Memory Tracing - Forensic Reverse Engineering

Endre Bangerter

Security Engineering Lab / RISIS
http://sel.bfh.ch
Bern University of Applied Sciences

Bern University
of Applied Sciences

Joint work

- Current and former team members at the Security Engineering Lab
  - Jonas Wagner
  - Dominic Fischer
  - Damien Schaefer
  - Thomas Ender

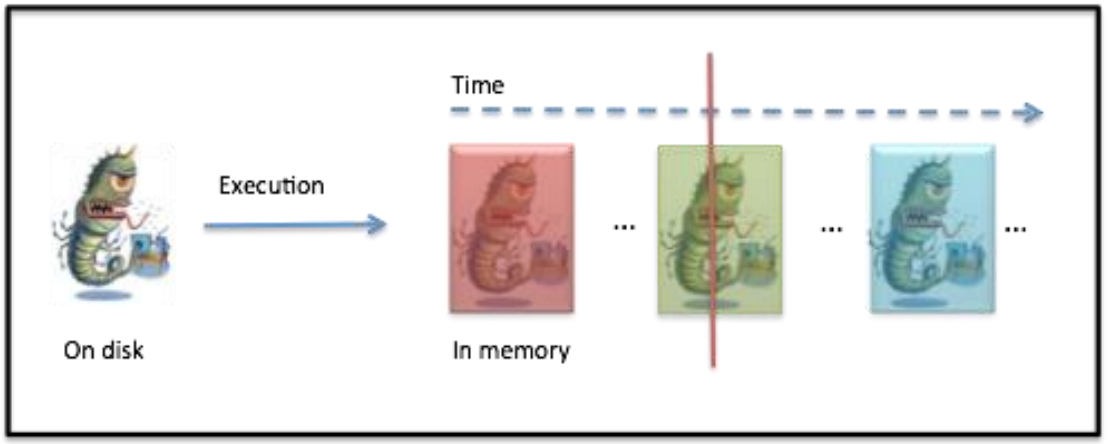

Background
&
Idea

Intrusions, malware, memory forensics

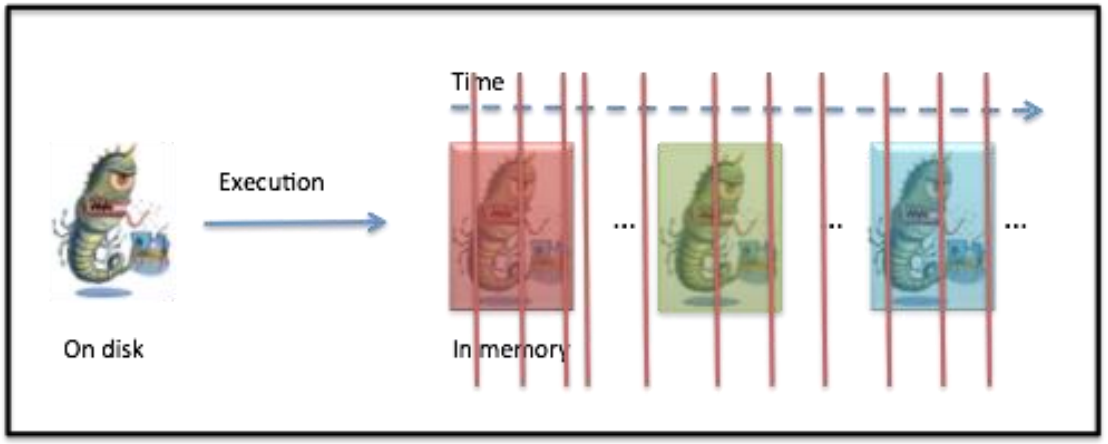
- Intrusions at some stage typically make use of malware/tools
- Detection and analysis

- **“**When it runs it is in memory”, memory forensics very effective to detect and partially analyze malware
- **Quest for automation and better analysis techniques, since humans don’t scale**

