Just-In-Time Malware Detection

Just-In-Time (JIT) Malware Detection is a novel technique that utilises dynamic binary optimisation to circumvent code obfuscation when analysing potential malware. We detail our early investigations into this technique, and present some initial results.

Malware is malicious software that, if executed, has some negative effect on the target machine – ranging from destructive ransomware to more subtle spyware. To prevent this, defenders must prevent malware from running, which in turn requires that they detect it. This results in an arms race: defenders analyse untrusted software to decide if it is safe, while attackers obfuscate their code to thwart this analysis.

Analysis can be static (e.g. examining the binary, or comparing hashes against a known virus database), or dynamic (e.g. by running the program in an isolated and monitored environment). Static analysis is easy to perform, and can be automated in antivirus software. However, code obfuscation can effectively counter this analysis by changing malware just enough to disrupt hash matching, or by making the control flow and behaviour tricky to follow. However, as the behaviour of the code becomes obvious once it executes, dynamic analysis bypasses this countermeasure. So dynamic analysis is potentially more useful, but requires a carefully managed environment so is not always practical.

We propose a potential hybrid approach using static analysis techniques (such as recognising particular patterns and behaviours common to malware) at runtime. As the malware executes, the obfuscated behaviour becomes easier to observe. If it matches these known patterns, the malware can be identified and stoped while it is executing, but before completion. This could, for example, stop ransomware after a few files are encrypted, saving the rest of the system.

This technique has particular potential in JIT compiler environments. Obfuscation can be framed as a "deoptimisation" that produces a more convoluted program with the same final result. Optimisation therefore has potential for undoing this and restoring the malware to a "straightforward" solution that is easier to analyse. With precompiled software, defenders do not normally have the opportunity to perform optimisations, but with JIT-compiled software dynamic binary optimisation at runtime is already part of the regular execution pipeline. As the optimisation work occurs regardless, there is a good opportunity to add just-in-time malware detection without changing the overall system architecture.

We outline how this technique works, including various approaches to fingerprinting suspicious applications at runtime, and present some initial results showing how obfuscation and dynamic binary optimisations changes these fingerprints.

This project is in early stages, and feedback is welcome. In particular, we are approaching this from a security/malware analysis perspective, so feedback and insights from a JIT compiler perspective would be particularly valuable.