Black Hat Europe 2014

Freeze Drying for Capturing Environment-Sensitive Malware Alive

FFRI, Inc.
http://www.ffri.jp

Ver 2.00.01

Contents

Background

- Sophisticated malware arms many anti-analyze techniques
  - using targeted attacks, cyber espionages, banking malware

- First, we need protection
- Second, we are curious about true intention

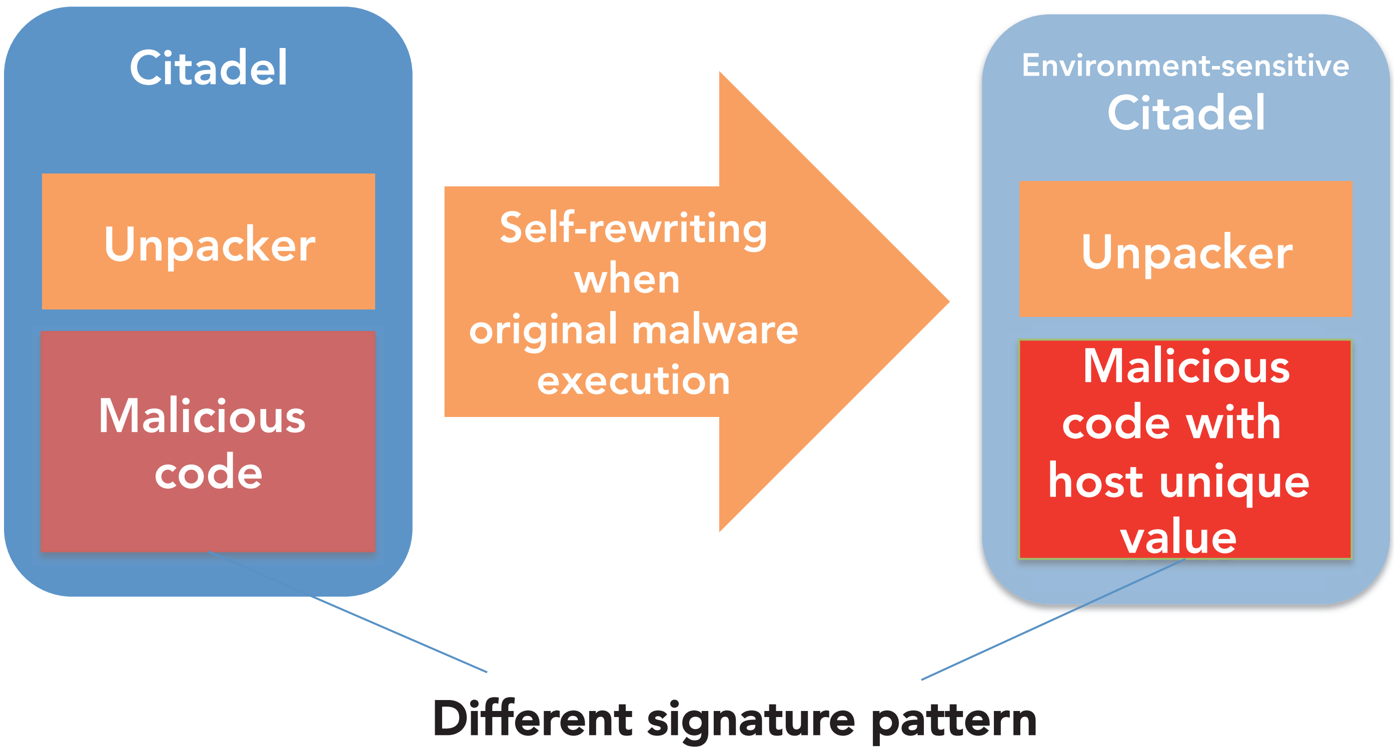
Case study: Citadel

- Some citadel detects the execution environment and do not engage in malicious behavior when the current host differs from the infected host[1]
  - To avoid behavior-based malware detection(like sandbox analysis)

- Showing 2 examples
  - Host-fingerprinting
  - VM/Sandbox detection

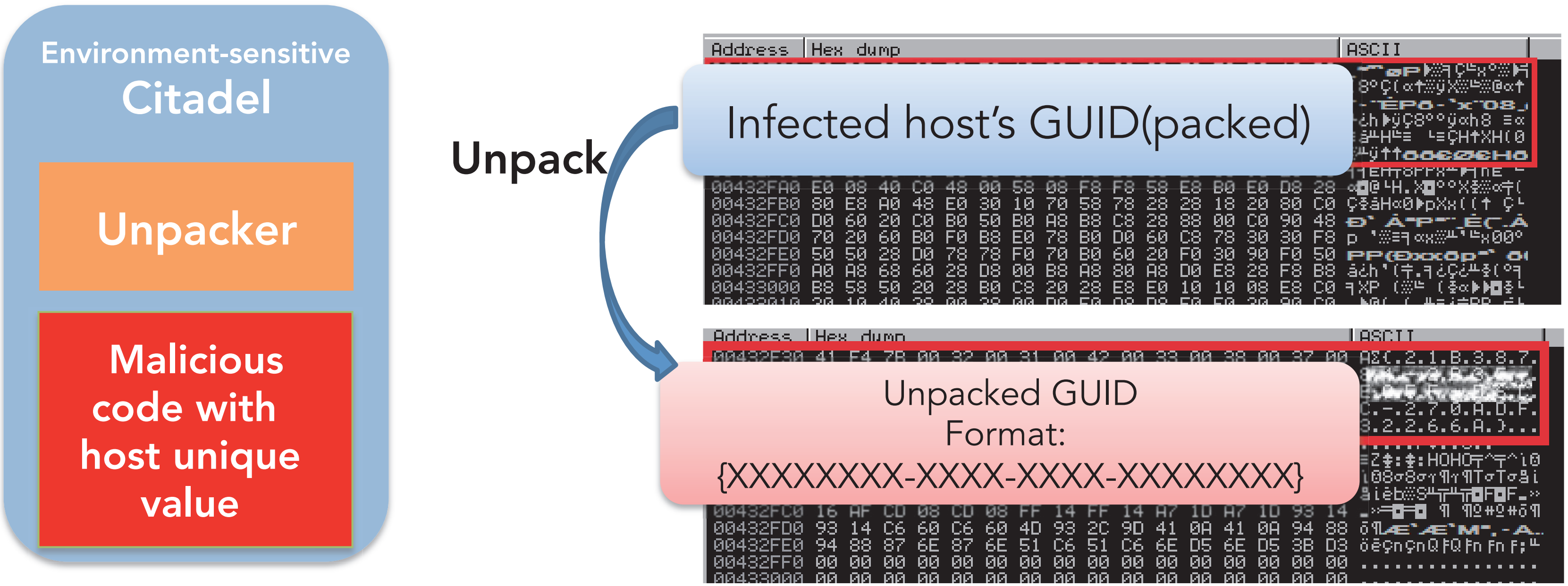
Host-fingerprinting

- Embedding infected host's unique value into execution binary



Host-fingerprinting(cont'd)

- Getting GUID on system drive using the GetVolumeNameForVolumeMountPoint()
- Comparing running host's GUID value and embedded infected host's value
- Process executes malicious code if GUID values are



