Dynamic Analysis: Knowing When to Stop

Paul Irofti
paul@irofti.net

Challenges in Analysing Executables:
Scalability, Self-Modifying Code and Synergy
Dagstuhl Seminar, 2014

Outline

Who Am I?

Reverse Engineer (6 years in the AV industry)

- anti-virus engines
- emulators: static and dynamic analysis research

OpenBSD Hacker:

- power management, ACPI
- mips64: Loongson and Octeon
- `compat_linux(8)` maintainer
- porter

Research Assistant and PhD student:

- Faculty of Automatic Control and Computers at the Polytechnic University of Bucharest
- PhD on parallel signal processing algorithms using GPGPU (OpenCL, CUDA)

Reverse Engineer

- Project Lead: AntiMalware Emulator Implementation
- JIT support
- IEEE 754 Floating Point Support
- API Emulation
- talked about it at "Analysis of Executables: Benefits and Challenges"
- Static Antivirus Engines Development
- Themida, SVKP, VMProtect, tELock etc.

Description

Context:

- in production (ex. mail server, end-user)
- multiple and very different samples run through one emulator
- each sample takes different paths through the emulator
- some samples take too long
- after a point a sample is deemed unacceptable for emulation
- passed that threshold the emulation is forced to stop

Stopping

Existing thresholds are based on:

- elapsed time
- number of emulated instructions

Time-based

Pros:

- intuitive
- easy to implement
- always there, at least as a watchdog

Cons:

- varies depending on CPU power
- can give false-positives due to low platform performance
- hard to find a good average
- non-deterministic

Instruction-based

Pros:

- deterministic
- reports from the field are easier to debug

Cons:

- not all instructions are equal
- time needed to process k-emulated instructions varies
- without a time-based watchdog it can hog the CPU
- hard to find a good average
- premature stops can lead to false verdicts

Goal

A deterministic way of stopping the emulation process in due time

- reproduceable results
- pin-pointing where the emulation stopped
- good on all platforms

Analysis

Setup:

- spot the important nodes in the dynamic analyzer
- add counters in these key positions
- run the emulator through lots of varied samples
- store the execution time and the final counter values

The Results

We should have:

- a tuple of n counters per sample
- a total of m samples
- with m corresponding execution times t
- and with $m \gg n$

Counter Weights (1)

With this data we are able to weigh each counter

For one sample:

$$t = \begin{pmatrix} c_1 & c_2 & c_3 & \dots & c_n \end{pmatrix} \times \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix}$$

Counter Weights (2)

For all *m* samples:

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_m \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & c_{2,3} & \cdots & c_{2,n} \\ c_{3,1} & c_{3,2} & c_{3,3} & \cdots & c_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & c_{m,3} & \cdots & c_{m,n} \end{pmatrix} \times \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix}$$

Counter Weights (3)

The resulting overdetermined system

$T^m = C^{m \times n} \times W^n$, with $m \gg n$

can be solved through least squares, SVD etc.

Properties

- fast start-up: small set of counters is good enough
- easy adaptation through counter addition/removal
- a sort of automated O calculator
- a good profiling tool

Metrics

For lack of a better word, we name the weight values metrics.

Definition

The speed of a platform is measured as metrics per second

Deterministic Threshold

We can now build a deterministic threshold:

- compute **only once** an average platform speed
- set a metric threshold based on the average speed
- if a process was stopped we know exactly where
- we also get an implicit time threshold for free

Example

Average speed of 50 m/s, set the threshold to 150 m, it results in a 3 s maximum emulation time per sample.

Mostly Harmless

Conclusions

- fair on all platforms, different speeds for different machines
- easier to reproduce reports and samples from the field
- determinism and time thresholding at once
- the limit can be easily bumped at runtime
- weight calculation is machine indepenedent (done in the lab)
- adding / removing code affects the weight system
- running a thorough analysis can be time consuming
- fresh calculations should be done per release, not per commit

Future Directions

Continuing research in the field (moving to academia)

- looking for ways of improvement or different approaches
- investigating different means of average speed calculation

Writing an article about the metrics method

So Long, and Thanks for All the Fish

Questions?