

A Survey on Virtual Machines for Malware Analysis

Daniel Reynaud, PhD Student
reynauddd@loria.fr

A Survey on the use of Virtual Machines for Malware Analysis

Daniel Reynaud, PhD Student
reynauddd@loria.fr

A Survey on the use of Virtualisation and Emulation Techniques for Malware Analysis

Daniel Reynaud, PhD Student
reynaud@loria.fr

Virtualisation and Emulation Techniques for Malware Analysis

Definitions

Emulation :

Complete software implementation of an execution environment, including the processor. Therefore every instruction is interpreted by the emulation software, not directly by hardware.

(ex: Bochs, QEMU, emulators in AV products)

Virtualisation :

Virtualisation software only interprets privileged instructions of guest applications, while the other instructions are run directly on hardware.

(ex: VMWare, Xen, Parallels, VirtualPC...)

Plan

1. Benefits of virtual machines

- dynamic malware analysis
- sandboxed environment
- low-level control
- automated analysis

2. Problems associated with virtual machines

- detection of virtualisation
- monitoring the malware
- detecting logic bombs

Conclusion

Dynamic Malware Analysis

- Entry Point Obscuring, obfuscation, anti-debugger techniques prevent efficient automated static analysis of malware
- manual analysis can be made very complex, and there are just too many samples to analyse

References

Filiol, Beaucamps - *On the possibility of practically obfuscating programs towards a unified perspective of code protection*

Collberg et al. - *Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs*

1. Benefits of Virtual Machines

Sandboxed Environment

- no need to use a dedicated machine for malware analysis
- quick and easy to restore the virtual machine to its initial state
- ... but there might be ways to jump out of the sandbox !

References

Ormandy - An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments

Virtualisation and Emulation Techniques for Malware Analysis

1. Benefits of Virtual Machines

Low-Level Control

- virtual machines provide a way to escape the cat and mouse game of userland monitoring
- access to emulated hardware state directly (particularly memory)
- emulation also grants access to individual CPU instructions for analysis (dynamic disassembling)

References

Carrera - Python Debugging Outside the "Bochs"

Virtualisation and Emulation Techniques for Malware Analysis

1. Benefits of Virtual Machines

Automated Analysis

- straightforward to launch a program in a virtual machine, log its activity and then restore the virtual machine to its initial state automatically

- several automated analysis services have appeared online :

<http://anubis.iseclab.org/>

<http://www.cwsandbox.org/>

<http://www.norman.com/microsites/nsic/Submit/>

<http://www.joebox.org/>

References

Willems et al. - *Toward Automated Dynamic Malware Analysis Using CWSandbox*

Bayer et al. - *Dynamic Analysis of Malicious Code*

Plan

1. Benefits of virtual machines

- dynamic malware analysis
- sandboxed environment
- low-level control
- automated analysis

2. Problems associated with virtual machines

- detection of virtualisation
- monitoring the malware
- detecting logic bombs

Conclusion

Detection of Virtualisation

- modern CPUs are extremely complex, therefore perfect emulation is not achievable in practice
- presence of covert channels in current virtualisation technologies
- the Intel Pentium is currently not virtualisable because 17 sensitive instructions are not privileged (including the SIDT instruction used in RedPill)
- subtle timing issues

References

Raffetseder, Kruegel, Kirda – *Detecting System Emulators*

Robert, Irvine - *Intel Pentium's Ability to Support a Secure Virtual Machine Monitor*

Ferrie - *Attacks on More Virtual Machine Emulators*

Monitoring Malware Activity

- user mode Win32 hooking is neither accurate nor stealthy. Kernel mode Win32 hooking is accurate but not stealthy.
- virtualisation technologies don't really provide better monitoring capabilities
- emulation technologies provide an exit to this cat and mouse game, including individual CPU instructions monitoring

