

# Similarity between Qealler/Pyrogenic variants -Part 0x3

---

[securityinbits.com/malware-analysis/similarity-between-qealler-pyrogenic-variants-part-0x3/](https://securityinbits.com/malware-analysis/similarity-between-qealler-pyrogenic-variants-part-0x3/)

February 4, 2020

In this last part, we will compare Old Qealler with the new Qealler/Pyrogenic variant. The previous posts [Pyrogenic Infostealer static analysis – Part 0x1](#) & [Unpacking Pyrogenic/Qealler using Java agent -Part 0x2](#) went through the latest Pyrogenic/Qealler <sup>[6]</sup> statically and dumping the unpacked code using Java agent.

## CONTENTS

### **Brief Timeline**

---

First Old Qealler sample <sup>[4]</sup> (MD5: 65ab1ef8e9cef5c489d4b01cbb8a2a22) found on ANY.RUN

The tweet<sup>[1]</sup> by @James\_inthe\_box first mentioned the Old Qealler. @jeFF0Falltrades posted Qealler Unloaded deep dive analysis <sup>[2]</sup> .

Multiple cyber security company posted articles<sup>[3]</sup> about Qealler variant using the Qazagne Python credential harvester.

Based on ANY.RUN submissions, Old Qealler variant using the Qazagne **stopped** around April 2019

Based on ANY.RUN submissions, Qealler tagged samples started around Aug 2019 and continue till now Aug 2020.

### **June 2018**

First Old Qealler sample <sup>[4]</sup> (MD5: 65ab1ef8e9cef5c489d4b01cbb8a2a22) found on ANY.RUN

### **Aug - Sep 2018**

The tweet<sup>[1]</sup> by @James\_inthe\_box first mentioned the Old Qealler. @jeFF0Falltrades posted Qealler Unloaded deep dive analysis <sup>[2]</sup> .

### **Jan-Feb 2019**

Multiple cyber security company posted articles<sup>[3]</sup> about Qealler variant using the Qazagne Python credential harvester.

### **Apr 2019**

Based on ANY.RUN submissions, Old Qealler variant using the Qazagne **stopped** around April 2019

## Aug 2019 - Now

Based on ANY.RUN submissions, Qealler tagged samples started around Aug 2019 and continue till now Aug 2020.

### grade

Note: When this post mention Old Qealler it means that the variant which was using the Qazagne Python credential harvester.

