Higher-Order Detection: Making Trace to Be Resilient to Obfuscation

Thanh Dinh Ta
Advisors: Jean-Yves Marion and Guillaume Bonfante

CARTE Team - INRIA Nancy

Saturday 25th May, 2013

Detection and Obfuscation (Co-)Evolution

- Detection techniques:
  - Syntactic based (classic, mostly used):

    $$\mathrm{a.b.}m_1.m_2\mathrm{.c.d}$$

  - Behavior based (modern, sometimes used):

    `NtOpenFile.NtOpenKey.NtSetValueKey.NtWriteFile`

Other supports: semantics template (for syntactic detection), abstract trace (for behavior detection), statistical data mining, etc.

Detection and Obfuscation (Co-)Evolution

- Obfuscation techniques:
  - (Poly,meta)-morphic:

$$\mathrm{a}.m'_1.\mathrm{jmp}\ m'_2.\mathrm{b}.\mathrm{c}.m'_2.\mathrm{d}$$

  - (Cryptographic,virtual instruction)-packer:

$$\text{messy data} \xrightarrow{unpack} \mathrm{a}.\mathrm{b}.m_1.m_2.\mathrm{c}.\mathrm{d}$$

Obfuscation is some potent transformation $t : \mathbb{P} \to \mathbb{P}$ that:

- preserve the I/O semantics: $[\![P]\!] \approx_{IO} [\![t(P)]\!]$
- but: $\mathtt{trace}(P) \neq \mathtt{trace}(t(P))$

for all program $P \in \mathbb{P}$.

Detection and Obfuscation (Co-)Evolution

They defeat each other, from time to time.

$$\texttt{syntactic} \xrightarrow{\textit{defeated by}} \texttt{polymorphic} \xrightarrow{\textit{defeated by}} \textit{behavior} \ldots$$

Problem

Given a (well-know) malware P, let Q be an unknown program. How can we estimate that Q is just an version of P, i.e.

$$Q = t(P)$$

for some potent transformation t ?.

Higher-Order Detection

Suppose that $Q = t(P)$ for some t, so

$$[\![P]\!] \approx_{IO} [\![Q]\!]$$

In other words, P and Q represent the same function but are implemented by different algorithms.

Higher-Order Detection

Some observations:

- Let tr_1^P and tr_1^Q are corresponding traces of P and Q with some input i_1, then normally:

  $$tr_1^P \neq tr_1^Q$$

  since P and Q are implemented by different algorithms.
- Given another input i_2 with $tr_2^P = tr_1^P$, then probably:

  $$tr_2^Q = tr_1^Q$$

  since P and Q represent the same function.

Higher-Order Detection

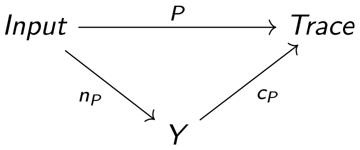
Main idea

We do not compare directly the traces of P and Q, but the trace-based equivalence relation $\approx_P$ and $\approx_Q$ on the set of inputs.

$$i \approx_P j \iff tr_i^P = tr_j^P$$
$$i \approx_Q j \iff tr_i^Q = tr_j^Q$$

Higher-Order Detection

Categorically, any program P is represented uniquely by



where n_p is called normalization function and mono-morphism c_P is called choice function.

We say that Q is an obfuscated version of P if $n_Q = n_P$.

Higher-Order Detection

Extracting n_P is hard since the traces are sophisticated,

I have prepared a truly impressive figure for that but this page is too narrow to contain. Pierre de Fermat (1637)[1].

and that is my current works.

[1] He would have said that if he saw the trace.

Your questions ?