Samba SPY

Description

Your organization has discovered an infection on one of its systems involving a malicious Java application. This malware performs environment checks to ensure it is not running inside a virtual machine and targets systems with specific configurations. Once the required conditions are met, it extracts files and executes malicious components that could compromise sensitive data or system integrity. The stealthy nature of the malware and its ability to evade detection pose a serious threat, requiring immediate action to secure the network and prevent further compromise.

Research Objectives

1. What is the name of the method thast check if the program is running inside a vm?

2. What system language is required for the program co continue execution?

3. What is the name of the method responsible for extracting the Prodotto.zip?

4. What is default path for .zip contents?

5. What file name does the programm look for after extraction to run as a JAR file?

6. What command is used to execute the extracted JAR file?

7. What process is used to check if the system is running a virtual machine besides the manufacturer string?

8. How many virtual machine vendors does the program check for?

Walkthrough

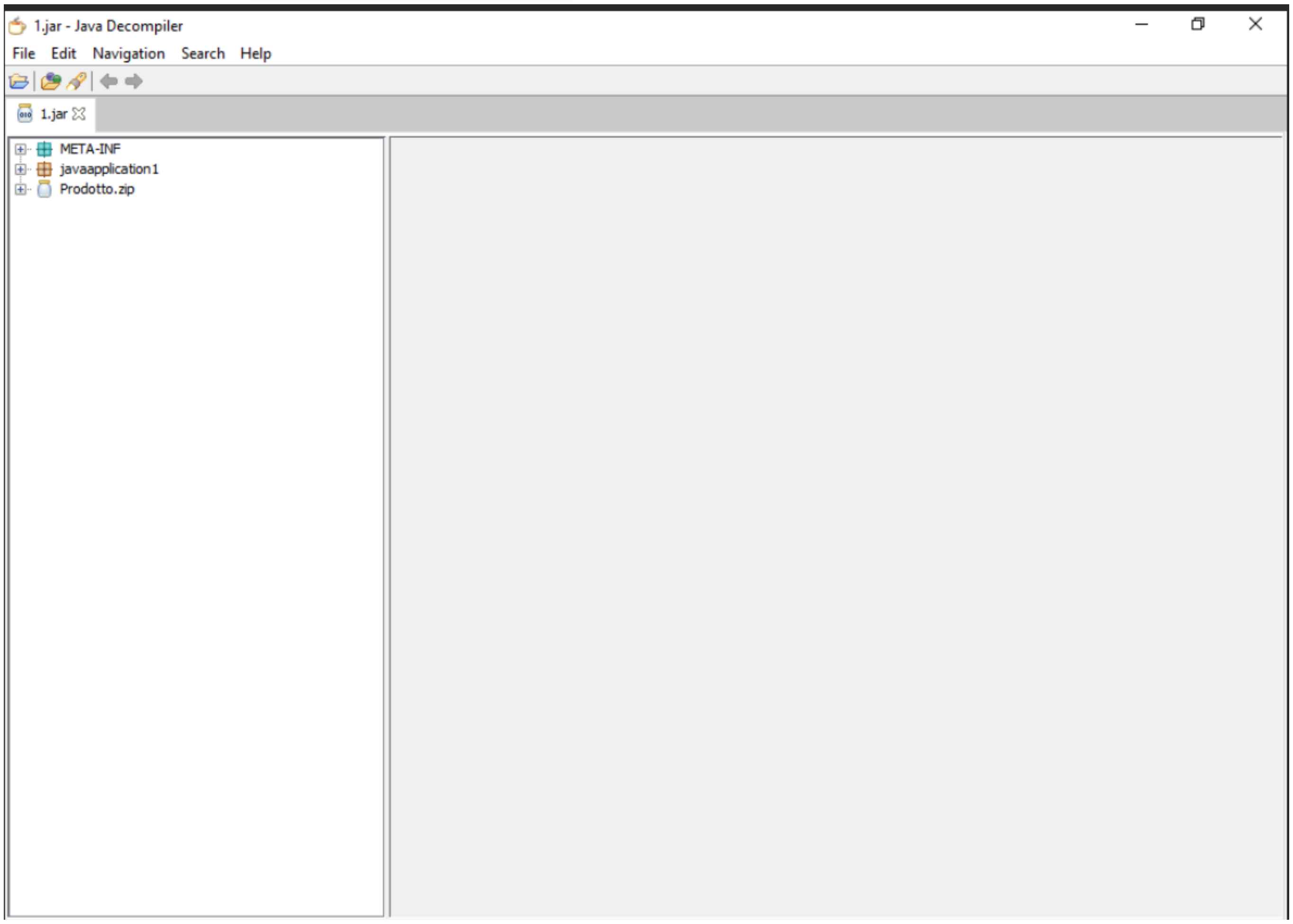
File hashsum

Firstly, need to get hash of the malware sample. And you too, if you want to examine sample closely through VirusTotal, AnyRun or, exctactly, `download` it from the Bazaar.

