swaldman / **c3p0** Public

Code Issues 70 Pull requests 3 Actions Projects Wiki Secur

Commit e14cbd8

swaldman committed on Feb 16

```
Reimplement userOverridesAsString without relying on unnecessarily dangerous Java
Serialization.
```

0.13.x · v0.12.0

1 parent 50a28b1 commit e14cbd8

3 files changed +111 -4

Top

Filter files...

- src/com/mchange/v2/c3p0
  - cfg
    - C3P0Config.java
  - impl
    - C3P0ImplUtils.java
- test/resources-local
  - c3p0.properties

Search within code

src/com/mchange/v2/c3p0/cfg/C3P0Config.java

```
@@ -11,6 +11,7 @@
11 11  import java.lang.reflect.Method;
12 12  import com.mchange.v1.lang.BooleanUtils;
13 13  import com.mchange.v2.c3p0.C3P0Registry;
   14 + import com.mchange.v2.csv.MalformedCsvException;
```

```
14  15
15  16    //all internal maps should be HashMaps (the implementation presumes HashMaps)
16  17
        @@ -439,7 +440,7 @@ public static Map getUserOverrides( String configName )
439 440         return (out.isEmpty() ? null : out );
440 441         }
441 442
442     -     public static String getUserOverridesAsString(String configName) throws
            IOException
    443 +     public static String getUserOverridesAsString(String configName) throws
            IOException, MalformedCsvException
443 444         {
444 445         Map userOverrides = getUserOverrides( configName );
445 446         if (userOverrides == null)
```

...com/mchange/v2/c3p0/impl/C3P0ImplUtils.java

```
        @@ -1,5 +1,6 @@
1   1   package com.mchange.v2.c3p0.impl;
2   2
    3 + import java.io.*;
3   4   import java.beans.*;
4   5   import java.util.*;
5   6   import java.lang.reflect.*;
        @@ -10,7 +11,8 @@
10  11  import com.mchange.v2.c3p0.*;
11  12  import com.mchange.v2.c3p0.cfg.*;
12  13
13      - import java.io.IOException;
    14  + import com.mchange.v2.csv.*;
    15  +
14  16  import java.sql.Connection;
15  17  import java.sql.SQLException;
16  18  import com.mchange.lang.ByteUtils;
        @@ -231,6 +233,12 @@ public static boolean supportsMethod(Object target,
        String mname, Class[] argTyp
231 233             }
232 234         }
233 235
```

```
      236 +     /*
      237 +     // Java Serialization-based userOverridesAsString format creates an unnecessary attack surface for
      238 +     // placing malicious objects in the serialized format and provoking deserialization.
      239 +     //
      240 +     // We'll transition to a simpler, less dangerous format.
      241 +
234   242       private final static String HASM_HEADER = "HexAsciiSerializedMap";
235   243
236   244       public static String createUserOverridesAsString( Map userOverrides ) throws IOException
          @@ -254,6 +262,90 @@ public static Map parseUserOverridesAsString( String userOverridesAsString ) thr
254   262           else
255   263               return Collections.EMPTY_MAP;
256   264       }
      265 +     */
      266 +
      267 +     // we serialize user overrides to "ragged CSV".
      268 +     // CSV lines containing only a single element are interpreted as the user for whom we are overriding config
      269 +     // lines following containing two elements are the config param overrides for that user.
      270 +     public static String createUserOverridesAsString( Map userOverrides ) throws IOException, MalformedCsvException
      271 +     {
      272 +         Writer w = new StringWriter();
      273 +         for (Object o : userOverrides.keySet())
      274 +         {
      275 +             // we don't check the type. We're treating this as basically an assertion, in our old-school, loosely typed Java
      276 +             // we'll let the user see a ClassCastException if something has been messed with
      277 +             String user = (String) o;
      278 +             w.append(FastCsvUtils.generateQuotedCsvItem(user));
      279 +             w.append("\r\n");
      280 +             Map userProps = (Map) userOverrides.get(user);
      281 +             String[] propNamePropValAsString = new String[2];
      282 +             for (Object pn : userProps.keySet())
```

```
283 +             {
284 +                 propNamePropValAsString[0] = (String) pn;
285 +                 propNamePropValAsString[1] = (String) userProps.get(pn);
286 +
      w.append(FastCsvUtils.generateCsvLineQuotedUnterminated(propNamePropValAsString
      ));
287 +                 w.append("\r\n");
288 +             }
289 +         }
290 +         return w.toString();
291 +     }
292 +
293 +     private static Map parseSingleUserMap(String userOverridesAsString,
      BufferedReader br, String[] nextUserHolder) throws IOException,
      MalformedCsvException
294 +     {
295 +         Map out = new HashMap();
296 +         nextUserHolder[0] = null;
297 +         String line = FastCsvUtils.csvReadLine(br);
298 +         if (line == null)
299 +         {
300 +             nextUserHolder[0] = null;
301 +             return Collections.EMPTY_MAP;
302 +         }
303 +         else
304 +         {
305 +             do
306 +             {
307 +                 String[] items = FastCsvUtils.csvSplitLine( line );
308 +                 switch ( items.length )
309 +                 {
310 +                 case 2: // this is an expected property override line
311 +                     out.put(items[0],items[1]);
312 +                     break;
313 +                 case 1: // this is the next user name
314 +                     nextUserHolder[0] = items[0];
315 +                     break;
316 +                 default:
317 +                     throw new IOException("Unexpected CSV line in
      userOverridesAsString ('" + line + "'). All line should have 1 or 2 items:\r\n"
```

```
+ userOverridesAsString);
318 +                 }
319 +             }
320 +             while (nextUserHolder[0] == null && (line =
    FastCsvUtils.csvReadLine(br)) != null); // either EOL or discovery of next user
    terminates
321 +             return Collections.unmodifiableMap(out);
322 +         }
323 +     }
324 +
325 +     public static Map parseUserOverridesAsString( String userOverridesAsString
    ) throws IOException, MalformedCsvException
326 +     {
327 +         String[] nextUserHolder = new String[1];
328 +         BufferedReader br = new BufferedReader(new
    StringReader(userOverridesAsString));
329 +         String line = FastCsvUtils.csvReadLine(br);
330 +         if ( line == null )
331 +             return Collections.EMPTY_MAP;
332 +         else
333 +         {
334 +             Map out = new HashMap();
335 +             String[] items = FastCsvUtils.csvSplitLine(line);
336 +             if (items.length != 1)
337 +                 throw new IOException("Cannot parse userOverridesAsString, one
    element line naming the user should come before other data:\r\n" +
    userOverridesAsString);
338 +             String username = items[0];
339 +             do
340 +             {
341 +                 Map overrides = parseSingleUserMap(userOverridesAsString, br,
    nextUserHolder);
342 +                 out.put(username, overrides);
343 +                 username = nextUserHolder[0];
344 +             }
345 +             while (username != null);
346 +             return Collections.unmodifiableMap(out);
347 +         }
348 +     }
257 349
```

```
258  350      public static void runWithContextClassLoaderAndPrivileges( final String
              contextClassLoaderSource, final boolean privilege_spawned_threads, final
              Runnable runnable )
259  351      {
```