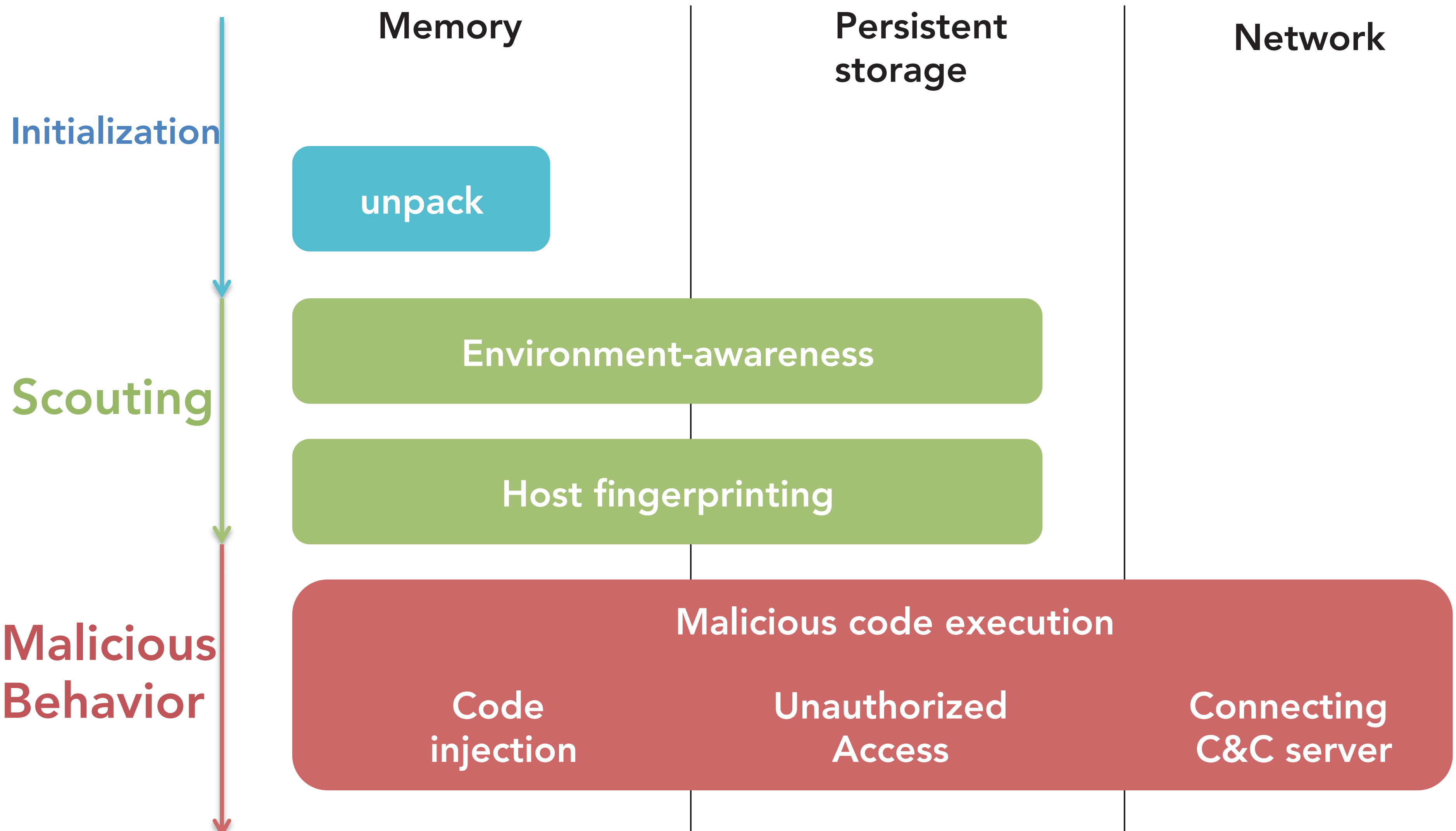
VM/Sandbox detection

- Checking process's product name
  - like "*vmware*", "*virtualbox"
- Scanning specific files and devices
  - C:¥popupkiller.exe
  - C:¥stimulator.exe
  - C:¥TOOLS¥execute.exe
  - ¥¥.¥NPF_NdisWanIp
  - ¥¥.¥HGFS
  - ¥¥.¥vmci
  - ¥¥.¥VBoxGuest

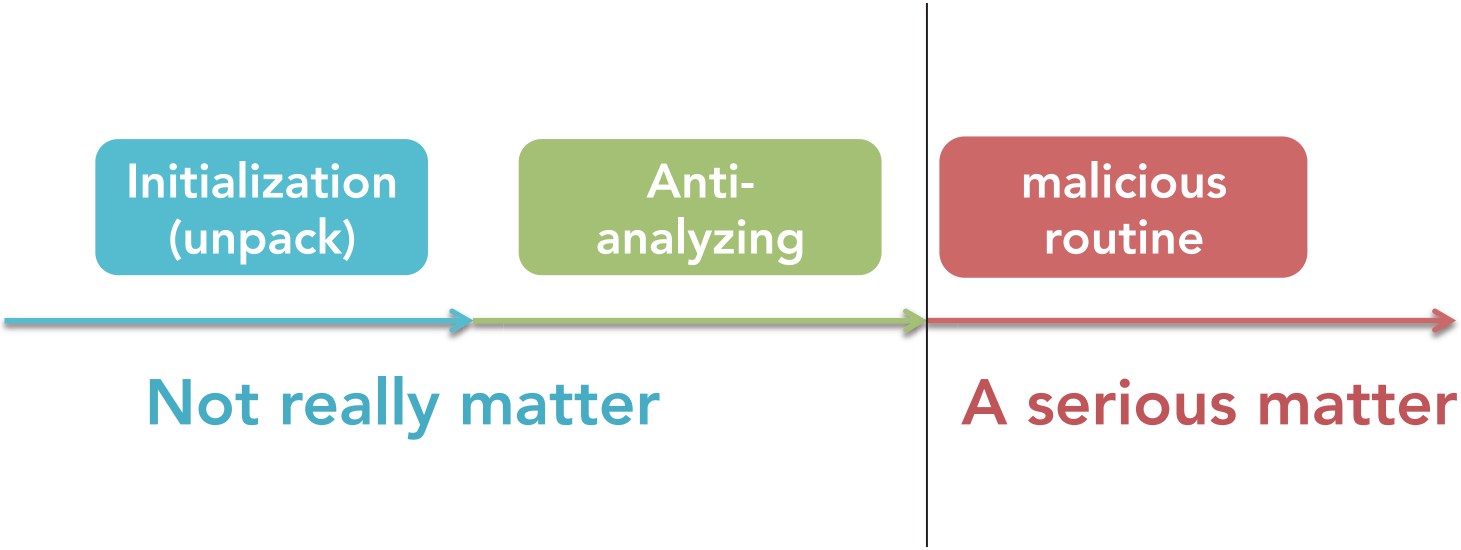
Citadel behavior of host/environment inconsistency

- For example:
  - Process termination
  - Running fake(or harmless) code

Citadel runtime activities

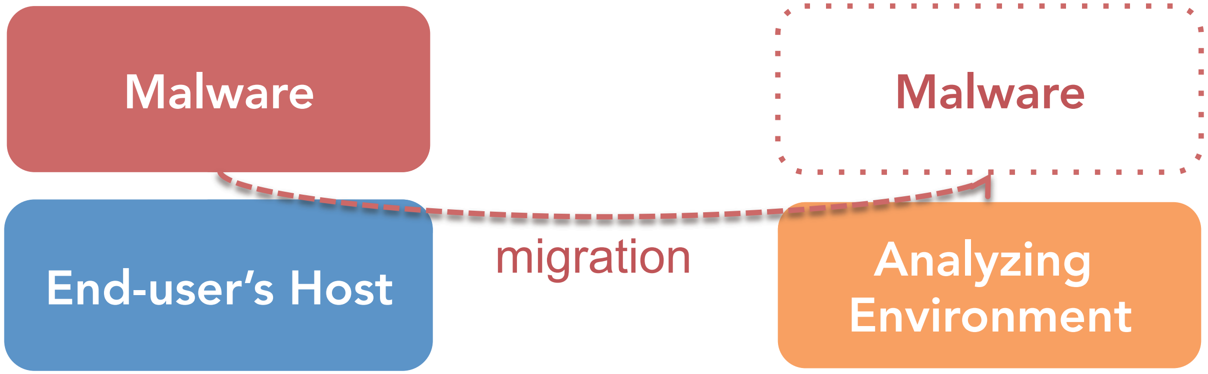


- I assume that scouting code carry out before main malicious routine



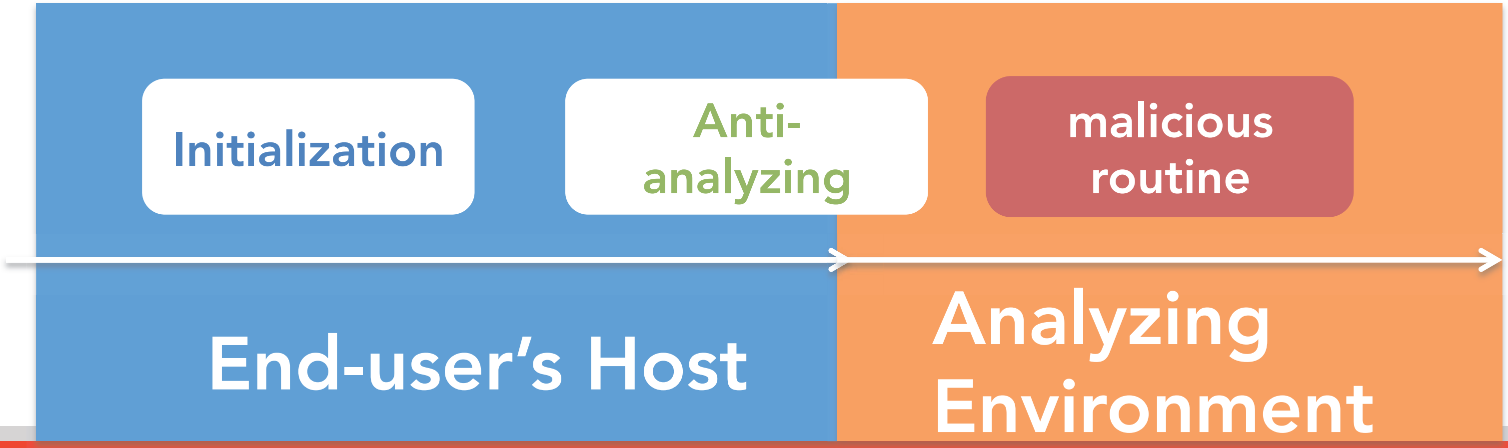
Idea

- Security analyst or incident handler concentrate malicious activity observation if he migrate malware process from infected host to analyzing environment(or honeypot) when anti-analyzing behavior



