Final Team Project Documentation

Course :- Software Architecture (CSCI578)

Team Member :- Akash Bajaj (akashbaj@usc.edu)

Dhruv Patel (dhruvrpa@usc.edu)

Sumukh Bharadwaj Venkatesha Murthy (sumukhbh@usc.edu)

Architecture Recovery Technique *:-* ***ACDC Improvement***

GitHub Link for the project :- https://github.com/dhruvp-8/cs578-acdc-improvements

Architecture Security Strategy :-

Security decision to limit or block the attacker to upload malicious JSP file onto tomcat server and execute the same. However, with version 7, under some circumstance attacker can upload the JSP file onto the Tomcat server and remote execute the same.

Under these conditions[1], attacker can upload malicious JSP file onto Tomcat server :-

1. Using Oracle Java 1.7.0 update or earlier.
2. A web application must be deployed to a vulnerable version of Tomcat.
3. The web application must use Servlet 3.0 File upload feature.
4. A file location within a deployed web application must be writeable by the user the Tomcat process is running as. The tomcat security documentation recommends against this.
5. A custom listener for JMX connections (e.g. the JMXRemoteListener that is not enabled by default) must be configured and be able to load classes from Tomcat's common class loader (i.e. the custom JMX listener must be placed in Tomcat's lib directory).
6. The custom JMX listener must be bound to an address other than localhost for a remote attack (it is bound to localhost by default). If the custom JMX listener is bound to localhost, a local attack will still be possible.

Note that requirements 2 and 3 may be replaced with the following requirement:

7. A web application is deployed that uses Apache Commons File Upload 1.2.1 or earlier.

In this case (requirements 1, 4, 5, 6 and 7 met) a similar vulnerability may exist on any Servlet container, not just Apache Tomcat[1].

Problem :-

We are trying to extent/modify the ACDC algorithm such that it will be able to better cluster the files which are related to security decision/strategy for the system. Currently ACDC does not encounter the

such decision in their clusters. We are trying to modify the algorithm for the test case of Vulnerability **CVE-2013-4444** and above-mentioned security decision.

File involved :-

a. tomcat/tc7.0.x/trunk/java/org/apache/tomcat/util/http/fileupload/
b. tomcat/tc7.0.x/trunk/java/org/apache/tomcat/util/http/fileupload/FileItem.java

Working of ACDC[2] :-

There are mostly 2 different ways for clustering in large software systems :-

1. Knowledge-Based approach :- In this approach, reverse engineering and pre-existing domain knowledge is used to understand the working of different piece of code. Then the programs are clustered with respect to their similar or complementary functionality. For example :- procedure working on user I/O can be grouped into one cluster.
2. Structure Based approach :- In this approach, using syntactic interaction like call, fetch between entities we determine the composition of the software systems. The problem can be considered as partitioning of vertex of graph such that nodes are defined as procedure or variables and the edges as relation between them.

ACDC employs structure-based approach as Knowledge base certain limitation when it comes to large software systems due to various reason like lack of problem domain specific semantics, etc.

For solving any large problem, we always look for pattern whether it will be respect to problem data or solution. Pattern is used to solve problem in various field like planning, design, etc. When it comes to software clustering, patterns refer to familiar subsystems structures that frequently appears in manual decompositions of the large software systems. ACDC uses 2 prominent pattern type to decompose the large software system into clusters. They are :-

1. Body – Header Pattern :- The program files in C is divided into .c and .h files. They are many such programming languages where the files are divided into parts. The Body-Header patterns basically combined the divided files into one cluster.
2. Subgraph Dominator Pattern :- In this, pattern looks for the type of subgraph in the system. There must be a node n_o known as dominator and set of Nodes N $n_i = n_1 . n_2 , \ldots$. The node must satisfy the properties[2] :-
   a. There exists a path from n_0 to every n_i .
   b. For Any node v *E* V that there exists a path *P* from *v* to any n_i *E* N, we have either n_0 *E P or v E N.*

