XSS in Memory Analyzer plugin for Eclipse

Product description:

When a Java application hits the limit of the dedicated memory a Heap dump is generated in order to provide the developers a way to find the root cause of the Out of memory exception. There is a plugin for Eclipse who is useful for reading the generated Heap Dump <u>https://wiki.eclipse.org/MemoryAnalyzer</u>

Issue description:

The Memory Analyzer parses the Heap Dump and generates several files. One of the generated html files shows generated data and along with that the name of the Thread in which the error have occurred. However, if an attacker programmatically changes the name of the current thread (look at the code below) to some script it will be executed when rendered in Eclipse.

Steps to reproduce:

1) Create a malicious app that generates an Out of memory exception and changes the name of the current thread to a malicious script.

package com.sap;

import java.util.ArrayList; import java.util.List;

```
public class ExhaustMemory {
```

```
private void setThreadName() {
  Thread.currentThread().setName("<script>alert(1)</script>");
}
```

```
public static void main(String[] args) {
  List<String> list = new ArrayList<String>();
  ExhaustMemory mem = new ExhaustMemory();
  mem.setThreadName();
  while (true){
    list.add("Will reach out of memory soon...");
  }
}
```

- 2) Configure the VM arguments to instruct the VM to produce a Heap Dump and set the available memory a low number in order to speed up the Out of memory exception (Xms and Xmx are optional to be set. It is just faster setting them.)
- -XX:+HeapDumpOnOutOfMemoryError

-Xms1M

-Xmx2M

Name: OutOfMemory	
😉 Main Arguments 💦 🛃 JRE 🧞 Classpath 💱 Source 🖾 Environ	ment 🔲 Common
Program arguments:	
	Variables
VM arguments:	
-XX:+HeapDumpOnOutOfMemoryError -Xms1M -Xmx2M	
	Variables

3) Run the above Java application with that configuration

Out of memory error was thrown.

	- 0
1 package com.sap;	
∠ 3⊕import java.util.ArrayList;[]	
5	
6 public class ExhaustMemory {	
<pre>80 private void setThreadName() {</pre>	
<pre>9 Thread.currentThread().setName("<script>alert(1)</script>"); 10 3</pre>	
11	
120 public static void main(String[] args) {	
13 ExhaustMemory mem = new ExhaustMemory();	
<pre>15 mem.setThreadName();</pre>	
<pre>16 while (true){ 17 list.add("Will reach out of memory soon"):</pre>	
18 }	
19 } 20	
21 }	
22	
😰 Notes 💱 Navigation History 📮 Console 🕱 <terminated> OutOfMemory [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home/bin/java (Oct 8, 2019, 11:26:04 AM)</terminated>	= X 💥 🖡 🖬 🖉 💭 🗗 🖬 - 🗂 - 🗖
Wotes transformed to the second s	= ¥ ¾ № ₩ ₽ ₽ ₽ ₽ ₽ • • • • •
Wotes & Navigation History Console 23 <pre>terminated> OutOfMemory [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home/bin/java (Oct 8, 2019, 11:26:04 AM java.lang.OutOfMemoryError: Java heap space Dumping heap to java_pid59310.hprof Heap dump file created [1868048 bytes in 0.006 secs] Exception in thread "<script>alert(1)</script>" java.lang.OutOfMemoryError at java.util.Arrays.copyOf(Arrays.java:3210) at java.util.ArrayList.grow(ArrayList.java:265) at java.util.ArrayList.ensureExplicitCapacity(ArrayList.java:231) at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:231) at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:231) at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:231)</pre>	= ¥ ¾ ℝ ₩ ₽ ₽ ₽ ₽ ₽ • • • • •
Notes : Navigation History : Console :: terminated: OutOfMemory [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home/bin/java (Oct 8, 2019, 11:26:04 AM) java.lang.OutOfMemoryError: Java heap space Dumping heap to java_pid59310.hprof Heap dump file created [1868048 bytes in 0.006 secs] Exception in thread " <script>alert(1)</script> " java.lang.OutOfMemoryError at java.util.Arrays.copyOf(<u>Arrays.java:3210</u>) at java.util.ArrayList.grow(<u>ArrayList.java:265</u>) at java.util.ArrayList.ensureExplicitCapacity(<u>ArrayList.java:239</u>) at java.util.ArrayList.add(<u>ArrayList.java:462</u>) at com son ExhaustMemory main(ExhaustMemory inva:17)	= X 🔆 R R B P P P d D • C • - □
Wotes : Navigation History : Console :: terminated: OutOfMemory [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home/bin/java (Oct 8, 2019, 11:26:04 AM java.lang.OutOfMemoryError: Java heap space Dumping heap to java_pid59310.hprof Heap dump file created [1868048 bytes in 0.006 secs] Exception in thread " <script>alert(1)</script> " java.lang.OutOfMemoryError at java.util.Arrays.copyOf(<u>Arrays.java:3210</u>) at java.util.ArrayList.grow(<u>ArrayList.java:265</u>) at java.util.ArrayList.ensureExplicitCapacity(<u>ArrayList.java:239</u>) at java.util.ArrayList.ensureCapacityInternal(<u>ArrayList.java:231</u>) at java.util.ArrayList.add(<u>ArrayList.java:462</u>) at com.sap.ExhaustMemory.main(<u>ExhaustMemory.java:17</u>)	E X X R R R P P P C P C C C C C C C C C C C C
Notes transformer to available to availa	Updates Available for your software. Click to review and install updates.