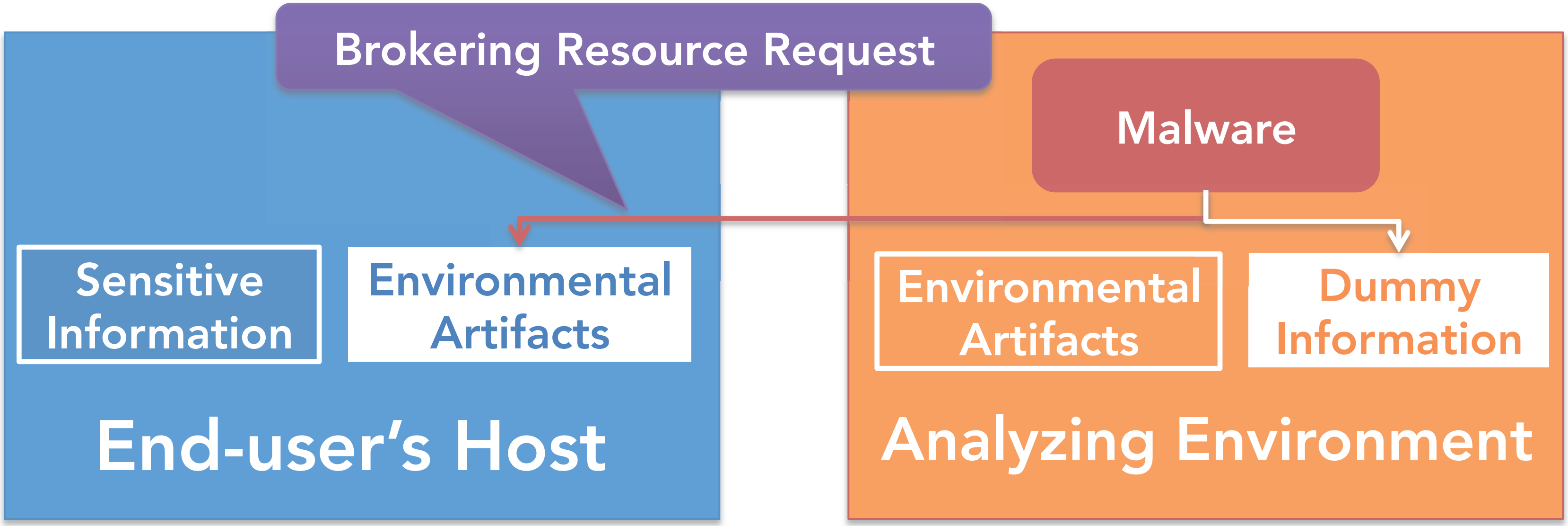
Use Case I: Malware live capturing

- End-users execute suspicious executable files anyway
- Capturing system will suspend program if to detect anti-analyzing behavior
- Malware analysts may observe to concentrate malicious activities



Use Case II: Honeypot

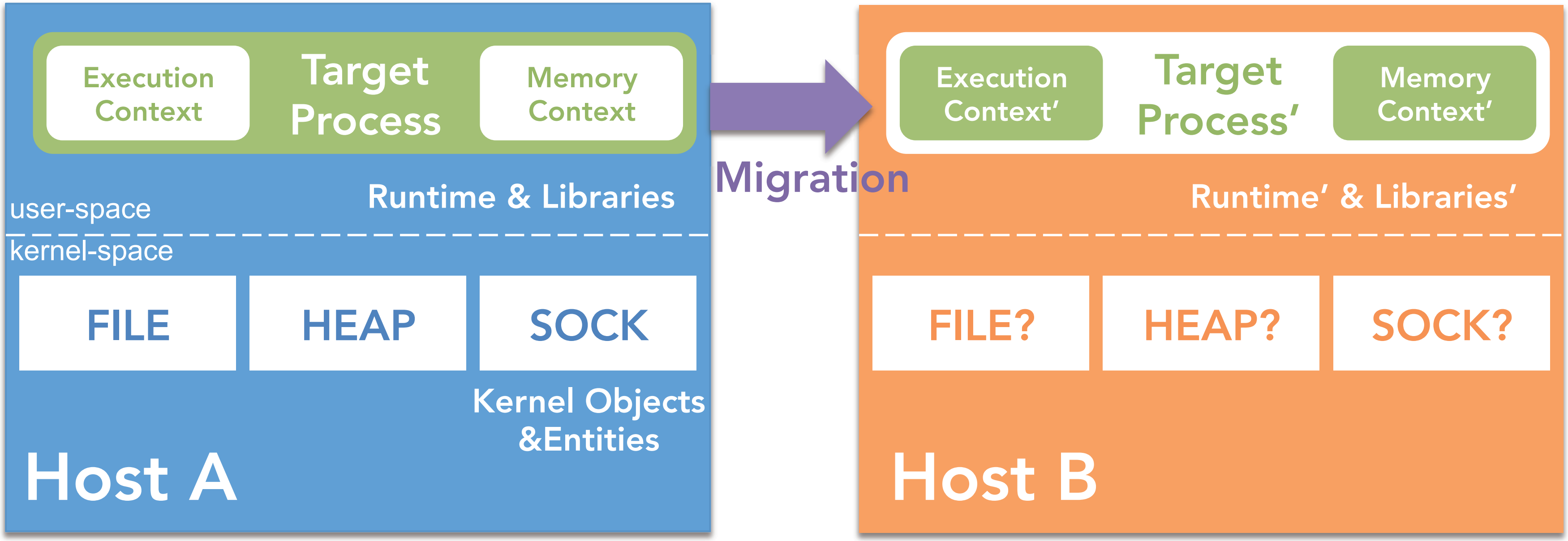
- Faking an artifact of the target host
  - To deceive cyber espionage malware



Challenges Ⅰ
PROCESS MIGRATION

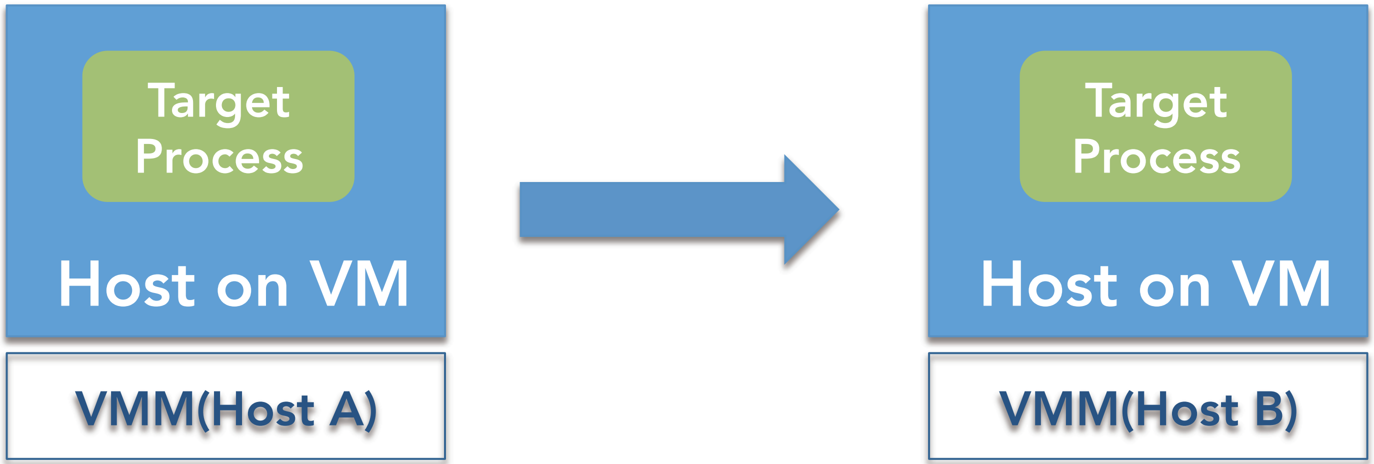
Challenges

1. Process migration is very difficult (well-known)
   - Needs to migrate execution contexts, memory contexts, persistent contexts and related kernel objects
   - Environment sensitivity



