  BufferedReader reader = new BufferedReader(new
InputStreamReader(sock.getInputStream()));

  while ((String socketData = reader.readLine()) != null) {
    System.out.println(socketData);
  }
}
```

Example 2: Under normal conditions, the following fix properly closes the socket and any associated streams. But if an exception occurs while reading the input or writing the data to screen, the socket object will not be closed. If this happens often enough, the system will run out of sockets and not be able to handle any further connections.

```
private void echoSocket(String host, int port) throws UnknownHostException,
SocketException, IOException
{
  Socket sock = new Socket(host, port);
  BufferedReader reader = new BufferedReader(new
InputStreamReader(sock.getInputStream()));

  while ((String socketData = reader.readLine()) != null) {
    System.out.println(socketData);
  }
  sock.close();
}
```

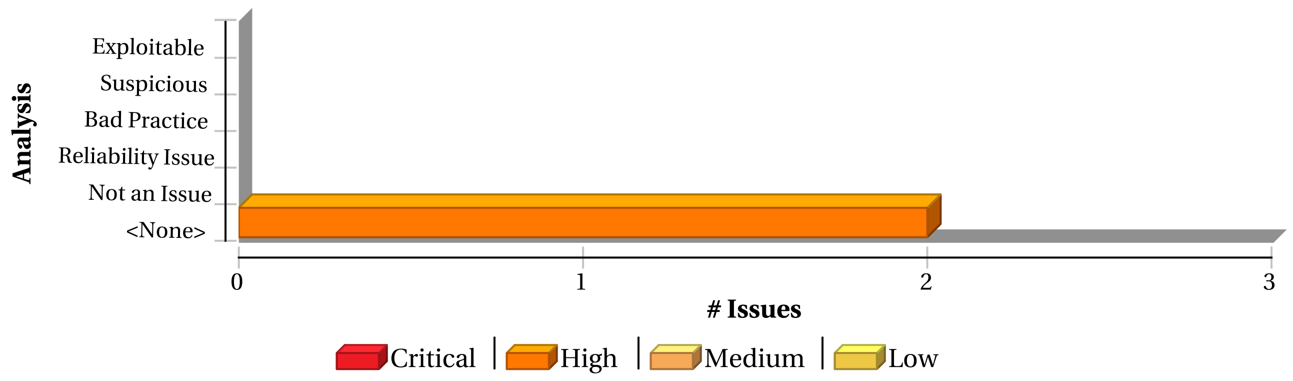
Recommendation
Release socket resources in a `finally` block. The code for Example 2 should be rewritten as follows:

```
private void echoSocket(String host, int port) throws UnknownHostException,
SocketException, IOException
{
  Socket sock;
  BufferedReader reader;

  try {
    sock = new Socket(host, port);
    reader = new BufferedReader(new InputStreamReader(sock.getInputStream()));

    while ((String socketData = reader.readLine()) != null) {
        System.out.println(socketData);
    }
  }
  finally {
    safeClose(sock);
  }
}

public static void safeClose(Socket s) {
  if (s != null && !s.isClosed()) {
    try {
      s.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the socket. Presumably this helper function will be reused whenever a socket needs to be closed.

Also, the `echoSocket()` method does not initialize the `sock` socket object to null. Instead, it checks to ensure that `sock` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `sock` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `sock` is initialized to `null` in a more complex method, cases in which `sock` is used without being initialized will not be detected by the compiler.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Sockets	2	0	0	2
Total	**2**	**0**	**0**	**2**

Unreleased Resource: Sockets — High

Package: com.philips.tasy.agent.commons.http

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 36 (Unreleased Resource: Sockets) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: out = s.getOutputStream()
Enclosing Method: stopServer()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:36
Taint Flags:

```
33 public static void stopServer(final int port) throws IOException {
34 try {
35 final Socket s = new Socket(InetAddress.getByName(LOCALHOST_ADDRESS), port);
36 final OutputStream out = s.getOutputStream();
37 LOGGER.info("*** sending jetty stop request");
38 out.write("STOP\r\n".getBytes(CHARSET_NAME));
39 out.flush();
```

commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 35 (Unreleased Resource: Sockets) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: s = new Socket(...)

Unreleased Resource: Sockets	High
Package: com.philips.tasy.agent.commons.http	
commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java, line 35 (Unreleased Resource: Sockets)	**High**

Enclosing Method: stopServer()
File: commons/src/main/java/com/philips/tasy/agent/commons/http/ServerMonitor.java:35
Taint Flags:

```

public static void stopServer(final int port) throws IOException {
try {
final Socket s = new Socket(InetAddress.getByName(LOCALHOST_ADDRESS), port);
final OutputStream out = s.getOutputStream();
LOGGER.info("*** sending jetty stop request");
out.write("STOP\r\n".getBytes(CHARSET_NAME));
```

Unreleased Resource: Streams (5 issues)

Abstract

The program can potentially fail to release a system resource.

