Sean Ulrich & Matthew Austin

12/2/17

CIS 425

Dr. Tian

Project Report

**Tasking:**

The task of this project lab is to exercise and practice the process of carrying out a digital forensics investigation. It also serves as a demonstration of tools for digital forensics, such as Wireshark and JD-GUI. The java file analyzed was downloaded from cis425@mathcswc.fontbonne.edu.

**Forensic Questions:**

Can we discern a programs behavior and intentions from its source code?

Can we discern a programs behavior and intentions by watching it run in an isolated environment?

How can we make sure these findings are accurate?

**Steps Taken:**

1. Ensure Wireshark, bmon, iptraf, java, JD-GUI are installed
2. Check system and network traffic for their normal behavior state
3. Download GroupProject.jar file
4. Split into Static and Dynamic Analysis teams

*Static:*

1. Opened GroupProject.jar with JD-GUI program
2. Saw the Java Classes "MyKeylogger" implementing "NativeKeyListener", "Test" (that enable URL connection), and two more .jar files "httpclient" and "jnativehook".
3. Read through the and analyzed classes and package code
4. Document findings.

*Dynamic:*

1. Record network traffic at normal state with Wireshark, bmon, and iptraf
2. Run network monitoring tools as the .jar file is executed.
3. Analyze network traffic. Identify number of packets, protocols used, and IP addresses contacted. Identify port numbers used.
4. Observe other system behavior.
5. Document findings

**Results:**

Through Dynamic Analysis, it was determined that ‘GroupProject.jar’ behaves like a key logger – that is a program that records the keystrokes of the infected host and transmits them to the server of the original coder’s choice. While the program openly claims to be both a key logger and a mouse hook, dynamic analysis failed to find sufficient evidence of the latter. Whereas the network traffic of the Virtual Machine(VM) was initially silent, after the .jar file was executed, the system was flooded with a constant stream of packets from the host machine to a foreign IP server. The purpose of the program was plainly displayed in the console after the .jar file was executed and plainly displayed characters typed by the infected VM’s keyboard.

Through Static Analysis, after opening the 'GroupProject.jar' with JD-GUI, the classes "MyKeylogger" and "Test" along with package contents that allowed the Keylogger to do what it does as describe through Dynamic Analysis above. The first thing that shown in the .jar was that package content folder named "org.eclipse.jdt.internal.jarinjarloader" which had a contained a class called "JiJConstants" that contained various constant variables. 'JarRsrcLoader' loads up each class within the jar. 'RsrcURLConnection', 'RsrcUrlStreamHandler' and 'RsrcURLStreamHandlerFactory' allows jar to connection to internet.

Continuing with the Static Analysis, the 'MyKeylogger' class contains variables Test 'testObject' (an object from Test class), BufferReader 'br'. And PrintWriter 'evil'. With these variables it is able to make a constant connection to the internet using Test: '/etc/passwd' (keeps track of every user on system", 'etc/shadow' which holds password hashes. A temp file is created and keys that are pressed (nativeKeyPressed), released (nativeKeyReleased), and typed (nativeKeyTyped) are written to it and sent to the url each key press, release, and type. The Test class request and loads the url "<http://primetechconsult.com/mission.html>?" Where the keystroke logs are sent to.

Finally, the last two .jar files within 'GroupProject.jar' were 'httpclient' which contained the resources for the program to connect and keep a constant connection to the internet. 'jnativehook.jar contained the resources for Global keyboard and mouse listeners for the malicious programmer to use.

**Conclusion:**

Through Dynamic Analysis, it was only after ‘GroupProject.jar’ was executed that the host system began experiencing large amounts of overall network traffic. The data was being contentiously communicated with the foreign server with 107.180.57.91 - the IP address. The program uses port 53 to get DNS information and TCP connections are made and updates are sent to keep the connection alive; the TCP connection being established through somewhat random high number ports that aren’t frequently used. The entire TCP connection is reset every time a key is stroked on the keyboard and thus when that information is sent to the foreign IP – suggesting the act of entering a key ends a loop in the code.

Through Static Analysis, it was clear that the .jar file was a keylogger that created temporary files of keys and mouse presses. This information was sent via constant url connection to the url: "<http://primetechconsult.com/mission.html>?". To confirm the findings of the Dynamic Analysis, the program seemed to be nothing more than a simple keylogger.