ACDC performs the cluster for the software system in two stages :-

Stage 1 :- In this stage, ACDC construct the skeleton of the system by using pattern-driven approach. In this stage, pattern like Body-Header, sub graph dominator is used to create the skeleton of the system with respect to the cluster. By convention, clusters are marked as subsystem and its names contain

suffix ".ss" to distinguish it from the files/classes. Apart from body-header and subgraph dominator, there are other pattern as well which ACDC consider deriving the skeleton of the system. The pattern is considered in below order of precedence :-

a. **Source File Cluster** :- ACDC clusters resources that reside in the same file together.
b. **Body-Header conglomeration** – Cluster based on Body-Header pattern.
c. **Leaf collection and support Library identification**:- ACDC identifies files for cluster according to leaf pattern and files as support library if they have in-degree greater than 20.
d. **Ordered and Limited subgraph domination**[2] : - This is the main step of the stage 1. In this step, subgraph dominator pattern logic is applied to define and find the cluster. Algorithm goes thru all the node in order to determine which satisfy the condition of the dominator node. The list of nodes is sorted in ascending order of the out-degree in order to reduce the cardinality for the final systems. When a non-empty dominated set is encountered, a new subsystem node is created with the base name + ".ss". *if the cardinality of the dominated set is more than 20 then it is not removed from the possible dominator node list to further decompose the subsystem.* If the cardinality is less than 20 for dominated set, then all nodes are removed from the possible dominator node list as they are not considered to be dominator anymore. After all nodes have been examined, ACDC organizes the obtain subsystems in hierarchical structure such as tree. All the subsystem which have cardinality low (4 or below) are removed and subsumed by higher subsystems. Finally, files left is considered again and combined into clusters.

After stage 1, almost major part of the system is divided into multiple clusters, but still some files might be left as they do not fit into any of the mentioned patterns.

Stage 2 :- In stage 2, orphan adoption technique is performed for the files left without clusters after stage 1. The technique is an incremental clustering technique where we assume that existing structure is well defined. It attempts to place the left file(orphaned files) into the cluster which it considers is suitable for. If any file cannot be assigned to any existing sub structure/cluster with confidence, then a new cluster is created for the left behind files and name of the cluster will be "leaf.ss" unless files resides in the same directory.

Reason for Not detecting the vulnerability/security strategy :-

1. ACDC employs hard condition of clustering that path to each element of a domination set(targets/dependent) must go through the dominator node. This result in deletion of dependency where files can have multiple path.
2. ACDC also divide the cluster with respect to structure of the package with multiple level. This result in files to divide in different cluster even with each having depends on same files.

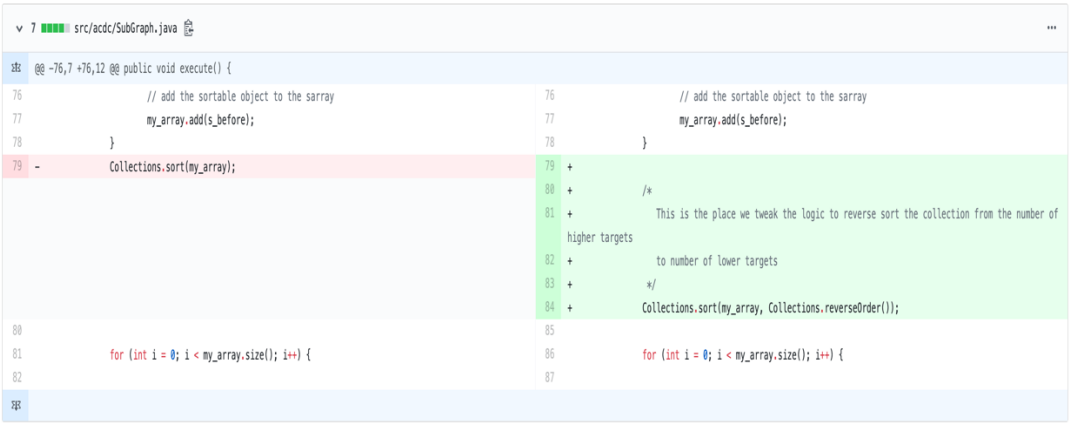
Before Change :-

```
contain org.apache.catalina.connector.ss org.apache.tomcat.util.buf.UEncoder$SafeCharsSet
contain org.apache.catalina.connector.ss org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory
contain org.apache.catalina.connector.ss org.apache.tomcat.util.http.fileupload.FileItem
contain org.apache.catalina.connector.ss org.apache.tomcat.util.http.fileupload.FileItemFactory
contain org.apache.catalina.connector.ss org.apache.tomcat.util.http.fileupload.FileItemHeadersSupport
contain org.apache.catalina.connector.ss org.apache.tomcat.util.http.fileupload.servlet.ServletRequestContext
contain org.apache.catalina.connector.ss org.apache.tomcat.util.net.URL
contain org.apache.catalina.connector.ss org.ietf.jgss.GSSCredential
contain org.apache.catalina.core.ss java.awt.Toolkit
```