Explanation

The program can potentially fail to release a system resource.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool.

Example: The following method never closes the file handle it opens. The `finalize()` method for `FileInputStream` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
private void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis = new FileInputStream(fName);
  int sz;
  byte[] byteArray = new byte[BLOCK_SIZE];
  while ((sz = fis.read(byteArray)) != -1) {
    processBytes(byteArray, sz);
  }
}
```

Recommendation

1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases.

Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang.

2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

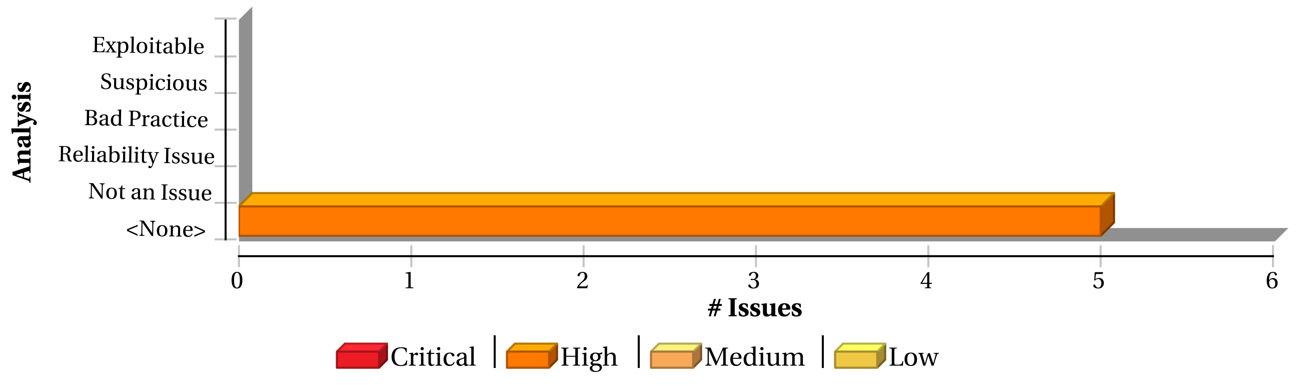
```
public void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis;
  try {
    fis = new FileInputStream(fName);
```

```
      int sz;
      byte[] byteArray = new byte[BLOCK_SIZE];
      while ((sz = fis.read(byteArray)) != -1) {
        processBytes(byteArray, sz);
      }
    }
    finally {
      if (fis != null) {
        safeClose(fis);
      }
    }
  }

  public static void safeClose(FileInputStream fis) {
    if (fis != null) {
      try {
        fis.close();
      } catch (IOException e) {
        log(e);
      }
    }
  }
```

This solution uses a helper function to log the exceptions that might occur when trying to close the stream. Presumably this helper function will be reused whenever a stream needs to be closed.

Also, the `processFile` method does not initialize the `fis` object to null. Instead, it checks to ensure that `fis` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `fis` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `fis` is initialized to `null` in a more complex method, cases in which `fis` is used without being initialized will not be detected by the compiler.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Streams	5	0	0	5
Total	**5**	**0**	**0**	**5**

Unreleased Resource: Streams — High

Package: com.philips.tasy.agent.client.core

client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java, line 49 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: newFilePrintStream(...)
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java:49
Taint Flags:

```
46 return;
47 }
48 if (!logToConsole) {
49 System.setOut(newFilePrintStream(new File(ClientPaths.logsFolder(), "output.log")));
50 System.setErr(newFilePrintStream(new File(ClientPaths.logsFolder(), "error.log")));
51 }
52 LogConfigurator.logToConsole = logToConsole;
```

client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java, line 50 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: newFilePrintStream(...)
Enclosing Method: run()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java:50
Taint Flags:

```
47 }
48 if (!logToConsole) {
49 System.setOut(newFilePrintStream(new File(ClientPaths.logsFolder(), "output.log")));
50 System.setErr(newFilePrintStream(new File(ClientPaths.logsFolder(), "error.log")));
51 }
52 LogConfigurator.logToConsole = logToConsole;
53 applyLogLevelConfiguration();
```

client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java, line 37 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Unreleased Resource: Streams — High

Package: com.philips.tasy.agent.client.core

client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java, line 37 (Unreleased Resource: Streams) — High

Sink Details

Sink: new PrintStream(new java.io.FileOutputStream(), ...)
Enclosing Method: newFilePrintStream()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/LogConfigurator.java:37
Taint Flags:

```
34
35 private static PrintStream newFilePrintStream(final File destination) {
36 try {
37 return new PrintStream(new FileOutputStream(destination), true, "UTF-8");
38 } catch (FileNotFoundException | UnsupportedEncodingException e) {
39 throw new IllegalArgumentException("Unable to create output", e);
40 }
```

Package: com.philips.tasy.agent.client.core.server

client-core/src/main/java/com/philips/tasy/agent/client/core/server/DeployCommand.java, line 33 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: in = getInputStream()
Enclosing Method: invoke()
File: client-core/src/main/java/com/philips/tasy/agent/client/core/server/DeployCommand.java:33
Taint Flags:

```
30 public Void invoke() throws Exception {
31 final String fileName = builder.getModuleName();
32 final String fileChecksum = builder.getMd5();
33 final InputStream in = builder.getInputStream();
34 final ApplicationContainer container = builder.getContainer();
35
36
```

Package: com.philips.tasy.agent.server.services

server/src/main/java/com/philips/tasy/agent/server/services/Lib.java, line 68 (Unreleased Resource: Streams) — High

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Unreleased Resource: Streams	High
Package: com.philips.tasy.agent.server.services	
server/src/main/java/com/philips/tasy/agent/server/services/Lib.java, line 68 (Unreleased Resource: Streams)	High

Sink: new FileInputStream(...)
Enclosing Method: getFile()
File: server/src/main/java/com/philips/tasy/agent/server/services/Lib.java:68
Taint Flags:

```
65  baseFile = new File(baseFile, storedLib.getFileName());
66
67  if (baseFile.exists()) {
68  response = Response.ok(new FileInputStream(baseFile), MediaType.APPLICATION_OCTET_STREAM)
69  .header(Libs.CONTENT_DISPOSITION, String.format(Libs.ATTACHMENT_FILENAME, storedLib.getFileName()))
70  .build();
71  } else {
```

Weak Cryptographic Hash: Missing Required Step (1 issue)

Abstract

The code misses invoking a required step during the process of generating a cryptographic hash.

Explanation

The generation of cryptographic hashes involves multiple steps, and missing any required step compromises the strength of the generated hashes.

Example 1: The following code skips the call to method `MessageDigest.update()` which will result in the creation of a hash based on no data:

```
...
MessageDigest messageDigest = MessageDigest.getInstance("SHA-512");

io.writeLine(MyUtilClass.bytesToHex(messageDigest.digest()));
....
```

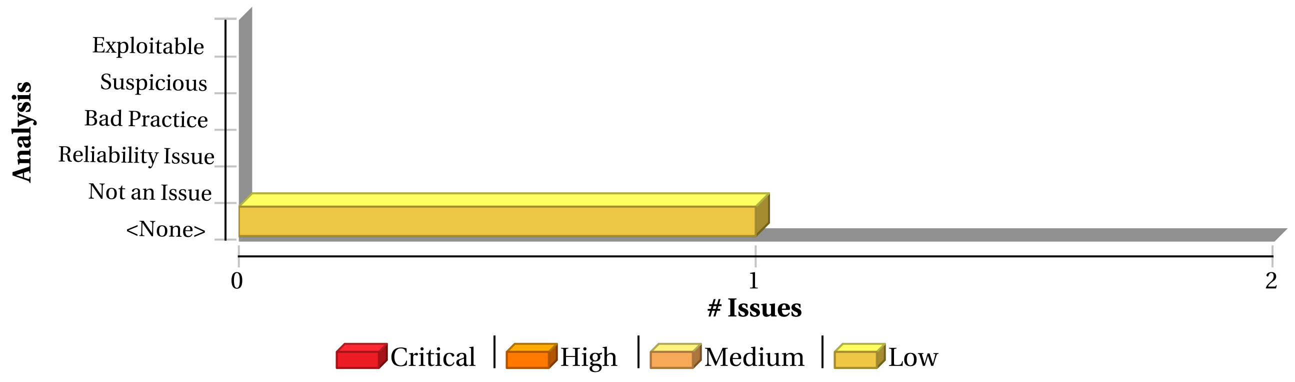
Recommendation

Implement all steps required in the generation of a cryptographic hash. Where possible, explicitly specify the parameters used to ensure that the strength of hash is not compromised.

Example 2: The code in Example 1 may be fixed as shown below:

```
...
MessageDigest messageDigest = MessageDigest.getInstance("SHA-512");
messageDigest.update(hashInput.getBytes("UTF-8"));
io.writeLine(MyUtilClass.bytesToHex(messageDigest.digest()));
....
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Cryptographic Hash: Missing Required Step	1	0	0	1
Total	**1**	**0**	**0**	**1**

Weak Cryptographic Hash: Missing Required Step — Low

Package: com.philips.tasy.agent.commons

commons/src/main/java/com/philips/tasy/agent/commons/Hash.java, line 49 (Weak Cryptographic Hash: Missing Required Step) — **Low**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Control Flow)

Sink Details

Sink: digest.digest() : Cryptographic hash finalized without update
Enclosing Method: checksum()
File: commons/src/main/java/com/philips/tasy/agent/commons/Hash.java:49
Taint Flags:

```
46 for (int length = in.read(block); length > 0; length = in.read(block)) {
47 digest.update(block, 0, length);
48 }
49 result = digest.digest();
50 } catch (Exception e) {
51 LOGGER.warn("Exception digesting file", e);
52 }
```