Off-topic: Virtual Machine migration

- VM migration is a practical way of process migration between hosts



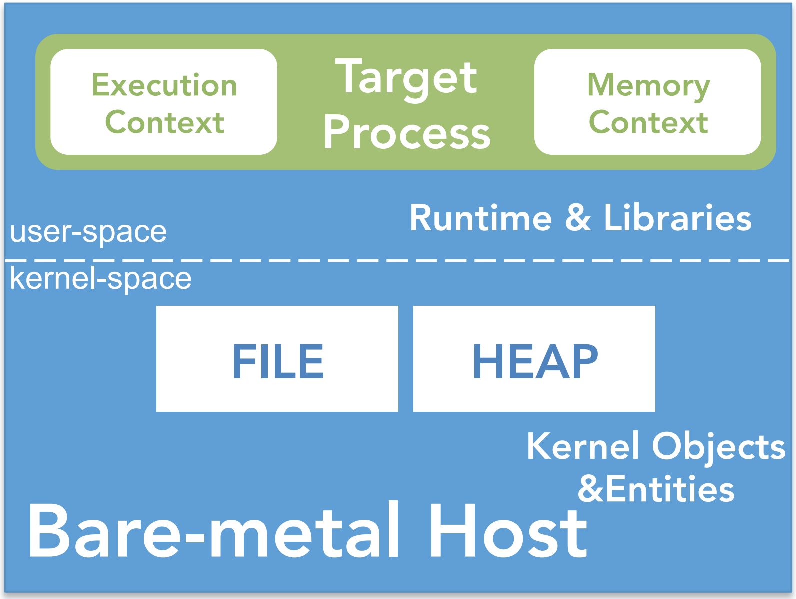
VM migration is too much larger

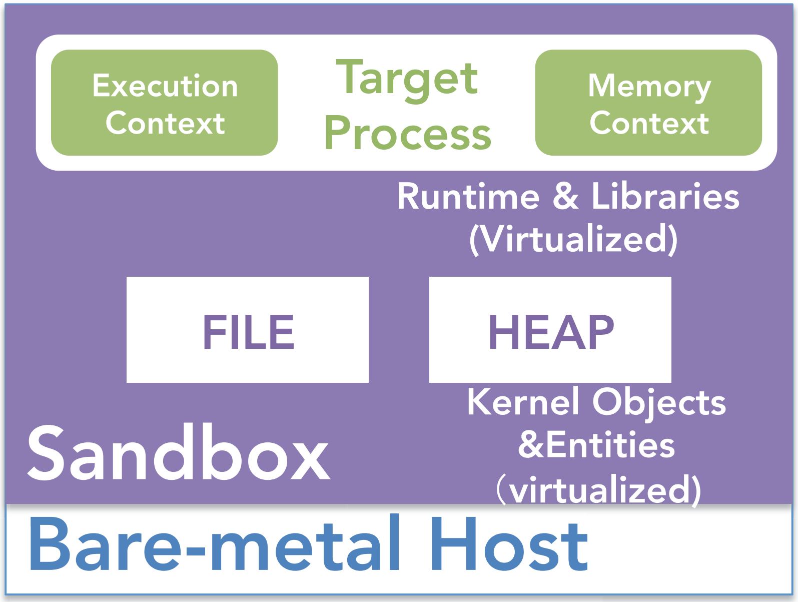
- Too many resources are migrated for malware analyzing
- VM solution forces additional system to end-users and employer
  - Increasing complexity, Maintainability and cost

Our solution: Using process-level sandbox

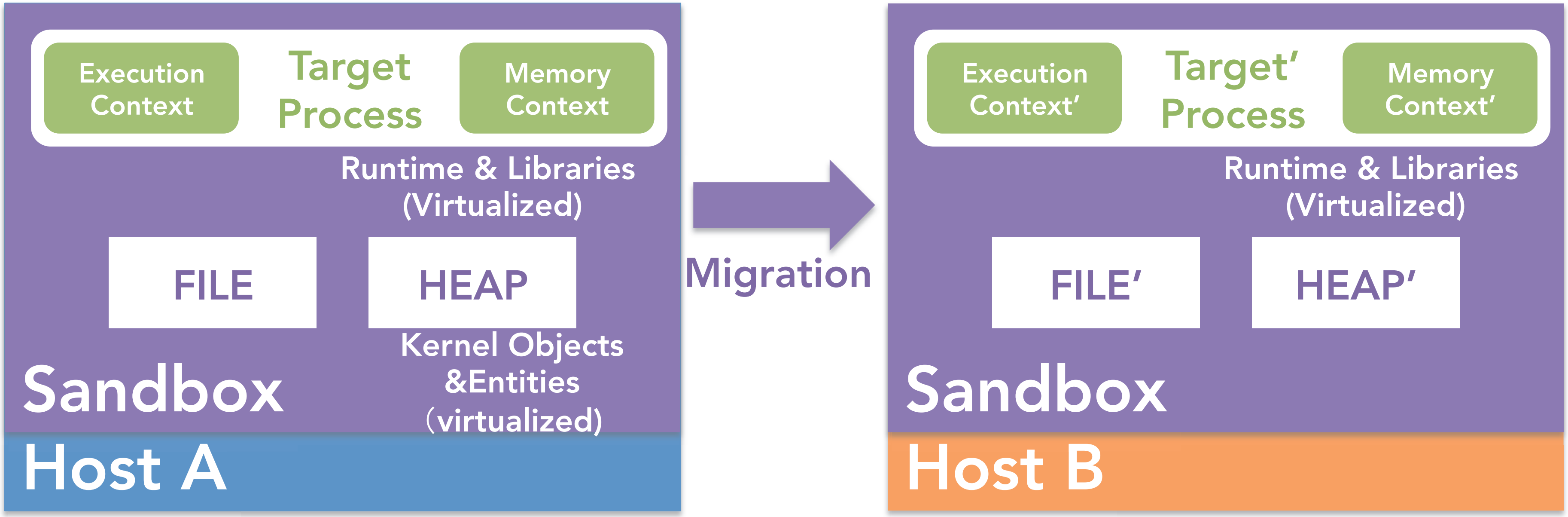
- CPU emulator-based sandbox is convenient for process migration
  - Grubbed all contexts
  - User-mode emulator virtualize process related kernel objects

CPU emulator-based sandbox



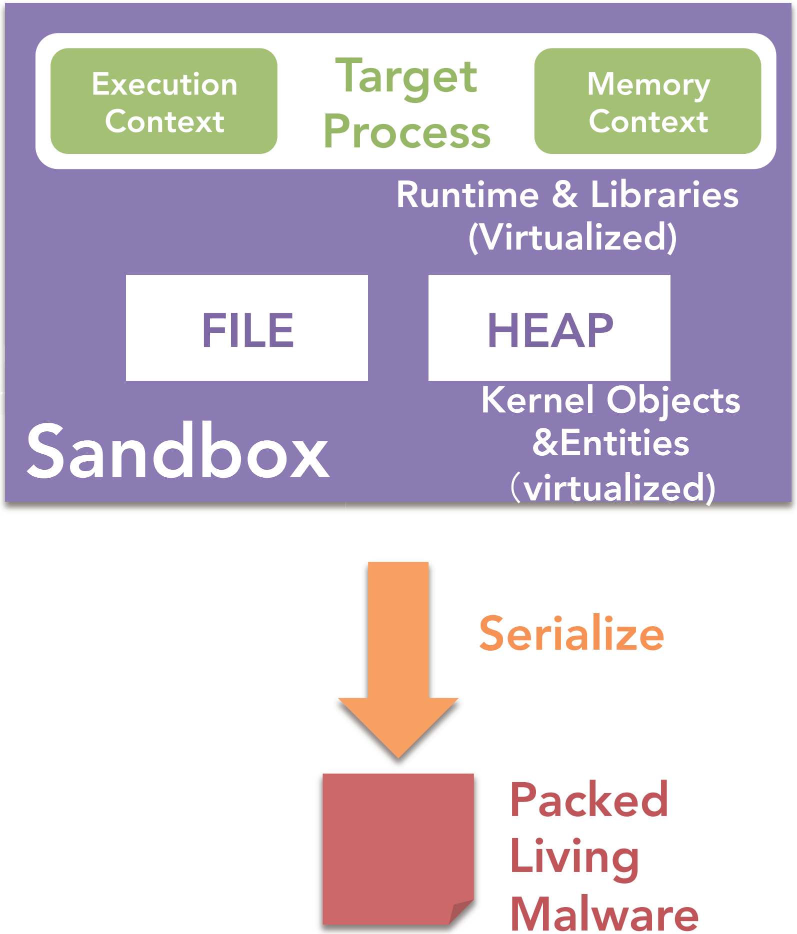


Process migration using CPU emulator-based sandbox



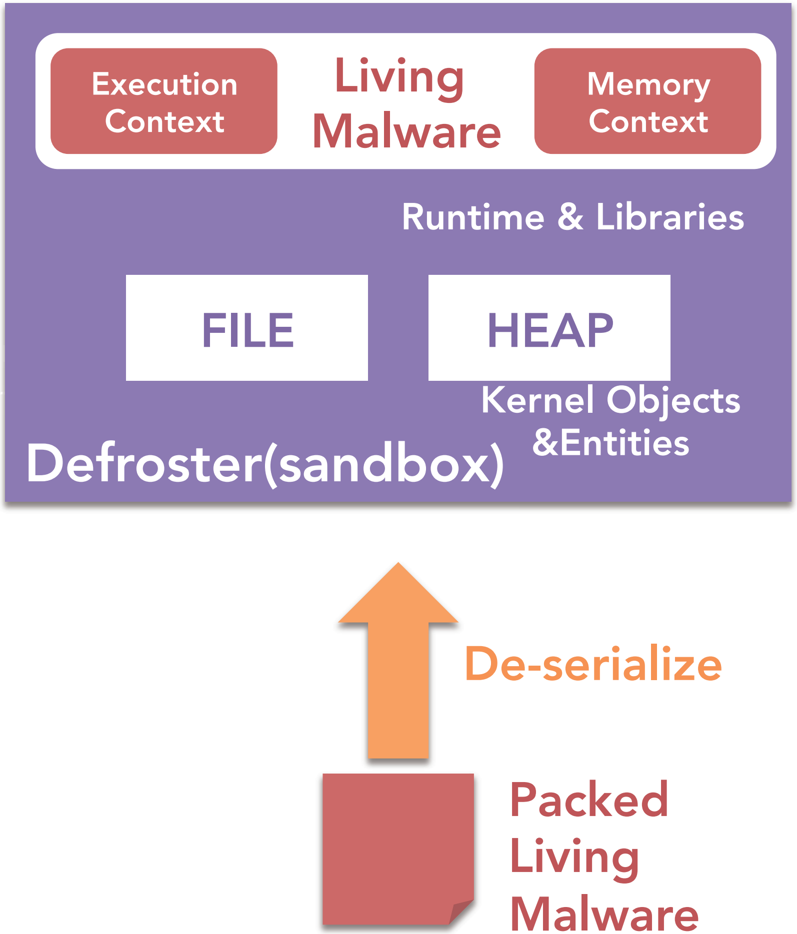
Malware freeze-drying

- Sandbox suspends target program when a trigger event occurred
- A suspended trigger is anti-analyzing behavior[2]



