Fake Emulation Environment to Prevent Malware from Executing

Collin Mulliner
Technische Universität Berlin

ABSTRACT

Today's malware contains sophisticated analysis countermeasures to protect itself against reverse engineering. Countermeasures fall into two categories: offline and runtime. Encryption and obfuscation of binaries are widely used offline protections. Therefore today, most analysis is done during runtime and so malware authors implement runtime countermeasures. Runtime countermeasures include anti-debugging techniques and detection of emulation and analysis environments. Upon detecting an emulation environment, said malware behaves normally and does not execute its malicious payload; often, it even goes into an idle state.

The specific countermeasure in which we are interested is the detection of analysis and emulation environments. There exist different methods for detecting such environments by abusing hardware specifics such as CPU bugs or timing issues [2] and by detecting leftovers in the operating systems such as driver and file names [1].

The main idea of this work is confusing the countermeasure code implemented by malware through faking parts of an emulation environment, in the hope that this makes it harder for malware authors to figure out how to really detect analysis environments. The whole malware/anti-malware situation is an arms race with each side always trying to get a head start. The state of the art is to try to make analysis and emulation environments as close as possible to real systems to prevent detection. This idea tries to do the opposite by sprinkling small *fake traces* of system emulators into real systems. The chief challenge for implementing such a system is that fake traces have to be carefully designed to not be easily detectable as fake and to not have an negative impact on the actual system.

BODY

Faking just enough of common emulation and analysis environments might prevent certain kinds malware from executing their malicious payload.

REFERENCES

[1] D. O'Neill. How to detect virtualization. `http://www.dmo.ca/blog/detecting-virtualization-on-linux/`, February 2010.

[2] T. Raffetseder, C. Krügel, and E. Kirda. Detecting System Emulators. In *ISC*, pages 1–18, 2007.

Volume 1 of Tiny Transactions on Computer Science