test/resources-local/c3p0.properties

```
                @@ -54,7 +54,7 @@ c3p0.testConnectionOnCheckout=true
54   54         #com.mchange.v2.c3p0.cfg.xml=classloader:META-INF/poop.xml
55   55         #com.mchange.v2.c3p0.cfg.xml=/data/home/swaldman/development/gitproj/c3p0/src/t
                est-properties/META-INF/poop.xml
56   56
57        -    com.mchange.v2.log.MLog=com.mchange.v2.log.jdk14logging.Jdk14MLog
     57   +    #com.mchange.v2.log.MLog=com.mchange.v2.log.jdk14logging.Jdk14MLog
58   58         #com.mchange.v2.log.MLog=com.mchange.v2.log.FallbackMLog
59   59         #com.mchange.v2.log.FallbackMLog.DEFAULT_CUTOFF_LEVEL=FINER
60   60         #com.mchange.v2.log.FallbackMLog.DEFAULT_CUTOFF_LEVEL=ALL
                @@ -71,7 +71,7 @@
                com.mchange.v2.log.MLog=com.mchange.v2.log.jdk14logging.Jdk14MLog
71   71
72   72         #com.mchange.v2.resourcepool.experimental.useScatteredAcquireTask=false
73   73
74        -    # make sure we warn on unknown config
     74   +    #make sure we warn on unknown config
75   75         #c3p0.poop=stinky
76   76
77   77         #prop-style named config
                @@ -88,3 +88,17 @@
                com.mchange.v2.log.MLog=com.mchange.v2.log.jdk14logging.Jdk14MLog
88   88         #com.mchange.v2.c3p0.impl.DefaultConnectionTester.querylessTestRunner=IS_VALID
89   89         #com.mchange.v2.c3p0.impl.DefaultConnectionTester.querylessTestRunner=SWITCH
90   90         #com.mchange.v2.c3p0.impl.DefaultConnectionTester.querylessTestRunner=THREAD_LO
                CAL
     91   +
     92   +    #define params for a user called 'steve'
     93   +    #c3p0.user-overrides.steve.maxPoolSize=15
     94   +    #c3p0.user-overrides.steve.minPoolSize=5
     95   +
     96   +    #define params for a user called 'ramona'
     97   +    #c3p0.user-overrides.ramona.maxPoolSize=50
```

```
98  + #c3p0.user-overrides.ramona.minPoolSize=20
99  +
100 + #c3p0.user-overrides.swaldman.preferredTestQuery=SXLECT 1
101 +
102 +
103 +
104 +
```

Comments 0

Please sign in to comment.