Live malware defrosting

- Sandbox resumed packed living malware
- Reconstructing address gaps



But…

- Migrated malware will probably executes anti-analyzing(anti-sandbox) continuously

- The system needs anti-anti-sandbox arming

Challenges(updated)

1. Process migration is very difficult
   →Using CPU emulator-based sandbox

2. Arming against anti-sandbox

Challenges Ⅱ

ANTI-ANTI-SANDBOX ARMING

Taxonomy of anti-sandbox techniques

- Anti-sandbox maneuver
  - Stalling code [3]
  - Environment awareness [4][5]
    - Using result of sandbox detection
  - (User interaction checks)

- Sandbox (debug/sandbox/vm) detection
  - Artifact fingerprinting[5][6]
  - Execution incongruousness[7][8]
  - Platform stimulation[9][10]

Stalling code

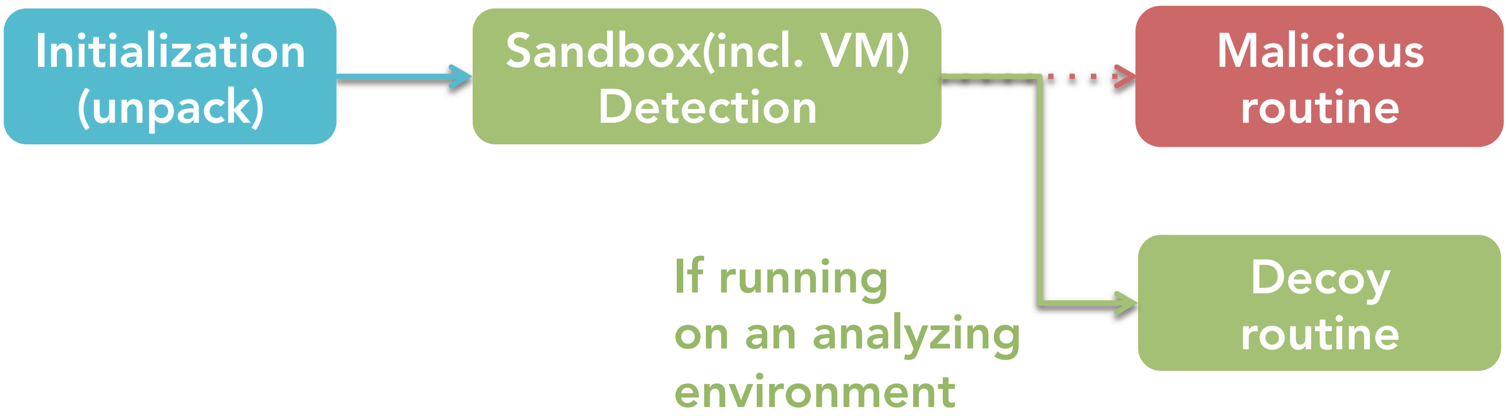
- Evasive malware[2] often uses
  - A sandbox limits malware execution time
- Stalling code detection and avoiding techniques already proposed[3]

```
unsigned count, t;
void helper() {
  t = GetTickCount();
  t++;
  t++;
  t = GetTickCount();
}
void delay() {
  count=0x1;
  do {
    helper(); // equal nop
    count++;
  } while
  (count!=0xe4e1c1);
}
```

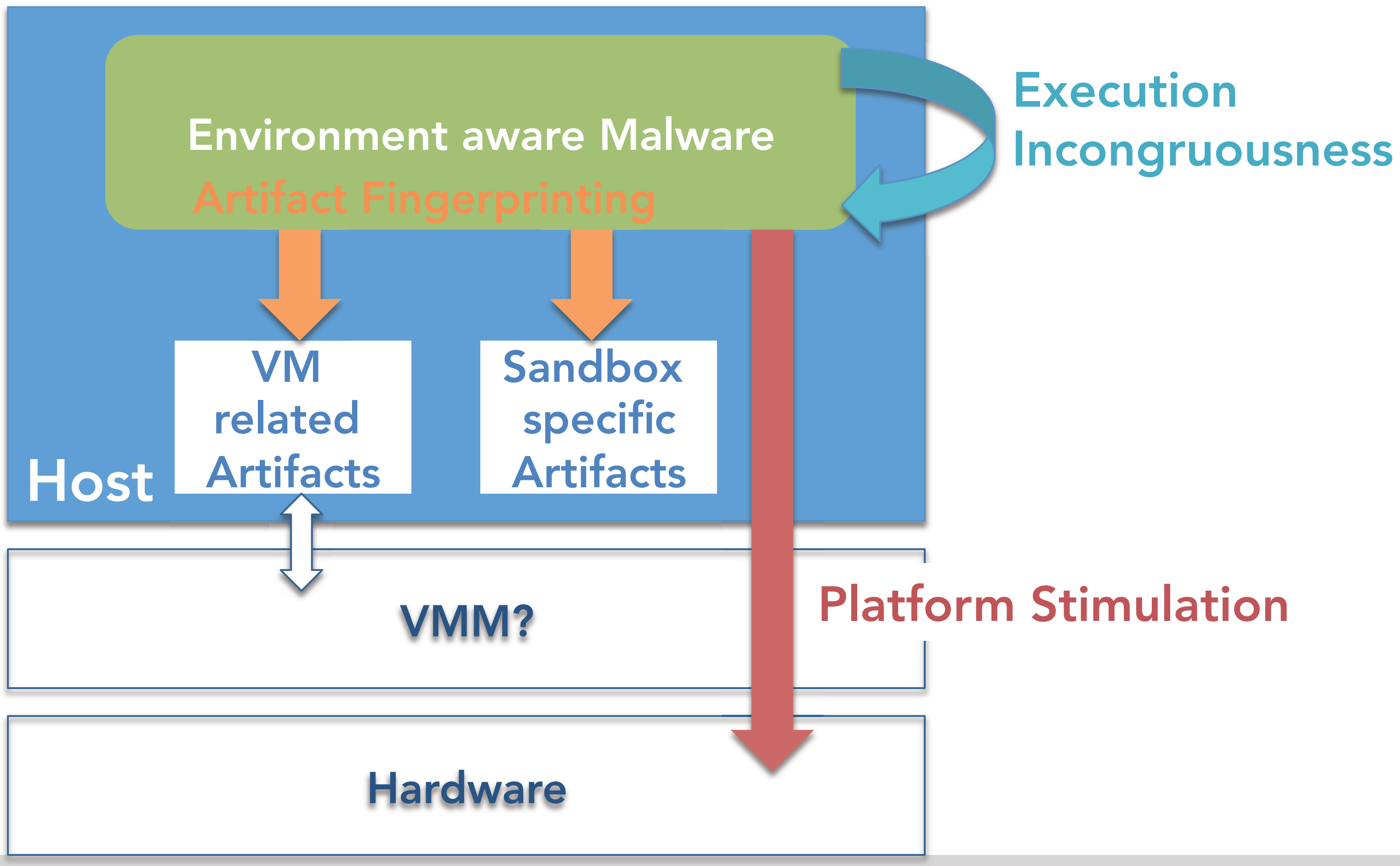
Stalling code in W32.DelfInj [3]

Environment awareness

- Checking host environments
- If malware runs decoy routine then it detects analyzer's sign
  - Malicious behavior never executed



Sandbox (debug/sandbox/vm) detection



Artifact Fingerprinting

- Sandbox/VM related processes
  - Like vmware, virtualbox etc.
- Sandbox/VM environment specific files
- Sandbox/VM environment specific registry keys
- Sandbox/VM environment specific devices and its attributes
  - ex). QEMU HDD vendor name
- Sandbox/VM Specific I/O port
  - VMWare backdoor port is most famous artifact in malware

Execution Incongruousness

- Using clock count differential
  - Traditional anti-debug technique
- Redpill[8]
  - Using LDT/GDT and IDT incongruousness