- powershell.exe `Get-FileHash .\filename.extension`
  Here it is:
  `SHA256 49BBFAC69CA7633414172EC07E996D0DABD3F7811F134EECAFE89ACB8D55B93A`

Jar opening

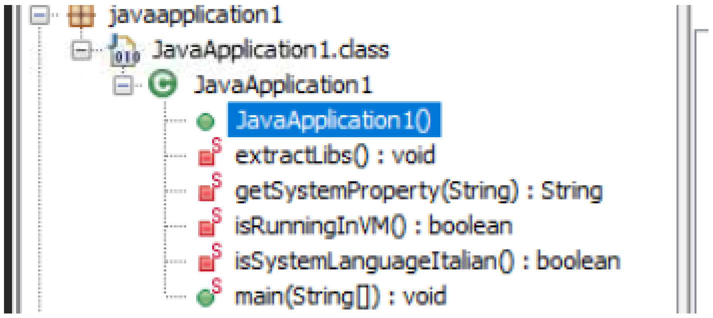
We have an .jar file - it is an Java Executable File. It is the compiled binary file, but we need look inside the code and resources. For that, we should use java-oriented decompiler named Java Decompiler (`jd-gui-windows-1.6.6`) and examine .jar with that tool



When file will opened, we will see three segments:

- META-INF - metadata of the file
- javaaplication1 - code segment
- Prodotto.zip - in this case it is an resource file - archive with the malicious file
  We are inrested to examine the source code.
  We can see ful code structure and included segments, such as:
- name of the code package
- functions with their return's types

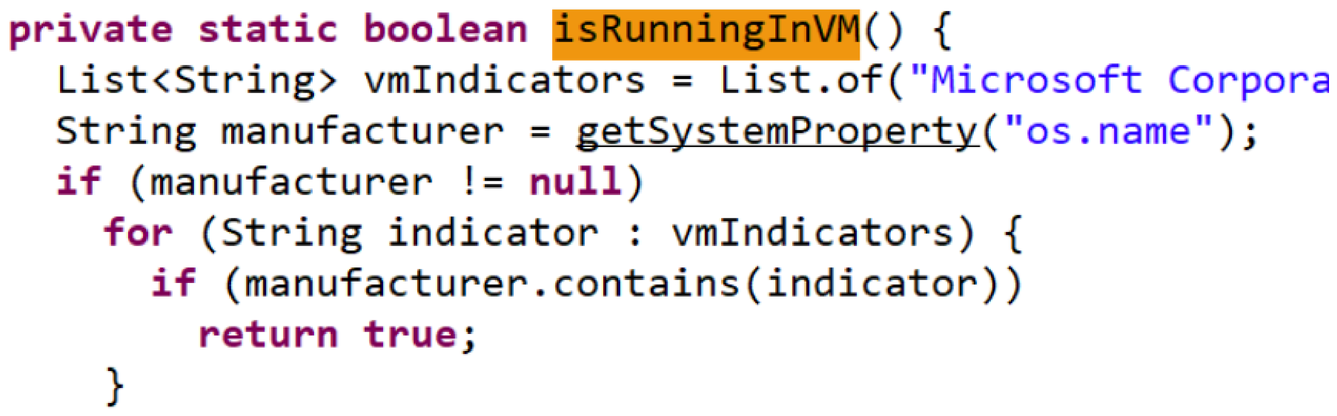
- main() function as entry point.



```
javaapplication1
  JavaApplication1.class
    JavaApplication1
      JavaApplication1()
      extractLibs() : void
      getSystemProperty(String) : String
      isRunningInVM() : boolean
      isSystemLanguageItalian() : boolean
      main(String[]) : void
```

Okay, then go to search the goals.

Firstly, we can understand which actions will be done by every methods cause we can see and read their names.
As we see, here is the method named `isRunningVM`. If we look closely, functions includes vm indicators list, request an os.name and look for entries. If there are - program runs on VM.

