



Arbitrary filesystem write access from velocity.

Details

Type:	Bug	Resolution:	Fixed
Priority:	Critical	Fix Version/s:	12.6.7, (2)
Affects Version/s:	2.3		
Component/s:	Old Core		
Labels:	attack_stability attacker_script security		
Environment:	*nix (for winx just switch the slashes.)		
Development Priority:	Low		
Difficulty:	Medium		
Documentation:	N/A		
Documentation in	N/A		
Release Notes:			
Pull Request Status:	Awaiting Committer feedback		
Similar issues:			

Description

This sample exploit abuses the file upload plugin but I reported this as a core bug because the problem is not constrained to the file upload plugin and I suspect I can find at least a half dozen other places where it manifests.

The root of the problem is that the File class is too widely available and thus too much code is relied upon to keep the filesystem safe.

The goods:

```
{{velocity}}
#if($request.getContentType().startsWith('multipart/'))
  #foreach($file in $xwiki.get('fileupload').getFileItems())
    #set($f = $file.getStoreLocation())
    #set($fi = $file)
  #end
  #set($f = $f.getAbsoluteFile())
  #foreach($part in $f.getCanonicalPath().split('/'))
    #set($path = $f.getParentFile())
  #end
  #set($newf = $f.getParentFile().getParentFile())
  #foreach($part in $request.getParameter('filename').split('/'))
    #if($part != '')
      #if($use != '')
        #set($use = '')
        #foreach($subf in $newf.listFiles())
          #if($subf.getName() == $part)
            #set($use = $subf)
          #end
        #end
      #end
    #end
    #if($use != '')
```

Issue Links

is related to

[XCOMMONS-2125](#) Class#getSimpleName is restricted by the SecureIntrospector

CLOSED

links to

[Github security advisory](#)

Activity

Newest first

Simon Urli added a comment - 06/Jan/21 10:29

Fixed by whitelisting some Files API in the velocity security introspector.

Thomas Mortagne added a comment - 23/Jul/14 17:58

You don't need to patch Velocity for that. We already override SecureIntrospectorImpl which is responsible for the existing configurable black list of classes and packages (see <https://github.com/xwiki/xwiki-commons/blob/master/xwiki-commons-core/xwiki-commons-velocity/src/main/java/org/xwiki/velocity/introspection/SecureIntrospector.java>). All we need to do is adding a more generic and configurable white listing in SecureIntrospector.

Andreas Jonsson added a comment - 25/Oct/11 19:23

Having thought about it for a while, I really like the idea about writing a SecurityManager. That would take some preparations, but it might also solve some of our other serious security problems at the same time.

Andreas Jonsson added a comment - 25/Oct/11 00:17

I tested a little, the basic idea seems to work, although the hook has to go into the class ASTReference.

```
Index: src/test/java/org/apache/velocity/test/ObjectVerifierTestCase.java
=====
--- src/test/java/org/apache/velocity/test/ObjectVerifierTestCase.java (revision 0)
+++ src/test/java/org/apache/velocity/test/ObjectVerifierTestCase.java (revision 0)
@@ -0,0 +1,182 @@
+package org.apache.velocity.test;
+
+/*
+ * Licensed to the Apache Software Foundation (ASF) under one
+ * or more contributor license agreements. See the NOTICE file
+ * distributed with this work for additional information
+ * regarding copyright ownership. The ASF licenses this file
+ * to you under the Apache License, Version 2.0 (the
+ * "License"); you may not use this file except in compliance
+ * with the License. You may obtain a copy of the License at
+ *
+ * http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing,
+ * software distributed under the License is distributed on an
+ * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
+ * KIND, either express or implied. See the License for the
+ * specific language governing permissions and limitations
```

CalebJamesDeLisle added a comment - 24/Oct/11 22:17

Interesting idea, I had considered writing a SecurityManager which peeked up the stack to determine if it is being called by trusted or untrusted code (if the jars are now signed then that's easier) and implemented that at a java level because then it would be applicable to not only velocity but groovy and other scripting languages.

Andreas Jonsson added a comment - 24/Oct/11 21:22


The general problem here is that of sanitizing the classes of the objects that are accessed from velocity code without programming rights. This is going to be a constant source of security flaws.

Here is what I propose to fix this: Use a custom velocity runtime, where the class of accessed objects are compared to a whitelist before use. I think that this would actually be easy to implement. I think that there are only two places which have to be guarded: execute method in the classes ASTMethod and ASTIdentifier. Hooks that validate the return value in those two places would completely guard against access to classes not on the whitelist.


Naturally, if programming rights are in effect, any object should be accepted and passed.

People

Assignee:

Simon Urli 

Reporter:

CalebJamesDeLisle 

Votes:

0 Vote for this issue

Watchers:

4 Start watching this issue

▼ Dates

Created:

07/May/10 06:21

Updated:

07/Jul/22 11:26

Resolved:

06/Jan/21 10:29

Date of First Response:

24/Oct/11 9:22 PM