Java Heap Dump Sanitizer

Myo Ohn
myohn@paypal.com
June 2020

Summary. We present a tool for redacting data from Java heap dump files so that sensitive data is sanitized and prevented from leaking out of a controlled production environment. The sanitized dump file, sans sensitive data, can then be redistributed to a local development environment for meaningful analysis using interactive graphical programs.

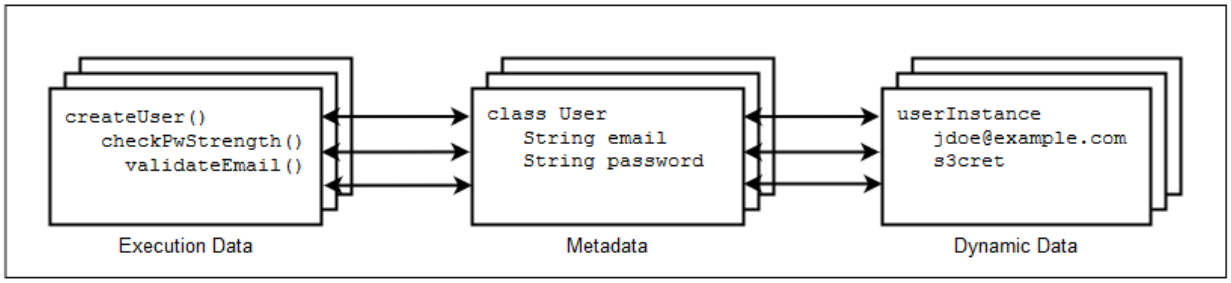
1. Introduction

A heap dump file for a Java process can be captured by, for example, executing an industry-standard tool such as jcmd at the command line:

```
$ jcmd <pid> GC.heap_dump <heap-dump-file>
```

The resulting output file contains a snapshot of the memory contents and program execution, such as:

- Execution data: stacktrace of live threads
- Metadata: all Java class names and field names
- Dynamic data: all Java objects and field values



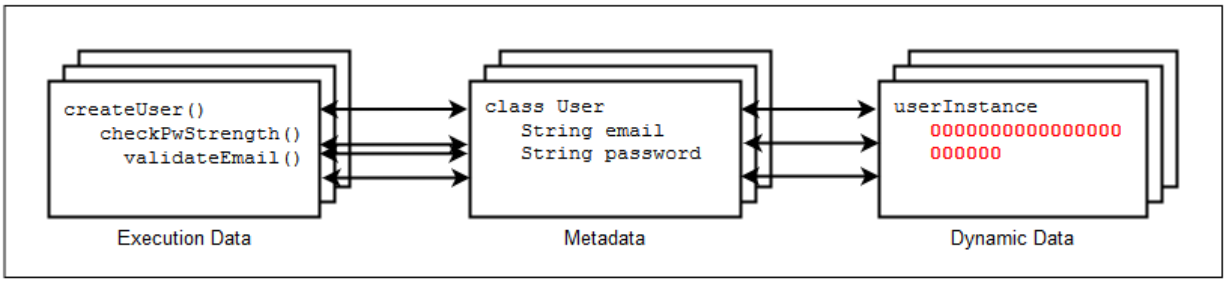
Therefore, it contains key information needed to accurately diagnose and debug memory problems that would otherwise be extremely impractical or impossible.

Analyzing the raw heap dump file however requires using interactive graphical programs such as Eclipse Memory Analyzer or VisualVM on a local development machine. Therefore, if captured on a production server, heap dump needs to be copied or moved to a different environment. Such an action would however trigger a high-risk security incident because heap dump contains all dynamic data, potentially including sensitive data such authentication tokens, passwords, or account numbers. Consequently, capturing a production heap dump or copying it out of the production environment is prohibited by information security policies.

What is needed is a tool that can scrub the dynamic data in the heap dump file so that the security risk is eliminated. The heap dump can then be responsibly copied and redistributed without fear of exfiltration of sensitive data.

2. Heap Dump Sanitizer

Heap dump sanitizer is a tool that first parses the heap dump file according to the documented Java specification and then replaces the field values with zero or null values. Metaphorically, it's as if we took a marker pen and redacted sensitive words on the heap dump paper. Or as if we performed a find/replace on a text document so that all characters are replaced with "0". Therefore, all data in heap dump, including authentication tokens, passwords, or account numbers, etc. would become indistinguishable blobs of zero values.



Sanitized Heap Dump File

Even though field values are sanitized, meaningful analysis of the program behavior can be performed because we retain logical structure of the heap dump. For example, we can find out which method or class references excessive amount of dynamic data.

The tool can be run at the command line as follows:

```
$ java -jar sanitizer.jar <original-heap-dump> <sanitized-heap-dump>
```

3. Business Value Proposition

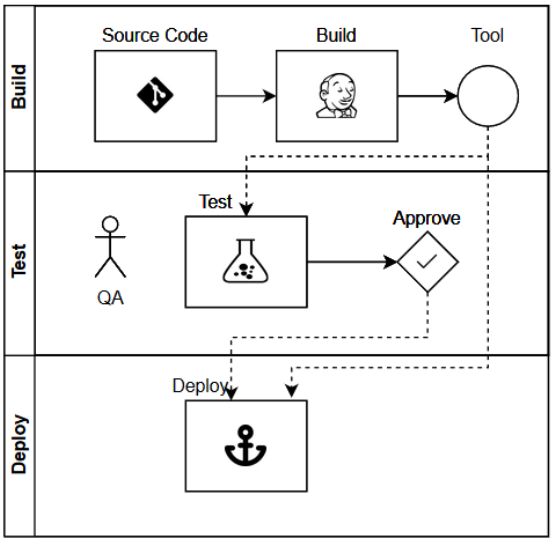
From time to time, our applications can become distressed due to unexpectedly high memory usage, leading to service degradation and interruption. To ensure delightful user experience and avoid loss of revenue, it's critical to be able to quickly find the root cause.

