BlueTracer:
a Robust API Tracer for Evasive Malware

Simone Nicchi

Thesis Advisor: Prof. Camil Demetrescu
Thesis Co-Advisors: Dr. Daniele Cono D'Elia, Dr. Emilio Coppa

Master of Science in Engineering in Computer Science

July 20, 2018

Malware Analysis

Malware is an ever-growing threat

- **Static Analysis:**
  involves the inspection of the different data and code sections
  of a binary

- **Dynamic Analysis:**
  a malware sample is executed and the actions it performs on
  the environment are observed

Dynamic analysis strongly favoured as it allows to dodge most code
obfuscations and deal with a large number of samples

Function call monitoring

- Functions can abstract implementation details providing a semantically richer representation of some functionality
- The abstractions embodied by **system calls** and **library functions** can be used to grasp the visible behavior of a malicious sample

Example:

```
RegCreateKey ("...\CurrentVersion\Run\monitor")
CreateDirectory ("C:\Windows\utils")
CreateFile ("C\Windows\utils\GFypmMVqJQOEQqy.exe")
```

Problem 1: limited logging capabilities

Available API tracing tools have **limited logging capabilities**

```
record_before(libcall_name, arg1)
retval = libcall(arg1, &arg2)
record_after(retval, *arg2)
```
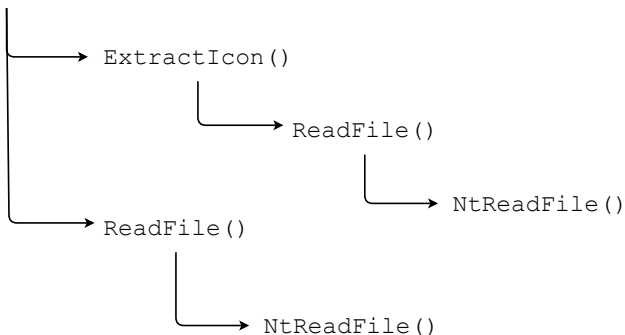
- Access to calls related information is required
    - Prototype (number of arguments, data types, input/output)
    - Need to know their run-time locations

- **Challenge**: heterogeneity of Windows libraries used in malware and lack of well-structured documentation for their prototypes

Problem 2: logging only calls made by sample

It is hard to distinguish the calls made directly by the sample from the ones made within libraries

- Resulting logs are large and contain irrelevant information

```
malwareFun()
                ┌──► ExtractIcon()
                │              │
                │              └──► ReadFile()
                │                          │
                │                          └──► NtReadFile()
                │
                └──► ReadFile()
                            │
                            └──► NtReadFile()
```

Problem 3: evasive malware

Evasive malware

Malware that conceals its harmful behaviour when detecting a hostile environment, such as a well-known sandbox solution

Is environment hostile?

?

Yes / No

Malicious behaviour exhibited

or

Malicious behaviour not exhibited
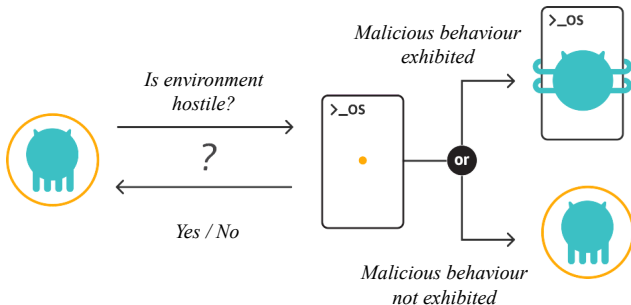
Current tracing tools are **easily detectable** and are not coupled with mechanisms to hide their presence

BlueTracer: accurate API logging

> **BlueTracer** is a robust library and system call tracer for Windows programs specialized in evasive malware based on Intel Pin

The tool possesses a remarkable logging power:

- Integration of rich, reliable external sources
  (**Dr. Memory** and **CISCO PyREBox**)
- Stealthy tracing of input parameters, output buffers and return values of over 17,000 system calls and library calls
- Logging of asynchronous events

BlueTracer: robust against evasive malware

Solution to the detection problem:

- Seamless integration with the **BluePill** stealthy execution framework

Simone Nicchi BlueTracer: a Robust API Tracer for Evasive Malware
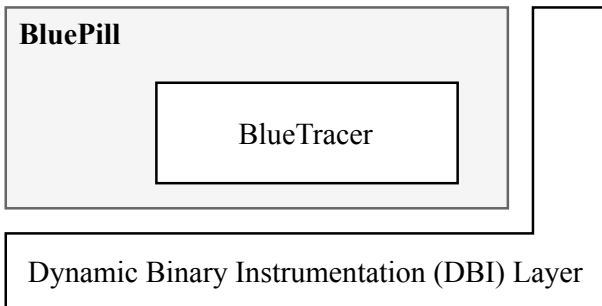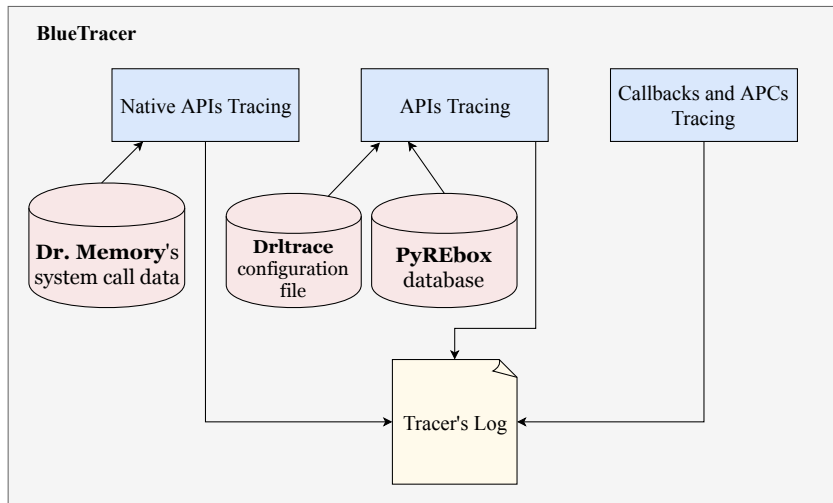
BlueTracer: focussed tracing of the sample's actions

Only calls made directly by the sample are recorded through the use of context-sensitive introspection

- Stack pointer and return address analysis
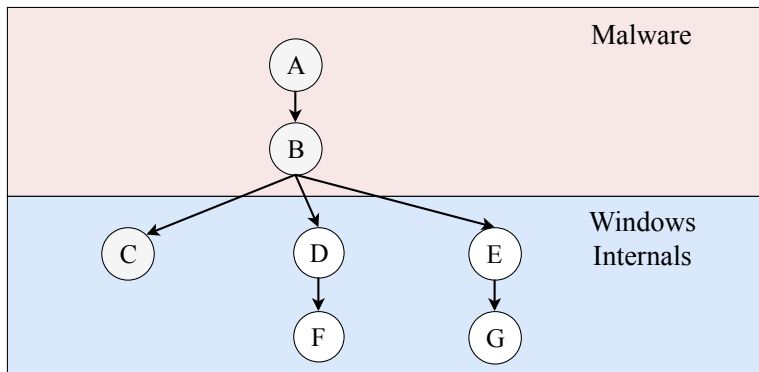
Architecture of BlueTracer

Validation (1): Al-Khaser

Al-Khaser is an open-source application which performs checks popular in malware samples spotted in the wild to determine if they are being executed in an analysis environment.

Many implemented techniques:

- **Anti-Debugging**
- **Timing-based**
- **Human Interaction Detection**
- **Anti-Virtualization**
- **Detection of Analysis Tools**

BlueTracer:

- remained undetected thanks to its integration with BluePill
- managed to track all arguments used to trigger evasion

Validation (2) with highly evasive malware samples

Five highly evasive samples collected by Joe Security:

ID	MD5	Name
1	0af4ef5069f47a371a0caf22ae2006a6	*banker*
2	9437eabf2fe5d32101e3fbf9f6027880	*dropper*
3	cbdda646a20d95f078393506ecdc0796	*trojan*
4	cfdd16225e67471f5ef54cab9b3a5558	Olympic
5	ef694b89ad7addb9a16bb6f26f1efaf7	Malicious CCleaner

Manual validation of all relevant actions performed by each sample:

- Check if logs are congruous with Joe Security reports
- Process Monitor as ground truth for system activity

The logs collected by BlueTracer reveal behaviors consistent with the analysis reports authored by Joe Security

Example of tracked malevolent action

Tracing a particular action of a malware instance allows to understand in detail what the sample's intentions are

Example: dropping a malicious executable

```
~~1116~~ 138 kernel32.dll!CopyFileA
138  arg 0: C:\Users\Simuset\Desktop\sample1.exe
(name=lpExistingFileName, type=char*, size=0x1)
138  arg 1: C:\Windows\system32\†\ffpb6966.exe
(name=lpNewFileName, type=char*, size=0x1)
138  arg 2: 0x0 (name=bFailIfExists, type=(long/int), size=0x4)
138    executed kernel32.dll!CopyFileA =>
138  retval:0x1 (name=Return value, type=(long/int), size=0x4)
```

Conclusions

Contribution:

Design and implementation of **BlueTracer**, a robust library and system call tracer for Windows programs specialized in evasive malware.

Future Developments:

- Automatic methodology for large-scale evaluation
- Improvement of logging capabilities
- Log filtering and aggregation techniques

Thank you for your attention!