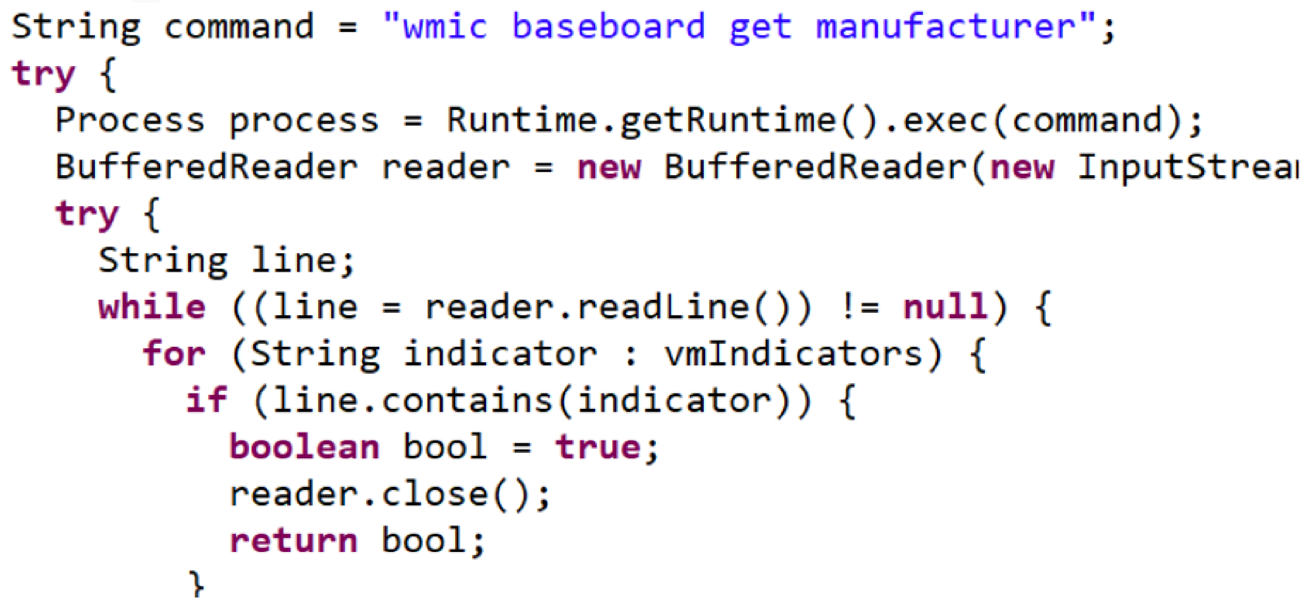


```
private static boolean isRunningInVM() {
  List<String> vmIndicators = List.of("Microsoft Corpora
  String manufacturer = getSystemProperty("os.name");
  if (manufacturer != null)
    for (String indicator : vmIndicators) {
      if (manufacturer.contains(indicator))
        return true;
    }
```

BUT
There is several our goals, cause during examining method, we can explain:

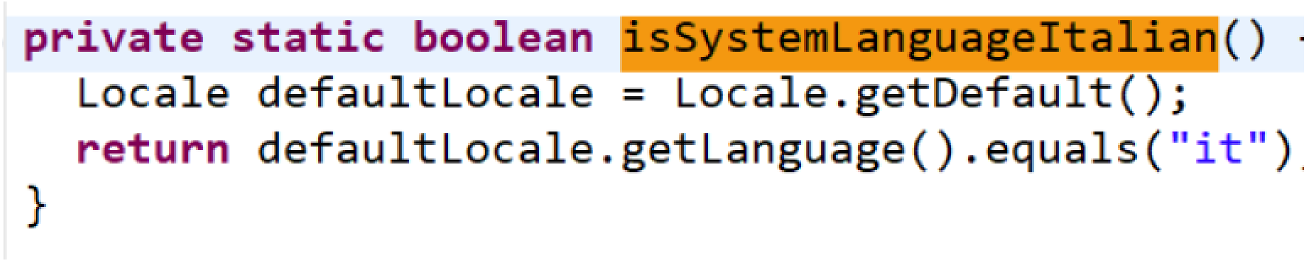
- goal #1 - `isRunningVM()`
- goal #7 - `wmic baseboard get manufacturer`

- goal #8 - `4` checked vendors



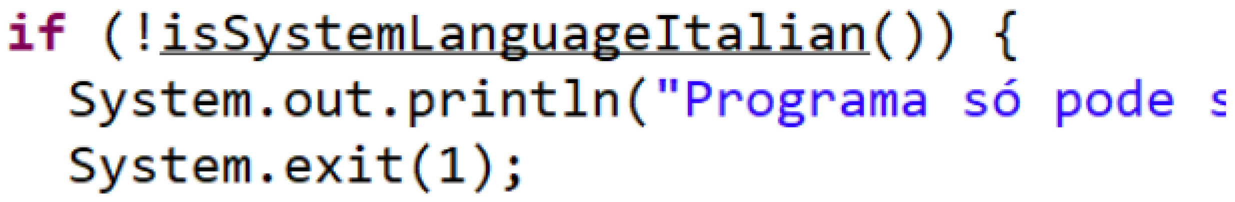
```
String command = "wmic baseboard get manufacturer";
try {
  Process process = Runtime.getRuntime().exec(command);
  BufferedReader reader = new BufferedReader(new InputStrea
  try {
    String line;
    while ((line = reader.readLine()) != null) {
      for (String indicator : vmIndicators) {
        if (line.contains(indicator)) {
          boolean bool = true;
          reader.close();
          return bool;
        }
```