References

Josse – *Secure and advanced unpacking using computer emulation*

Detecting Logic Bombs

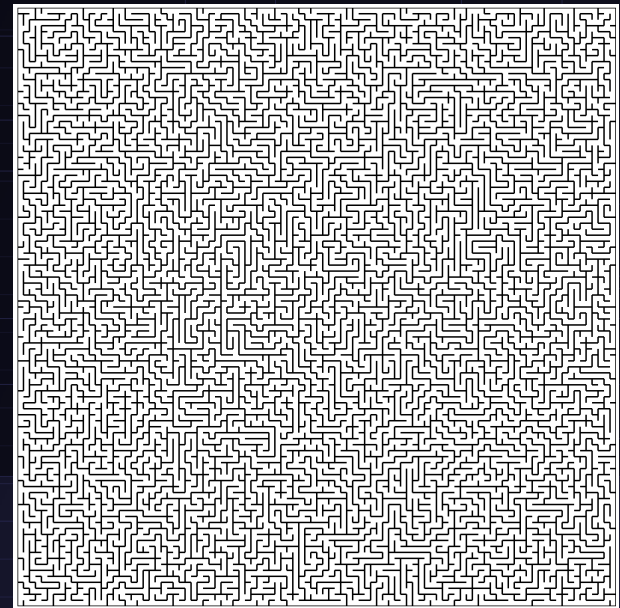
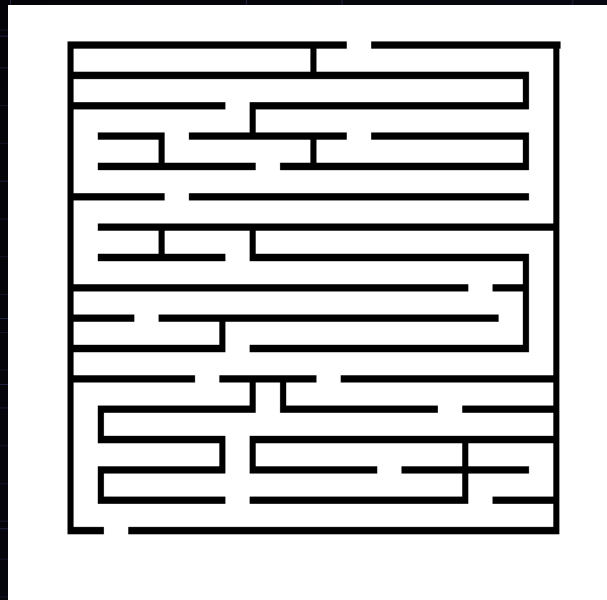
- a logic bomb is any program that deliberately delays its malicious activity for a certain time, until a certain date or a particular event
- it is possible to force the malware into exploring multiple execution paths, but (CLAIM) the problem is intractable in practice...

References

Moser, Kruegel, Kirda – *Exploring Multiple Execution Paths for Malware Analysis*