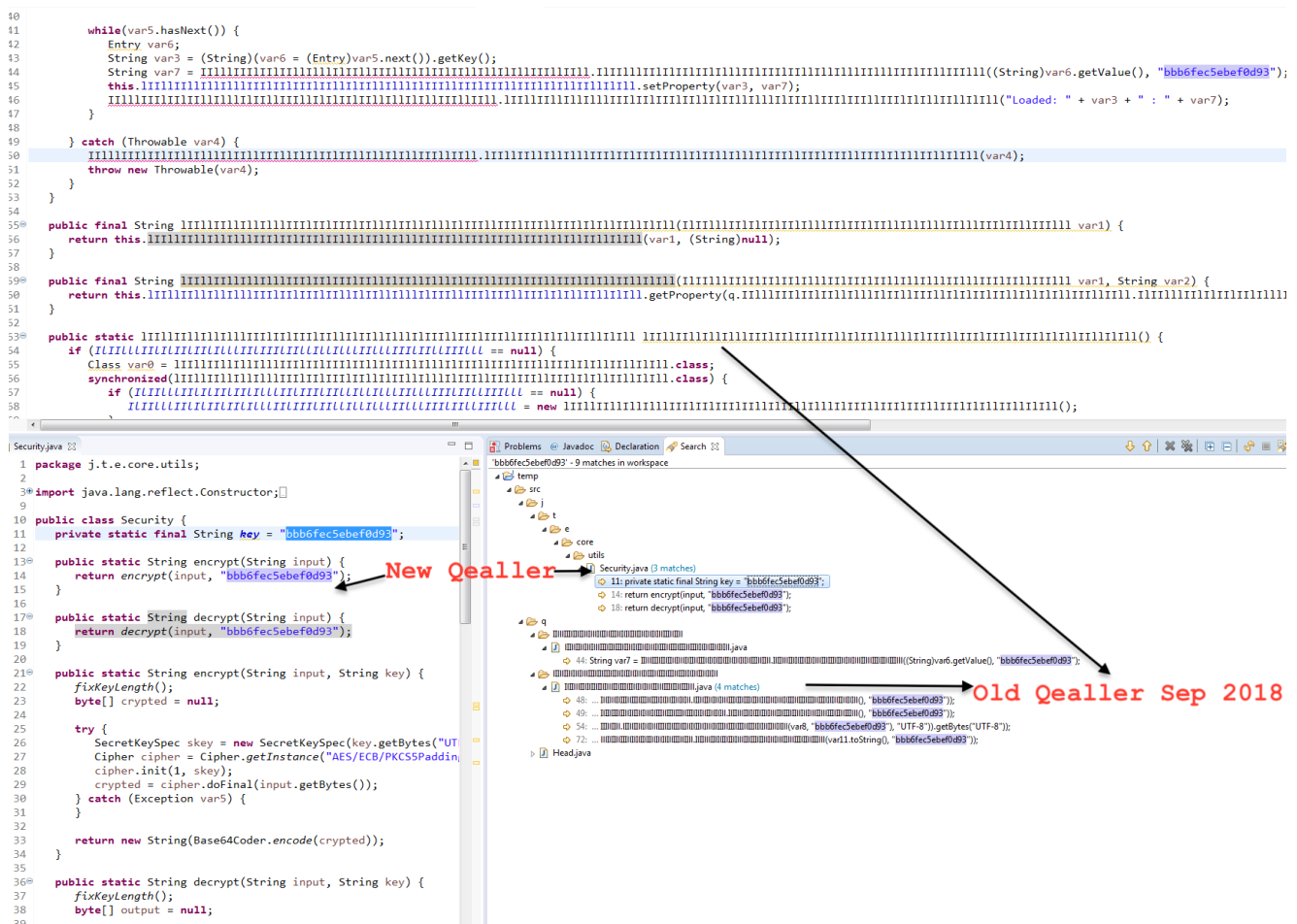
## Similarity between Qealler variants

For easy/fast comparison, I have imported unpacked code of both Qealler variant in Eclipse IDE. I will compare this Old Qealler (MD5: 8D564A18B902461C19936CCB1F4E2F12) [5] and new Pyrogenic/Qealler sample (MD5: F0E21C7789CD57EEBF8ECDB9FADAB26B) [6] used in the previous posts. Highly recommended to read through the existing analysis of Old Qealler Unloaded [2] by @jeFF0Falltrades & article [3] by Zscaler.

Both Qealler variants use the same Qrypter packer variant.

### 1. AES Key bbb6fec5ebef0d93

You will find multiple references to bbb6fec5ebef0d93 as shown below. This is the AES key used in both variant.



```
10 while(var5.hasNext()) {
11     Entry var6;
12     String var3 = (String)(var6 = (Entry)var5.next()).getKey();
13     String var7 = .....((String)var6.getValue(), "bbb6fec5ebef0d93");
14     this ..... .setProperty(var3, var7);
15     .....("Loaded: " + var3 + " : " + var7);
16 }
17 }
18 }
19 } catch (Throwable var4) {
20     .....(var4);
21     throw new Throwable(var4);
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }

public final String .....(var1, (String)null);
}

public final String .....(var1, String var2) {
return this ..... .getProperty(q .....);
}

public static .....() {
if (..... == null) {
class var0 = .....;
synchronized(.....) {
if (..... == null) {
..... = new .....();
}
}
}
}

package j.t.e.core.utils;
import java.lang.reflect.Constructor;
public class Security {
private static final String key = "bbb6fec5ebef0d93";
public static String encrypt(String input) {
return encrypt(input, "bbb6fec5ebef0d93");
}
public static String decrypt(String input) {
return decrypt(input, "bbb6fec5ebef0d93");
}
public static String encrypt(String input, String key) {
fixKeyLength();
byte[] crypted = null;
try {
SecretKeySpec skey = new SecretKeySpec(key.getBytes("UTF-8"), "AES");
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(1, skey);
crypted = cipher.doFinal(input.getBytes());
} catch (Exception var5) {
}
return new String(Base64Coder.encode(crypted));
}
public static String decrypt(String input, String key) {
fixKeyLength();
byte[] output = null;
}
```