Also we can see method named `isSystemLanguageItalian()`. Let's take a closer look:



```
private static boolean isSystemLanguageItalian()
  Locale defaultLocale = Locale.getDefault();
  return defaultLocale.getLanguage().equals("it")
}
```

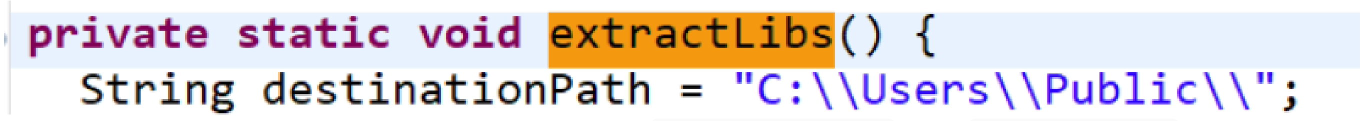
As it show, the return system language is italian, but to be sure to this is language is requested to running we should look for this method implementation and usind.
Go to `main()` and look for this method call:



```
if (!isSystemLanguageItalian()) {
  System.out.println("Programa só pode s
  System.exit(1);
```

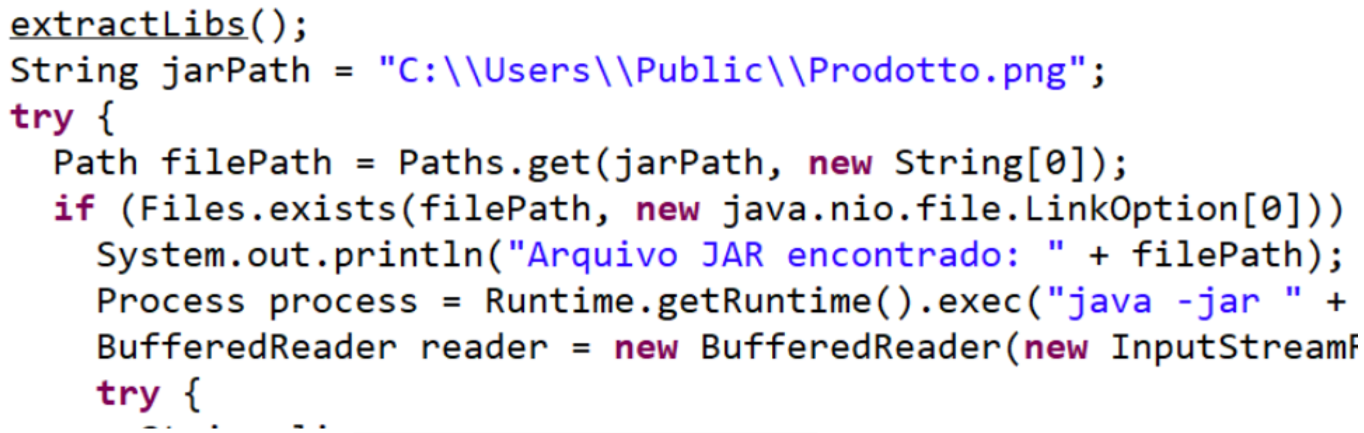
Yeah, here what we looked for. The method return the system language equality to `italian`, and in the main function there the `!method()` run-con condition. So, goal #2 is `italian`

Well, lets examine suspicious function named `extractLibs()`
Firstly we will see destpath



```
private static void extractLibs() {
  String destinationPath = "C:\\Users\\Public\\";
```

This method uses unpackaging functions like `zipInputStream` and `Prodotto.zip` as resource.
So, goal #3 is `extractLibs()` and goal #4 is `C:\Users\Public`
Lets continue to examine `main()`
After extraction, we have the extracted file path and the command to execute it



```
extractLibs();
String jarPath = "C:\\Users\\Public\\Prodotto.png";
try {
  Path filePath = Paths.get(jarPath, new String[0]);
  if (Files.exists(filePath, new java.nio.file.LinkOption[0]))
    System.out.println("Arquivo JAR encontrado: " + filePath);
    Process process = Runtime.getRuntime().exec("java -jar " +
    BufferedReader reader = new BufferedReader(new InputStreamR
    try {
```

And this is our goal #4 - `C:\Users\Public\Prodotto.png`

Also goal #6 is `java -jar C:\Users\Public\Prodotto.png`

Let's summarize all

1. isRunningVM()

2. Italian

3. extractLibs()

4. C:\Users\Public

5. Prodotto.png

6. java -jar C:\Users\Public\Prodotto.png

7. wmic baseboard get manufacturer

8. 4