400022A2	60	PUSHAD
400022A3	0F31	RDTSC
400022A5	31C9	XOR ECX,ECX
400022A7	01C1	ADD ECX,EAX
400022A9	0F31	RDTSC
400022AB	29C8	SUB EAX,ECX
400022AD	3D FF0F0000	CMP EAX,0FFF
400022B2	61	POPAD
400022B3	0F83 11010000	JNB 400023CA

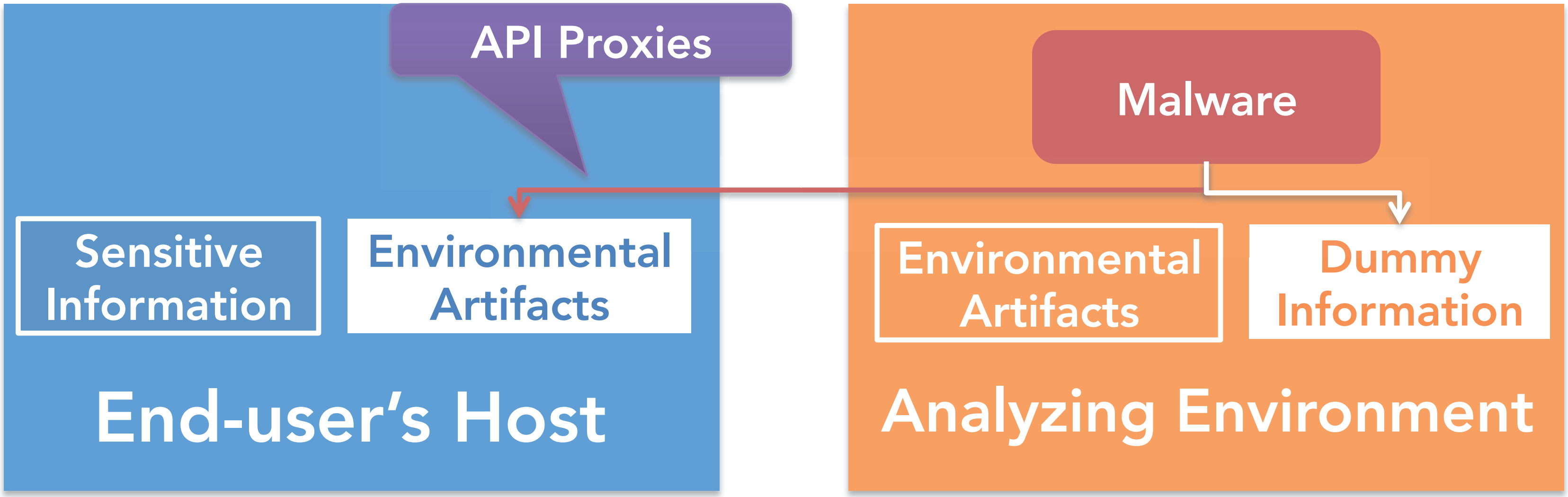
Comparing two TSC differentials

Platform stimulation

- Using virtual machine implementation differentials
  - Like CPUID instruction result
  - Interesting research here: Cardinal Pill Testing[9]

Our solution: Anti-anti-sandbox arming

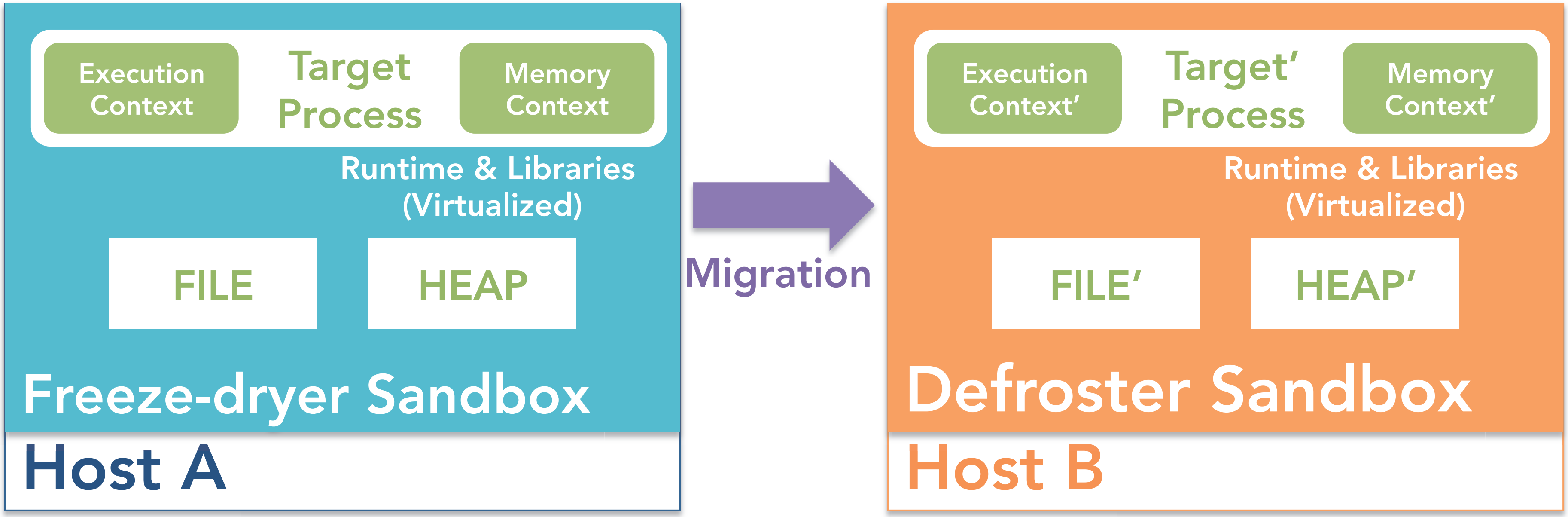
- <u>Hiding an artifact using API proxies</u>
- Stalling code detection and evasion(future work)
  - Following prior works
- Faithful CPU emulation(future work)
  - Following prior works and showing GUTS



IMPLEMENTATION

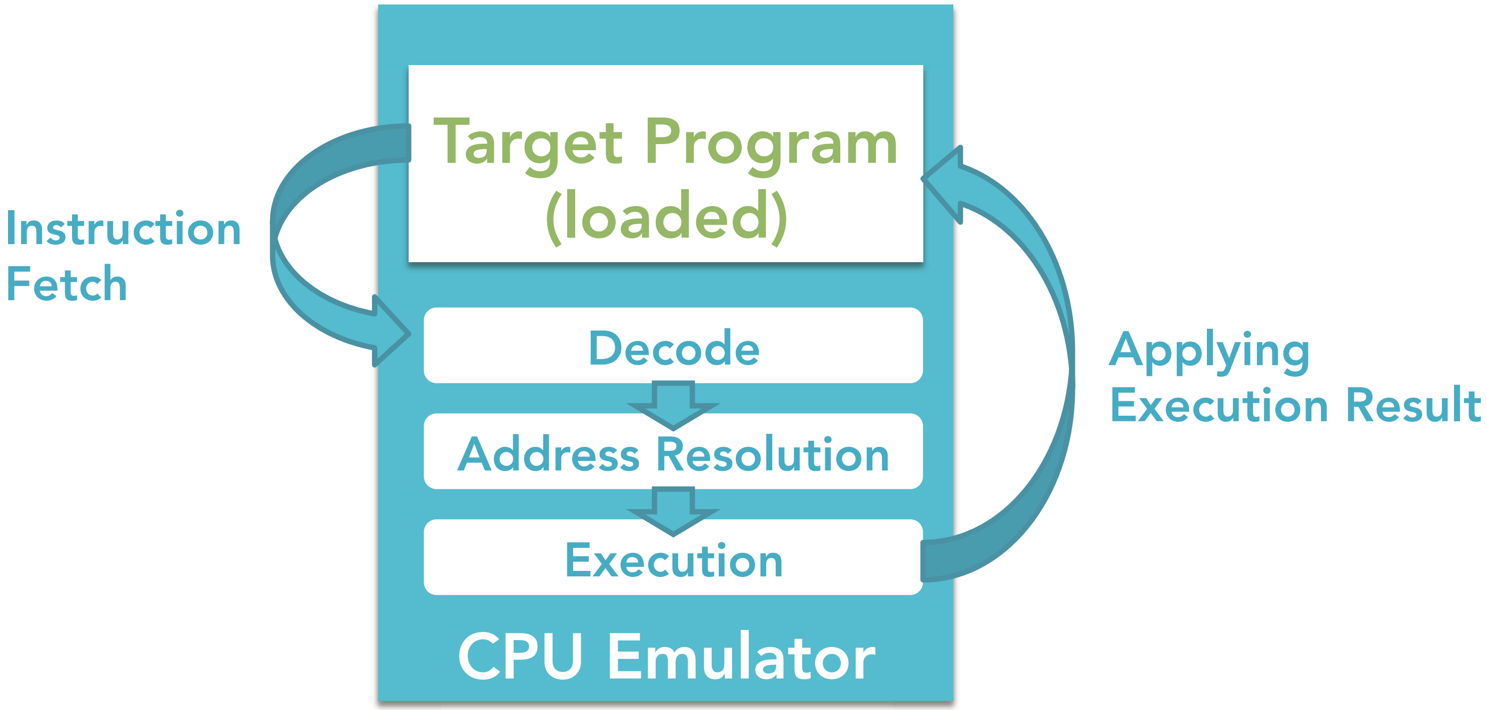
Sweetspot Overview

- Sweetspot have two sandbox, **Freeze-drying Sandbox** and **Defroster Sandbox**
- Sandboxes are based on IA-32 CPU emulator