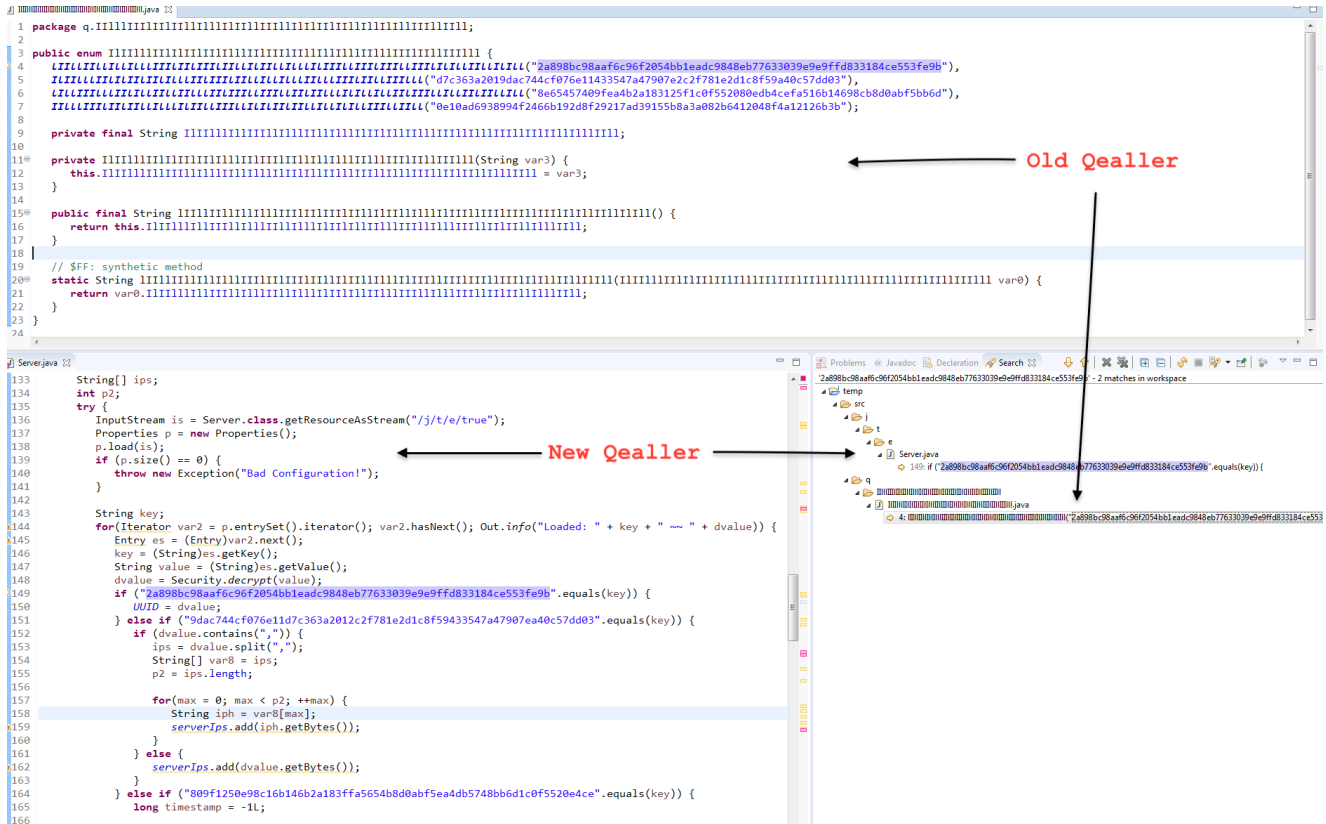
Search results for "bbb6fec5ebef0d93" - 9 matches in workspace:

- Security.java (3 matches)
  - 11: private static final String key = "bbb6fec5ebef0d93";
  - 14: return encrypt(input, "bbb6fec5ebef0d93");
  - 18: return decrypt(input, "bbb6fec5ebef0d93");
- Security.java (4 matches)
  - 48: .....("Loaded: " + var3 + " : " + var7);
  - 49: .....(var6, "bbb6fec5ebef0d93");
  - 54: .....(var6, "bbb6fec5ebef0d93", "UTF-8");
  - 72: .....(var1.toString(), "bbb6fec5ebef0d93");
- Head.java

Same AES Key bbb6fec5ebef0d93

## 2. UUID Key 2a898bc98aaf6c96f2054bb1eadc9848eb77633039e9e9ffd833184ce553fe9b

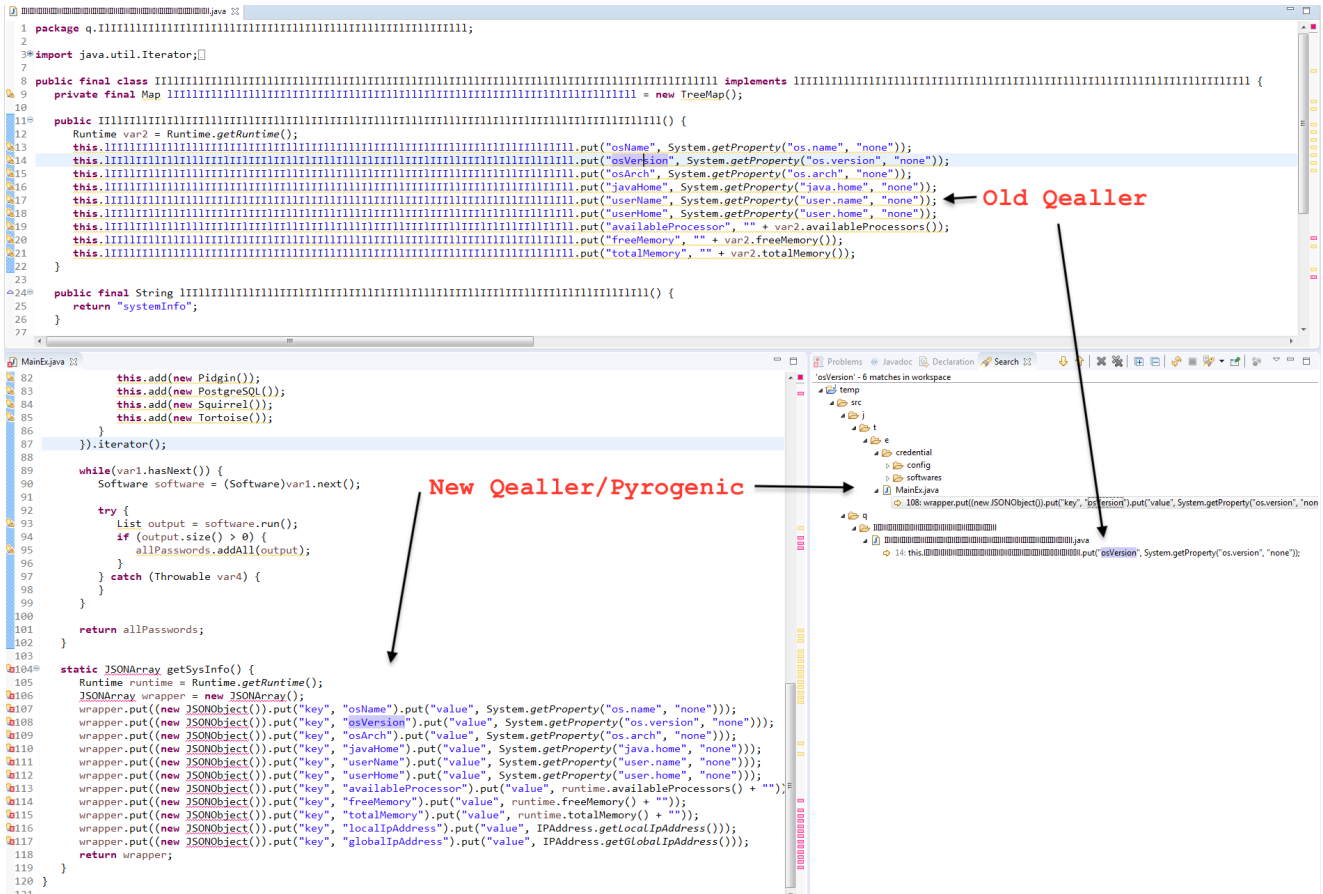
Config is stored in a key value pair and key for UUID present in both variants. This key is also present in the Old Qealler Unloaded<sup>[2]</sup> article and the same string “Loaded:” is used in both variants.