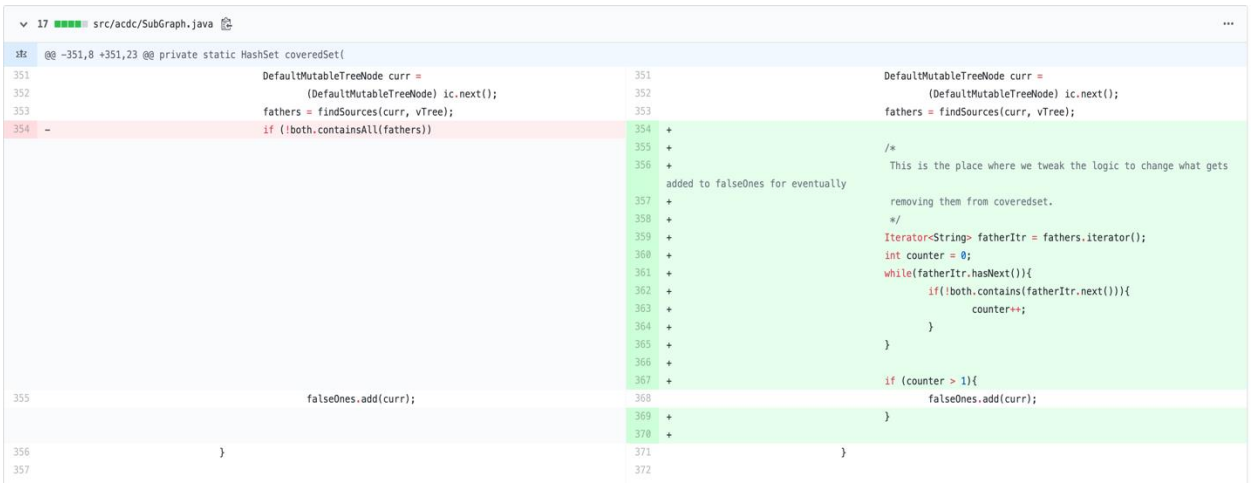
```
contain org.apache.tomcat.util.http.fileupload.disk.ss org.apache.tomcat.util.http.fileupload.disk.DiskFileItem
contain org.apache.tomcat.util.http.fileupload.servlet.ss org.apache.tomcat.util.http.fileupload.FileUpload
contain org.apache.tomcat.util.http.fileupload.servlet.ss org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload
contain org.apache.tomcat.util.http.fileupload.ss java.io.OutputStream
contain org.apache.tomcat.util.http.fileupload.ss java.io.UnsupportedEncodingException
```