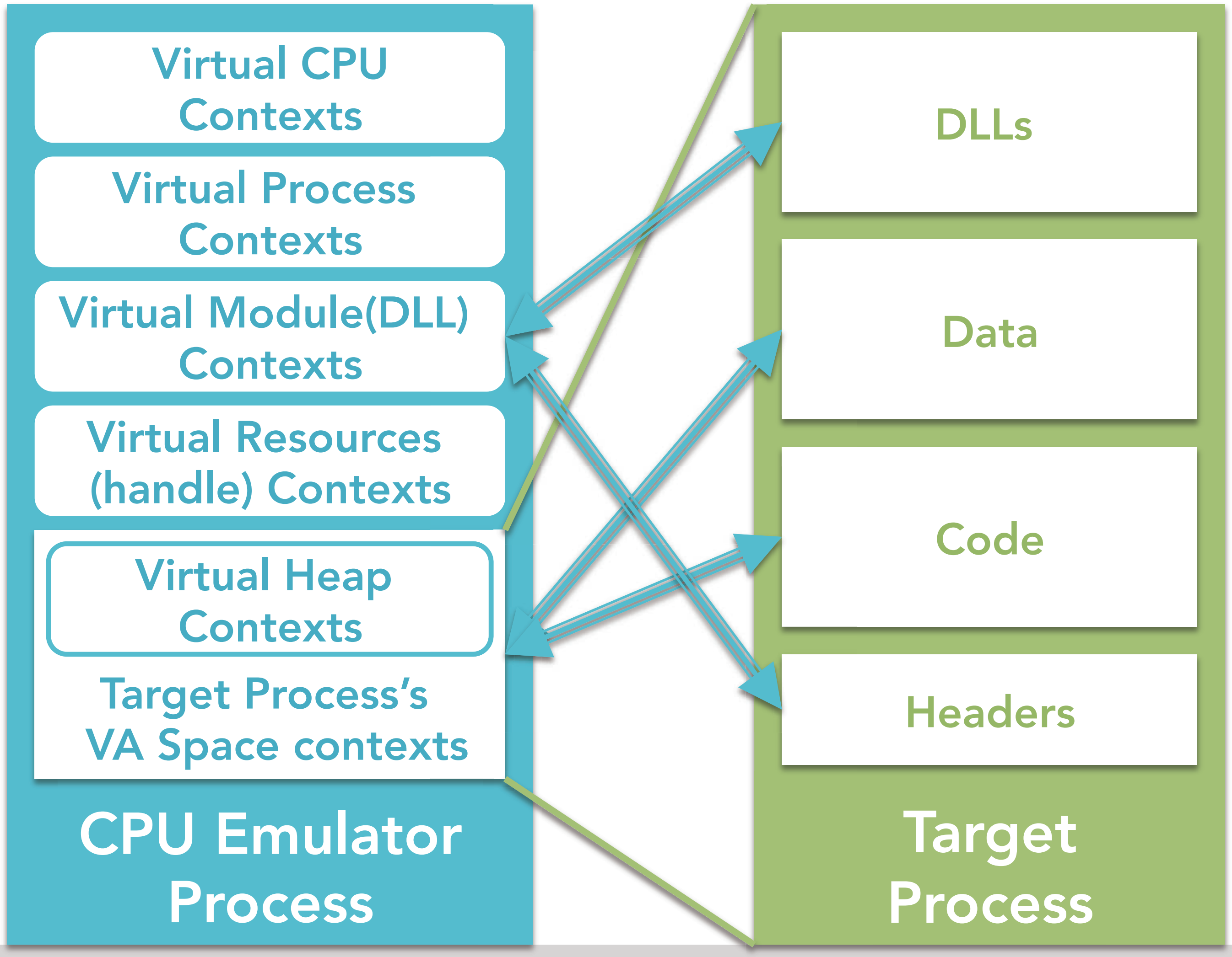


IA-32 CPU emulator-based sandbox

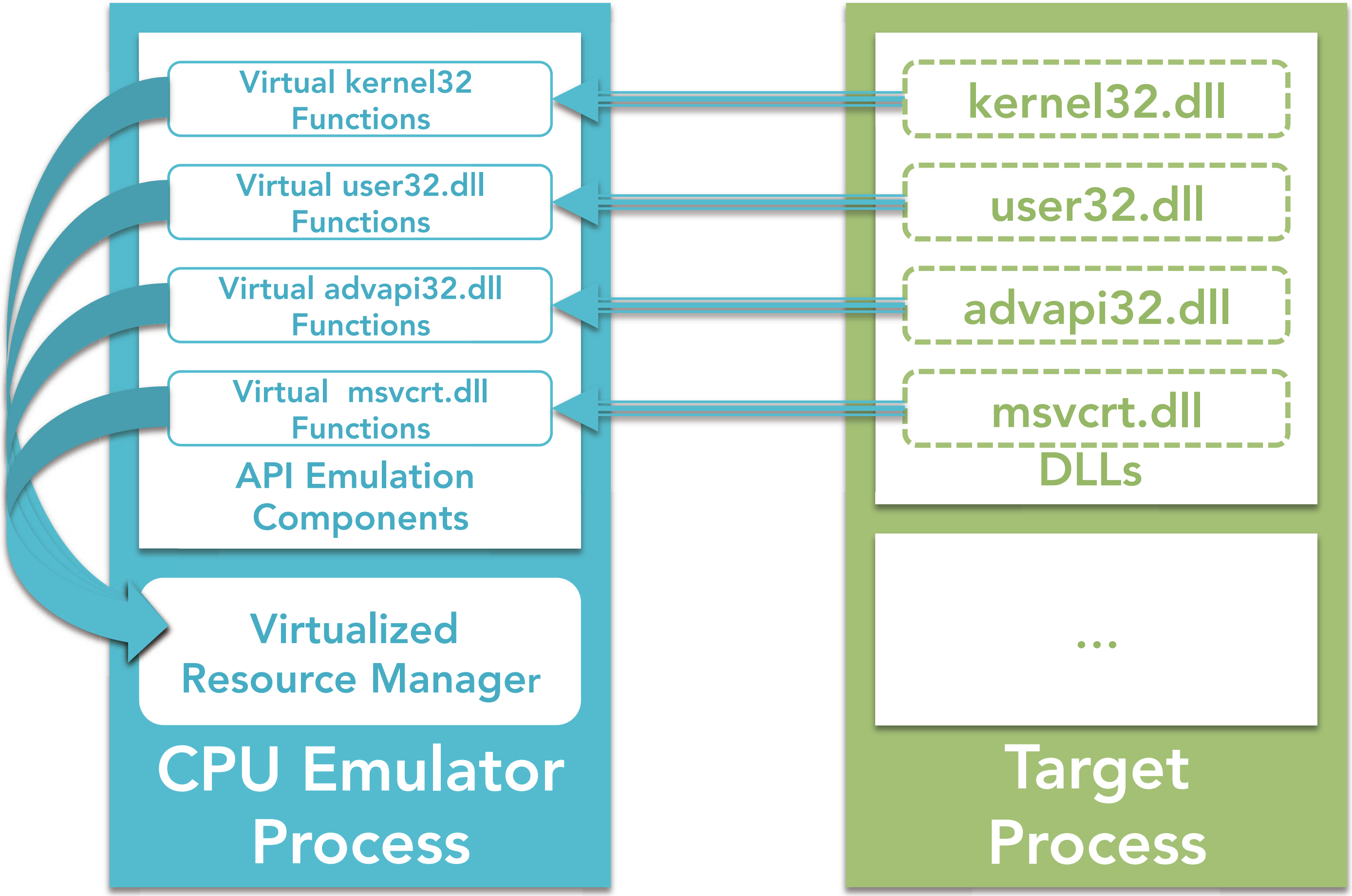
- We have already CPU Emulator-based sandbox for win32 execution (in-house use)
  - Like IDA Bochs PE operation mode[11]