Same UUID key

## 3. Systeminfo in JSON format

It collects the system info in JSON format before encrypting and sending it to CC. Both Qealler variants use the same key e.g **osName**, **osVersion**, **osArch**, **totalMemory** and code structure as shown below. **localIpAddress** & **globalIpAddress** keys are added to the new Qealler version.



### Systeminfo in JSON format Qealler Pyrogenic

#### 4. ShutdownHook

It is used when we want to run some code when JVM is shutting down and both variants use the **addShutdownHook()** to delete the files.

```

1 package q;
2
3 import java.io.File;
4
5 public final class {
6     private static final List<File> files = new LinkedList();
7
8     public static void add(File var0) {
9         files.add(var0);
10    }
11
12    // $FF: synthetic method
13    static List<File> files() {
14        return files;
15    }
16
17    static {
18        Runtime.getRuntime().addShutdownHook(new Thread(new Runnable() {
19            public void run() {
20                // ...
21            }
22        }));
23    }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

### ShutdownHook

#### 5. QeallerV4 string

Found this string “**obfuscated/META-INF/QeallerV4.kotlin\_module**” in memory in the new Qealler/Pyrogenic sample. Maybe this is *version 4* ?

### Conclusion

In this Java malware analysis series we started with static analysis, then moved to Unpacking code using Java agent and in this last part we compared the Qealler variant. These above similarities are the most significant which I can find based on code analysis. I can conclude that the Malware author moved the Credential stealing from Python to Java based code. Malware authors are experienced coder

as they divided the source code in multiple sensible packages and gave proper name to functions, variables and classes.

Hope you enjoyed this post, please [Follow @Securityinbits](#) me on Twitter to get the latest update about my malware analysis & DFIR journey. Happy Reversing 😊

### References

1. [Tweet by @James\\_inthe\\_box](#) – (MD5: 65ab1ef8e9cef5c489d4b01cbb8a2a22) First Old Qealler tweet Aug 2018
2. [Qealler Unloaded by @jeFF0Falltrades](#) – Sep 2018

3. Qealler – a new JAR-based information stealer – Feb 2019
4. ANY.RUN – (MD5: 65ab1ef8e9cef5c489d4b01cbb8a2a22) Old Qealler June 2018
5. ANY.RUN – (MD5: 8D564A18B902461C19936CCB1F4E2F12) Old Qealler Sep 2018
6. ANY.RUN – (MD5: F0E21C7789CD57EEBF8ECDB9FADAB26B) New Qealler/Pyrogenic Nov 2019