Java Obfuscation SA

Zac Kologlu, z5257261.

Description

My something awesome project revolved around obfuscation in the Java ecosystem. I wanted to understand the interactions between compilation / decompilation, obfuscation / deobfuscation, and reverse engineering. Finally, for my project, I wanted to implement a working obfuscator - with the possible hopes of breaking existing decompilers.

Outcome

I was able to write a paper describing the concepts of the processes outlined above, and taking a deep dive into my project, including how it works and how different decompilers react to it. In my project, I successfully implemented a powerful obfuscator that is able to break almost all current Java decompilers (including all the most popular), and employs new obfuscation techniques invented by me. This included writing 400+ lines of raw JVM Bytecode (Java equivalent of assembly code).

The paper is available here:
https://github.com/insou22/sa-obf/blob/master/Obfuscation%20Paper.pdf

The obfuscator is available here: https://github.com/insou22/sa-obf

What I learned

This project taught me a lot about the inner workings of the Java Virtual Machine, and the Class-file format. I also learnt a lot about discrepancies between the Java language and the Java compiled format - most importantly, the discrepancies of things allowed in class files that aren't allowed in source files. Eg, your variable names can't have null-bytes in them (obviously), but the JVM doesn't care if a variable has a null-byte in a class-file. I also learnt more about writing bytecode from scratch, and increased my appreciation of working with extremely low level IL's

Evidence

My evidence is highlighted through: my github repo - containing commits with the code I have written for the obfuscator, my paper - demonstrating my aptitude that I have gained in the topics I chose to explore, and my screenshots - in the paper, demonstrating the obfuscator's behaviour, and how different industry decompilers reacted to my obfuscation.

Original proposal

Something Awesome project proposal:

The Java Virtual Machine (JVM) is a standardised virtual machine runtime for executing compiled Java binaries in the class file-format made up of assembly of the JVM bytecode instruction set. It is a tried and true virtual sandbox, trusted by a terrifyingly large proportion of high net-worth companies even today.

Being a compiled language, decompilation is always of interest to security researchers, attackers, defenders, all for their own individual reasons. Something to note of the class file-format, is that it is MUCH more easily reverse-engineered back to Java code compared to something like a clang / (god forbid,) gcc generated ELF file.

In order to combat this, an often used tactic in the Java world is automated obfuscation: as javac leaves lots of extra information in the class file-format, including line-markers, variable names, method / class names, etc., obfuscators can rip out lots of additional information and rearrange bytecode to become harder to decompile, but still behave the same.

For my Something Awesome project, I would like to take a deep dive into this area. Considering the technical requirements of COMP6841, I would like to create my own automated obfuscator / deobfuscator. This is an extremely technical process, as the class file-format is extremely detail-oriented, and attempting to programmatically edit these types of binaries is an exceedingly intricate process. As such, I have designed my marking criteria with this in mind:

- PS:
  Write a report on the topics covering and surrounding: [compilation, decompilation, obfuscation, deobfuscation, reverse-engineering]. This will include: [purpose, how it works, how it is vulnerable, examples]
- CR:
  The above report and an at least partially-working obfuscator with evidence of a significant amount of work, viewable on git.
- DN:
  A working obfuscator that is able to strip extra compilation information and perform basic obfuscation techniques to attempt to break decompilers, and either: the above report, or an at least partially-working deobfuscator with evidence of a significant amount of work, viewable on git.
- HD:
  A working obfuscator as of DN with some more interesting features, a working deobfuscator which is able to fix some common obfuscation techniques to prepare the binary for decompilation. (Note: deobfuscation is MUCH harder than obfuscation)

And approval:

Lachlan Jones
Mon 9/03/2020 3:37 PM
Zac Kologlu

Approved. Put it in an Openlearning blog post and send me a link so I can approve it publicly please.

Lachlan

Reflection on proposal

I believe that I have definitely made an obfuscator that is of a HD level, as it breaks almost all industry decompilers, and my final component would require specialised tools or large amounts of time from someone very experienced in Java bytecode and reverse engineering to deobfuscate.
Unfortunately, I was not able to make a deobfuscator as I severely underestimated the complexity of that project, so instead supplemented it with my report with a detailed overview of my obfuscator with examples. Such, I believe my Something Awesome sits between the DN / HD level.