IA-32 CPU Emulator: Virtual contexts



IA-32 CPU Emulator: API emulation

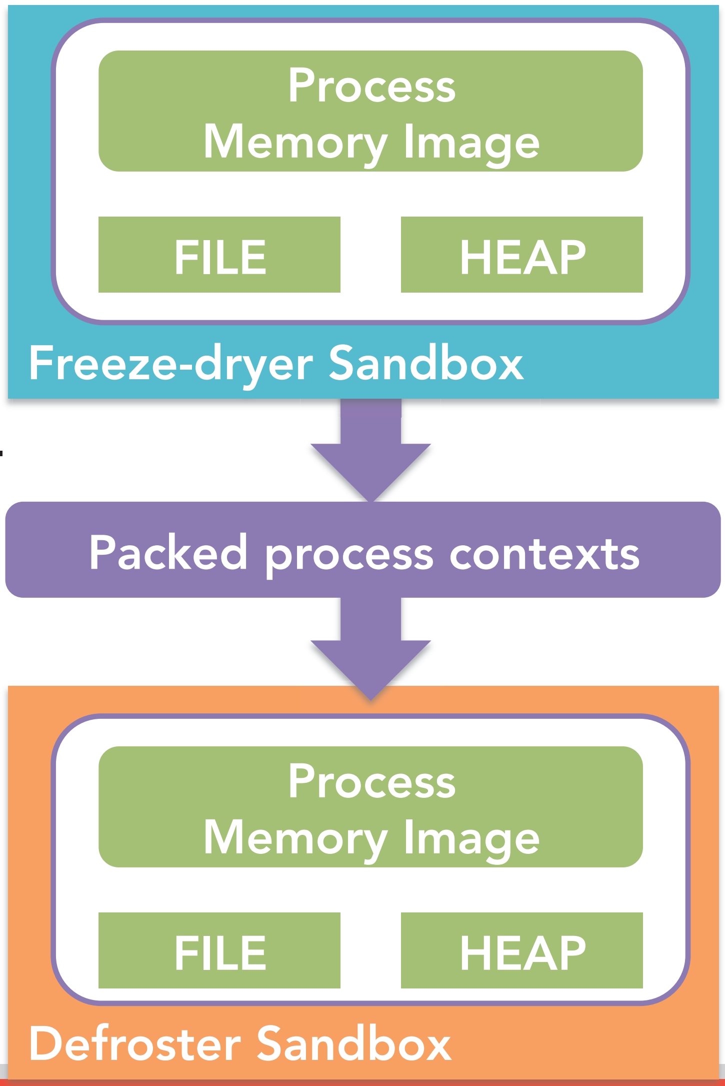


IA-32 CPU Emulator: Virtual resource handling

- File system is almost virtualized
- Registry hive is almost virtualized
- GUI components and user interaction function is virtualized partially
- media components is not virtualized(squashing request)

Sweetspot: Malware Live Capturing System

- Freeze-dryer
  - Serializing process contexts and execution file if detected suspend trigger
  - All malware activity sealed in the sandbox
- Defroster
  - Restoring execution context
  - Address reconstruction
  - API-proxies for faking an artifacts



Freeze-dryer

- End-user can use like an anti-virus's file scanner
- Freeze dryer serialize process context if detects anti-sandbox behavior occurred
  - Dumping all VA space anyway
- Using msgpack[12] library for serialization

Defined suspend trigger (Work in progress)

- Specific API-call
  - GetVolumeNameForVolumeMountPoint()
  - GetVolumeInformation()
- Specific API-call and its arguments
  - Searching vm-related artifacts
    - Virtual file system and virtual registry hive except finding sandbox artifacts
- Detecting stalling code(WiP)

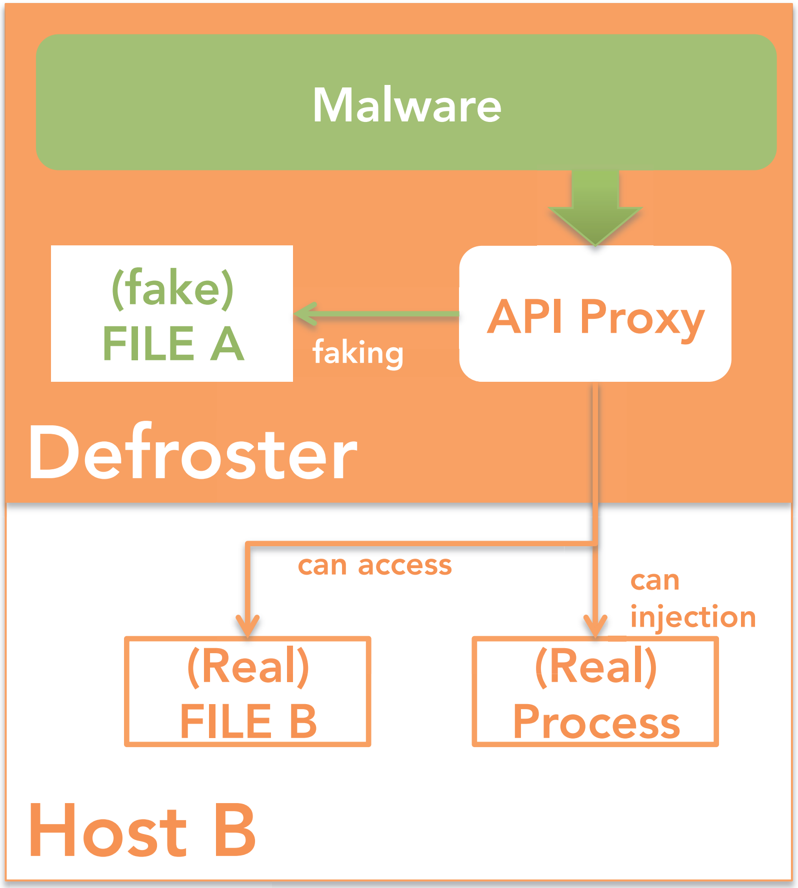
Defroster – Execution replaying

1. Unpacking process contexts(incl. execution file)
   - Allocating sandbox's heap
2. Loading execution file before entry point
3. Restoring current process context from unpacked contexts
   - Remapping address in unpacked process contexts
     - Covering all virtual address space

Demo: Process migrated!

API Proxies

- Malware can access specified directories on Defroster
  - Like %APPDATA%
- API Proxies enable to provide arbitrarily resources for malware



Anti anti sandbox arming using API Proxies

- Defroster performs play innocent with sandbox/vm related artifacts
  - No vm-related artifact exist in sandbox's virtual file system and virtual registry hive
- For faking an artifacts
  - Fake artifacts mounting virtual file system before malware resuming

Limitations

- The original CPU emulator supports a limited API
  - ex). Cannot CreateProcess and CreateThread
- The original CPU emulator supports a limited CPU instruction
  - ex). Cannot complete emulation with SSE instruction
- Anti-anti sandbox implementation is not enough
- API Proxies not supported Network API(winsock2) yet

Demonstrations

- Simple program (incl. heap and handle migration)
- Anti-anti-sandbox PoC

Future work

- Improving anti-sandbox detection and anti anti-sandbox
  - Stalling code detection and evasion
  - More faithful CPU/API emulation
- Improving API proxies utility
- Defroster-based stealth debugger

Conclusions

- This is proof of concept of live malware capturing using process migration with CPU emulator-based sandbox
- We introduced anti-sandbox taxonomy and proposed API-proxy based countering approach

References

- [1]: Analyzing Environment-Aware Malware, Lastline, 2014.05.25(viewed) http://labs.lastline.com/analyzing-environment-aware-malware-a-look-at-zeus-trojan-variant-called-citadel-evading-traditional-sandboxes
- [2]: Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. 2011. Detecting environment-sensitive malware. In *Proceedings of the 14th international conference on Recent Advances in Intrusion Detection* (RAID'11). Springer-Verlag, Berlin, Heidelberg, 338-357.
- [3]: lemens Kolbitsch, Engin Kirda, and Christopher Kruegel. 2011. The power of procrastination: detection and mitigation of execution-stalling malicious code. In Proceedings of the 18th ACM conference on Computer and communications security (CCS '11). ACM, New York, NY, USA, 285-296.
- [4]: Min Gyung Kang, Heng Yin, Steve Hanna, Stephen McCamant, and Dawn Song. 2009. Emulating emulation-resistant malware. In Proceedings of the 1st ACM workshop on Virtual machine security (VMSec '09). ACM, New York, NY, USA, 11-22.
- [5]: Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. 2014. Barecloud: bare-metal analysis-based evasive malware detection. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA, 287-301.
- [6]: Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. 2009. A view on current malware behaviors. In Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET'09). USENIX Association, Berkeley, CA, USA, 8-8.
- [7]: Aurélien Wailly. Malware vs Virtualization The endless cat and mouse play, 2014.05.25(viewed) http://aurelien.wail.ly/publications/hip-2013-slides.html
- [8]: Lorenzo Martignoni, Roberto Paleari, Giampaolo Fresi Roglia, and Danilo Bruschi. 2009. Testing CPU emulators. In Proceedings of the eighteenth international symposium on Software testing and analysis (ISSTA '09). ACM, New York, NY, USA, 261-272.
- [9]: Hao Shi, Abdulla Alwabel and Jelena Mirkovic. 2014. Cardinal Pill Testing of System Virtual Machines. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA,271-285.
- [10]: Lorenzo Martignoni, Roberto Paleari, Giampaolo Fresi Roglia, and Danilo Bruschi. 2010. Testing system virtual machines. In Proceedings of the 19th international symposium on Software testing and analysis (ISSTA '10). ACM, New York, NY, USA, 171-182.
- [11]: IDA Boch PE operation mode https://www.hex-rays.com/products/ida/support/idadoc/1332.shtml
- [12]: MessagePack, 2014/09/28(viewed) http://msgpack.org/

Thank you !

FFRI, Inc.
http://www.ffri.jp