4) Open the generated Heap Dump file (.hprof) with Eclipse



5) Select "Leak Suspects Report" option in the wizard

ExhaustMemory	y.java 📔 java_pid59310.hprof 🕱	
i III 🖫 💀 🎕	≥ № • € ₀ • ♀	
i Overview 🔀		
- Details		
Size: 706 KB Cl	asses: 443 Objects: 6.2k Class Loader: 3 Unreachable Objects Histogram	
▼ Biggest Obj	Getting Started Wizard	
(Getting Started	
	Choose one of the common reports below. Press Escape to close this dialog.	
	Leak Suspects Report	
A a	Automatically check the heap dump for leak suspects. Report what objects are kept alive and why they are not garbage collected.	
(Component Report	
A	Analyze a set of objects for suspected memory issues: duplicate strings, empty collections, finalizer, weak references, etc.	
(Re-open previously run reports	
E	existing reports are stored in ZIP files next to the heap dump.	141.3 КВ
	Show this dialog when opening a heap dump.	
	Cancel Finish	-
- Actions	✓ Reports	✓ Step By Step
II Histogram : I E Dominator Tri keep alive. <u>Top Consum</u> grouped by o <u>Duplicate Cli</u> class loaders	Lists number of instances per class tege: List the biggest objects and what they tegs: Print the most expensive objects class and by package. apper: Detect classes loaded by multiple 5.	d a system <u>Component Report</u> : Analyze objects which belong to a common root package or class loader.

6) When rendered the malicious code is executed in the Memory Analyzer perspective in Eclipse



The malicious code is executed not only on the first rendered page but on every page where the Thread name is shown like this one:

	u - ⊘ <i>⊝ ∧</i> - ∘	· 3) · (2) (2) · (2) ·	Quick Access 🔡 😫 👹 💞 🎋 PyDev
🖳 Inspector 🕄 🏻 😤 🗖 🗖	ExhaustMemory.java Signa_pid59310.hprof		
		i III 🖫 💀 🖗 🔎 - 🗞 - 🔍	
		i Overview Codefault_report org.eclipse.mat.api:suspects 🔀	
		Leak Suspects » Leaks » Problem Suspect 1	
		- Description	
Statics Attributes Class Hierarchy Value	•8	The thread java.lang.Thread @ 0x7bff86508	
Type Name Value			
		Javascript	
		1	
		ОК	
			Updates Available
			Updates are available for your software
			1 10X TO FOULDH 200 IDSTALL UD03105