Solution Changes –

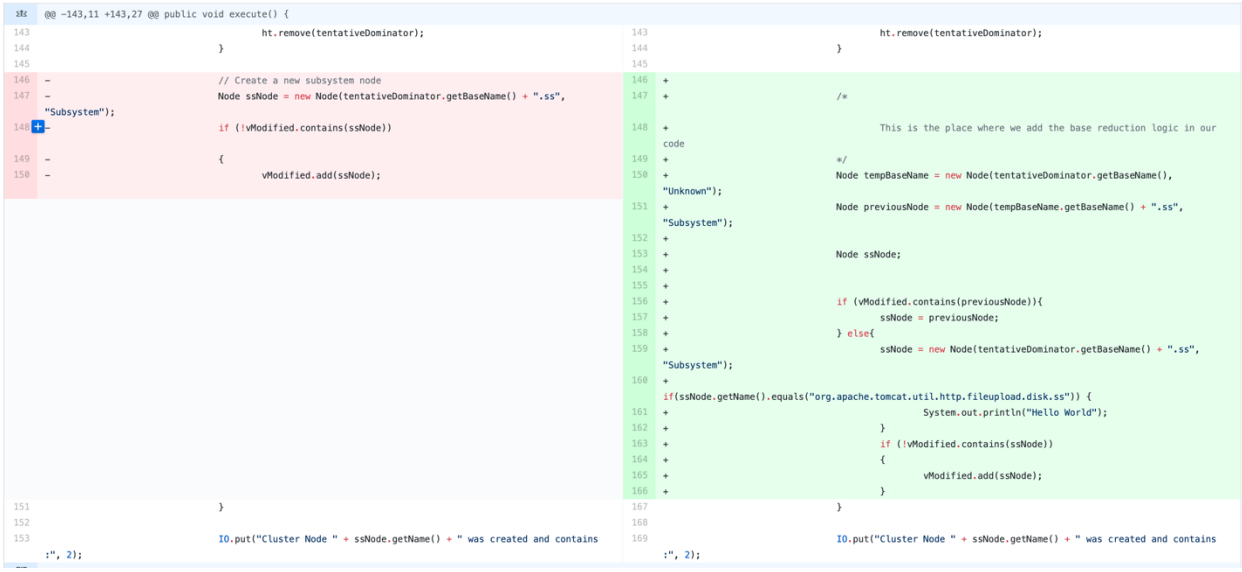
1. We reverse sort the Nodes/Class files with respect to number of dependent files. The sorting is done in descending order. We employ this method, in order to give more priority to nodes having more dependent files. This will also be better to cluster the nodes dependent on same files into one cluster.



2. We also relax the 1st condition from the reason to not detect. We removed the element from domination set if the element has more than 1 path to it which does not go thru its respective dominator node.



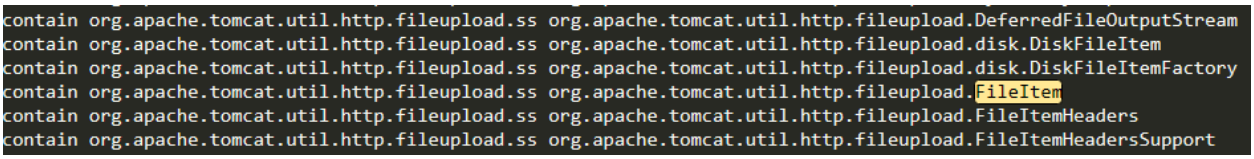
3. We also changed/limit the decomposition of clusters into multi-cluster with respect to structure. This helps in clustering the file of same functionality into the same cluster even if the files are declared into multiple level of same package.



Although above changes can detect and cluster the files responsible for a security strategy when the files responsible for the decision are declared under package with multiple level package structure.

After Change :-

```
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.DeferredFileOutputStream
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.disk.DiskFileItem
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.FileItem
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.FileItemHeaders
contain org.apache.tomcat.util.http.fileupload.ss org.apache.tomcat.util.http.fileupload.FileItemHeadersSupport
```



References & Citations:-

1. Tomcat Version 7 Security :- https://tomcat.apache.org/security-7.html
2. ACDC : An algorithm for Comprehension-Driven Clustering :- http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=5D97B85B2B245C9B9273037589545475?doi=10.1.1.42.8215&rep=rep1&type=pdf