Memory tracing



“Traditional” memory forensics



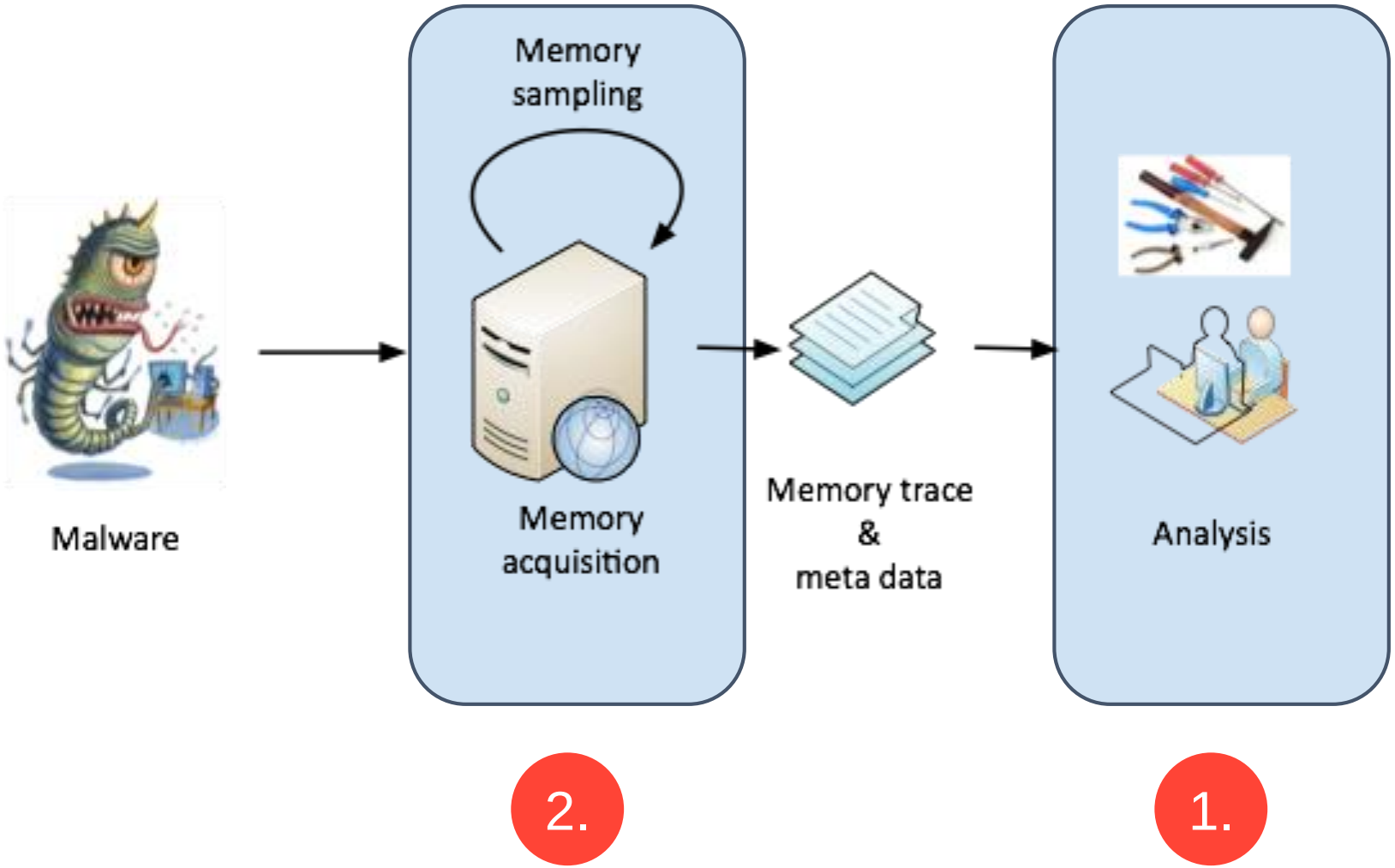
Memory tracing

- series of memory snapshots

Memory tracing, why potentially good?!?

- **Intuition**
  - Comprehensive capture of system behavior
  - Captures **transient** memory contents (i.e., short lived data & code)
    - Obfuscated data & code / self modifying code
    - Cypto keys & buffers
    - Networks buffers, URLs, config data, passwords…
- **We'll show**
  - Can be used for analysing malware, **automate *some* aspects** of malware analysis
  - **Guide analysts quickly to interesting memory regions,** for further manual analysis