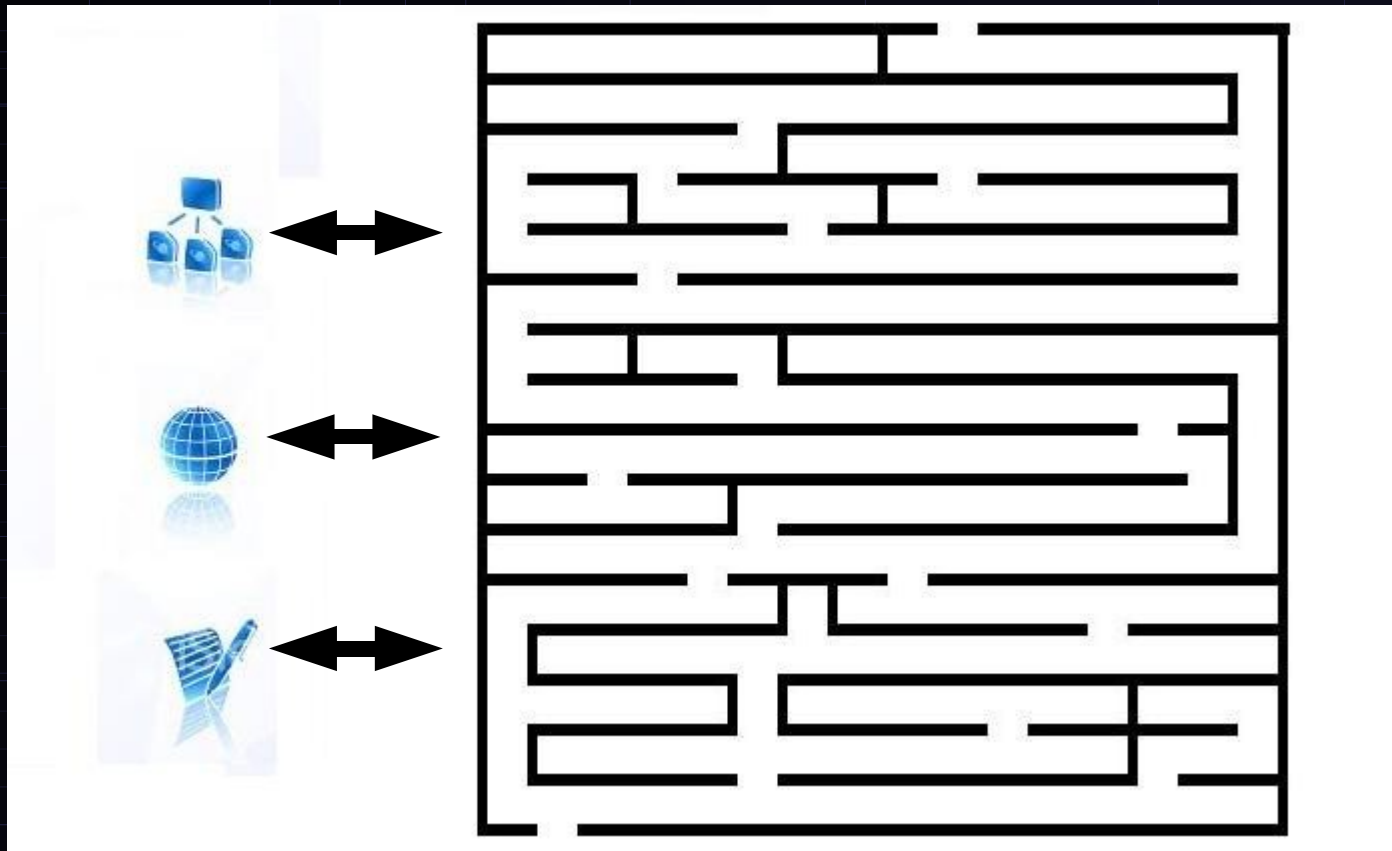
Detecting Logic Bombs – Exploring Multiple Paths

- detection can be delayed by dead code insertion
- worst-case complexity is exponential in the number of conditional branches



Detecting Logic Bombs – Exploring Multiple Paths

- everything does not happen in memory (I/O problems)



Detecting Logic Bombs

- execution prevention by non-linear dependencies insertion

(ex:

```
if (date == « 15th May 2008 » { ... } → easy to set date
```

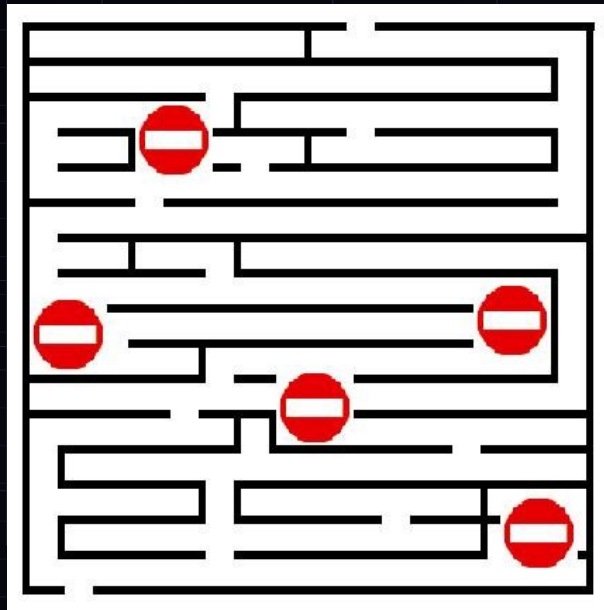
Detecting Logic Bombs

- execution prevention by non-linear dependencies insertion

(ex:

`if(date == « 15th May 2008 » {...}` → easy to set date

`if(hash(date) == 0x781A0FB7) {...}` → can't set date
(might lead to inconsistent states)



Plan

1. Benefits of virtual machines

- dynamic malware analysis
- sandboxed environment
- low-level control
- automated analysis

2. Problems associated with virtual machines

- detection of virtualisation
- monitoring the malware
- detecting logic bombs

Conclusion

Virtual Machines are not the Solution

- they allow virtually **perfect monitoring** of program behaviour
- complete software emulators are virtually **undetectable**
- but **logic bombs are inherently hard to detect** due to the offline nature of the monitoring that virtual machines perform