Unfortunately, such problems can be difficult to replicate in a development environment because they are typically triggered by unique characteristics in a production environment such as volume of data and amount of traffic. Further, traditional methods such as log analysis or performance metrics can provide only vague and circumstantial hints of the root cause. Therefore, such traditional methods cannot provide quick answers. We often have to employ multiple rounds of guesswork, software change, build and deployment cycle.

Inspecting a sanitized heap dump can bypass those steps and quickly pinpoint the hotspots. Instead of laboriously slogging for a needle solution in the metaphorical haystack, the heap dump sanitizer can magnetically pull out the needle and solve the problem. Instead of guesswork, we can leverage sophisticated heap dump analyzers to hone in on the answer. The tool can be a lifesaver in stressful emergency situations.

4. Build and Deployment

Because source code for this Java tool is available, we can build and deploy it like we would for any other Java programs we develop, following the same approved procedures. Note that, once deployed, the tool would not run continuously like a user-facing webapp or a backend service. Rather it would be manually run from the command-line on as-needed basis by operational staff like they would run any other command-line troubleshooting tools.



5. Operating Procedure

To request the operational staff to run the tool, the application developer must follow the same procedures like requesting any other troubleshooting assistance:

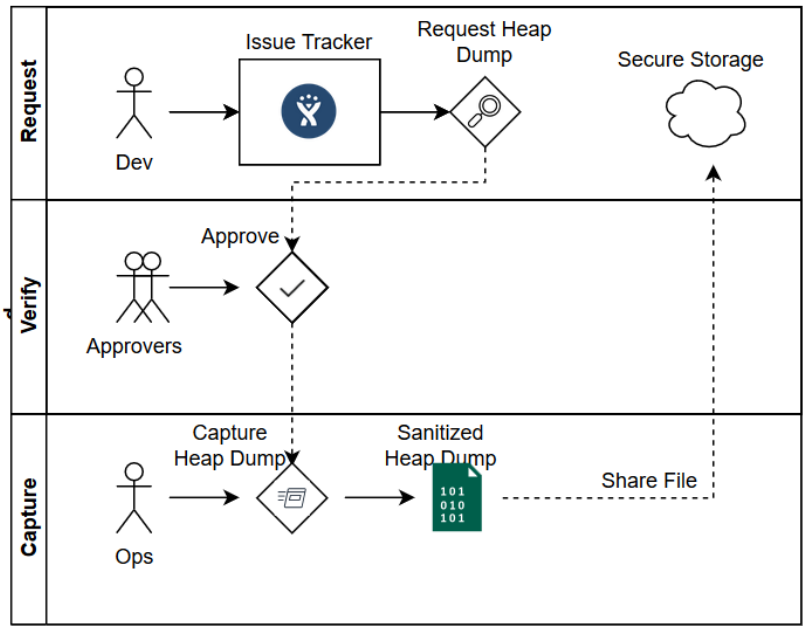
- Create an issue ticket, which must be approved by the supervisor or other designated person.
- Operational staff would capture the sanitized heap dump and upload it an approved online file sharing system.

To capture the sanitized heap dump, the operational staff would first login to the production server like they would normally by following approved procedures. For example, they would connect to VPN, perform multi-factor single-sign-on authentication, and then login to the server using individualized ssh key.

Once logged in, the operational staff would capture a sanitized heap dump file and then upload it to an approved secure file sharing/storage system:

```
$ jcmd <pid> GC.heap_dump heap-dump-file
$ java -jar sanitizer.jar heap-dump-file sanitized-heap-dump-file
$ upload-to-secure-file-storage sanitized-heap-dump-file
$ rm heap-dump-file sanitized-heap-dump-file
```

The application developer can then download the sanitized heap dump file from file storage system, and analyze the saved application behavior without being able to access sensitive production data.



6. Risk Management

Source code for the sanitizer tool is relatively small and available. Therefore, we can line-by-line review the code to ensure that the tool does not introduce risk. We can verify that, for example, it does not exfiltrate data, connect to the internet, or perform any kind of malicious activity. We can verify that the only action it performs is sanitizing the input heap dump file.

We will follow the same approved procedures for QA testing, building and deploying the tool. That way we minimize the chance of introducing bugs into the tool.

To request the operational staff to run the tool, the application developer has to create an issue ticket, just like when requesting any other troubleshooting assistance in production. That way we follow the same approved procedures. We also maintain an audit trail of who, what, and when of the invocation.

Operational staff will be only direct users of the tool. These users have specific roles and strong authentication is required, following the same approved procedures to operating in production.

The captured and sanitized heap dump files are uploaded to an approved secure file sharing system that is accessible only the authenticated corporate users. The files will be shared only to those on as-needed basis.

7. Legal and Licensing

We anticipate absolutely no legal or licensing entanglements. The tool is based on public-domain information in specification documents or software under business-friendly permissive licenses.

8. Additional Info

- Source code of the tool: https://github.paypal.com/myohn/heap-dump-sanitizer
- More engineering info on the tool. Visualization screenshots: https://engineering.paypalcorp.com/confluence/display/~myohn/Heap+Dump+Sanitizer