The system perspective



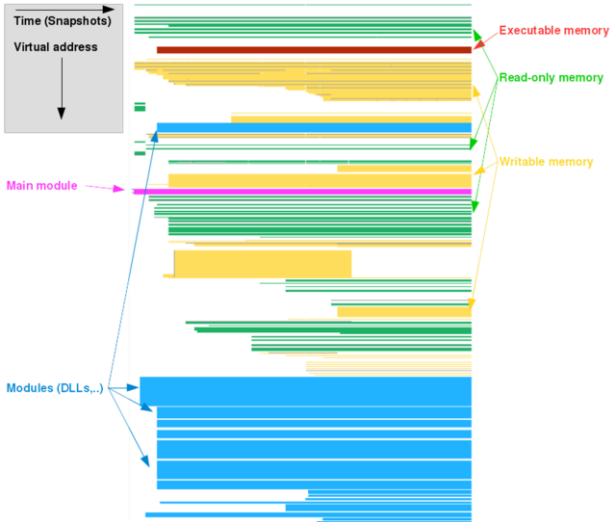
Analysis

Analysis

- Main idea is to **understand temporal behavior of memory** and in particular **modifications & anomalies that are characteristic for malware**

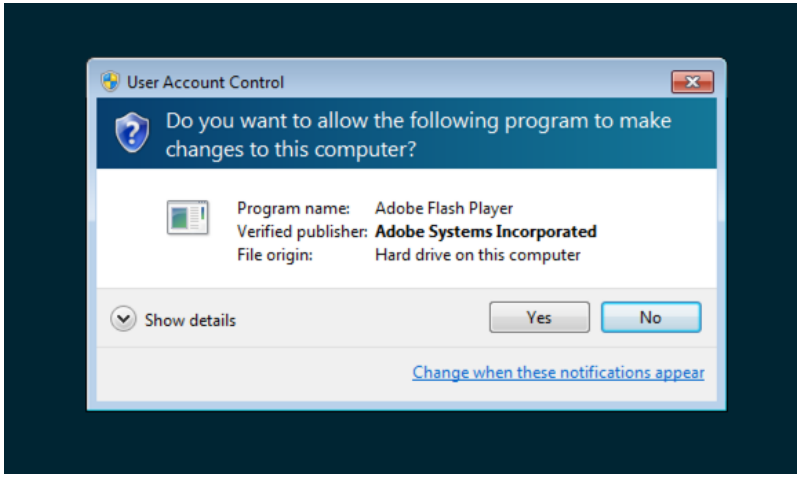
- **“Big data”** - Memory traces contain vast amount of information
  - Visualization
  - Heavy computation

- **Various analysis techniques**
  - Reconstruction of system call like events
  - Code modification detection (whitelisting)
  - Code correlation / matching across processes
  - Detection of self modifying code
  - …



Malware analysis demo - Zeroaccess

- P2P bot
- What user sees upon infection



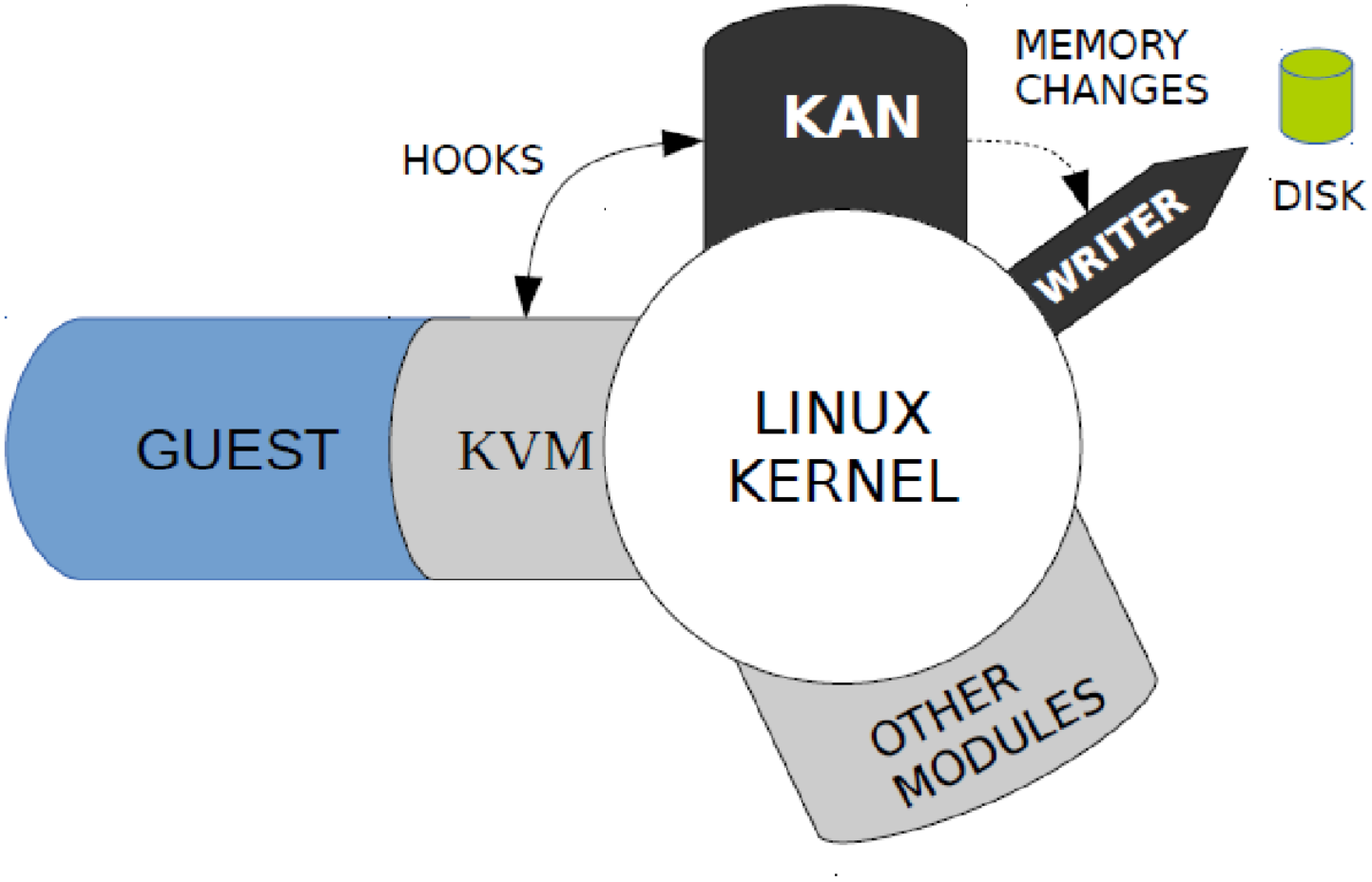
- Analysis demo will be shown in video (live demo takes too much time…)

Malware analysis demo - Zeroaccess

- Malware analysis using memory traces works
- Quick understanding of installation behavior, strong indicators for malicious behavior
- **Seamless transition to follow up analysis**
  - Guide to interesting code locations
  - Code can be easily extracted and analyzed further
  - Integrates wit existing analysis toolchain

Memory trace acquisition

Architecture memory tracing engine



Triggering – when to take memory snapshots

- Temporal trigger
  - e.g. Snapshot every 20ms
  - ...and when new process is created

- System call trigger

```
snapshot_before = NtWriteFile NtWriteVirtualMemory NtCreateThread NtFreeVirtualMemory
snapshot_after = NtAllocateVirtualMemory NtDeviceIoControlFile NtProtectVirtualMemory
NtReadFile NtWriteVirtualMemory NtReadVirtualMemory NtFlushInstructionCache
```

- Choice of triggers matters a lot!

Memory tracing engine - properties

- **Performance**
  - depends on triggering frequency and processes running
  - system under acquisition can be used interactively (typically)
- **Operating system independent**
  - Whatever runs under KVM is fine
  - In particular, Linux, Windows
  - 16bit / 32bit / 64bit
- **“Relatively” stealthy**
  - As stealthy as KVM
  - Minimal guest instrumentation (just the syscall trigger)
  - Other timing characteristics than plain KVM

Conclusion & Outlook

What is it?

- **Novel malware analysis technique**

- **Bridges gap between automated and manual analysis**, currently automates some analysis and points analyst to interesting code
  - mix between dynamic and static analysis basically
  - in-between traditional sandboxes and instruction tracers (QEMU, Pin…)

- **“Forensic reverse engineering” - record once and re-analyze paradigm**
  - Can go back to an analysis month later, e.g., to correlate recent with past attacks

Outlook

- **Evolve towards practically usable reverse engineering tool** for individual malware analysis & reverse engineering, not mass analysis

- **Adding more unique automated analysis features**