ARTIFACT EVALUATED
usenix ASSOCIATION
PASSED

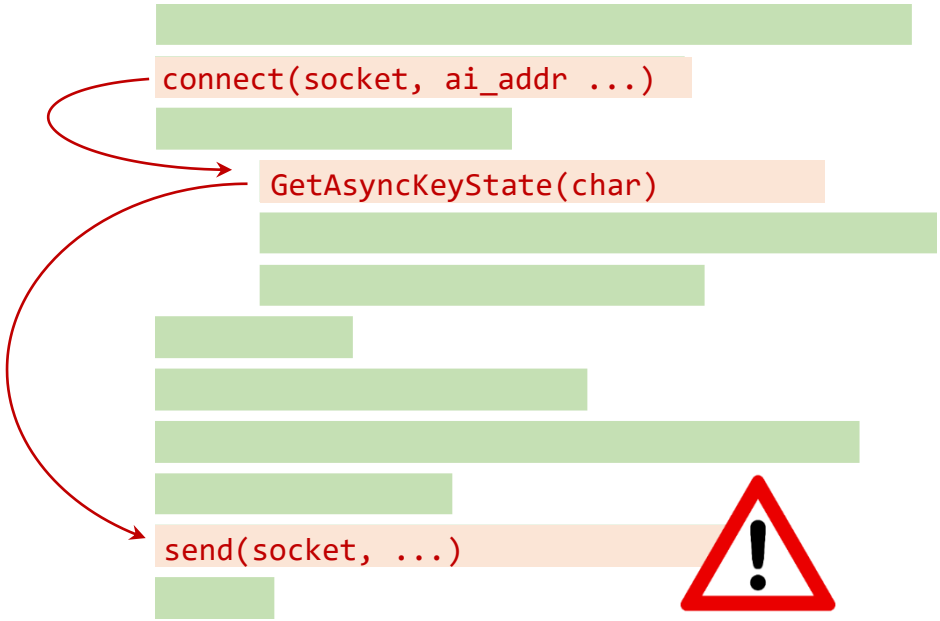
D-TIME
Distributed Threadless Independent Malware Execution for Runtime Obfuscation

13th USENIX Workshop on Offensive Technologies

Jithin Pavithran
Milan Patnaik
Chester Rebeiro

Advanced Malware Detection Techniques

Behavioral Detection → **Run time behavior**

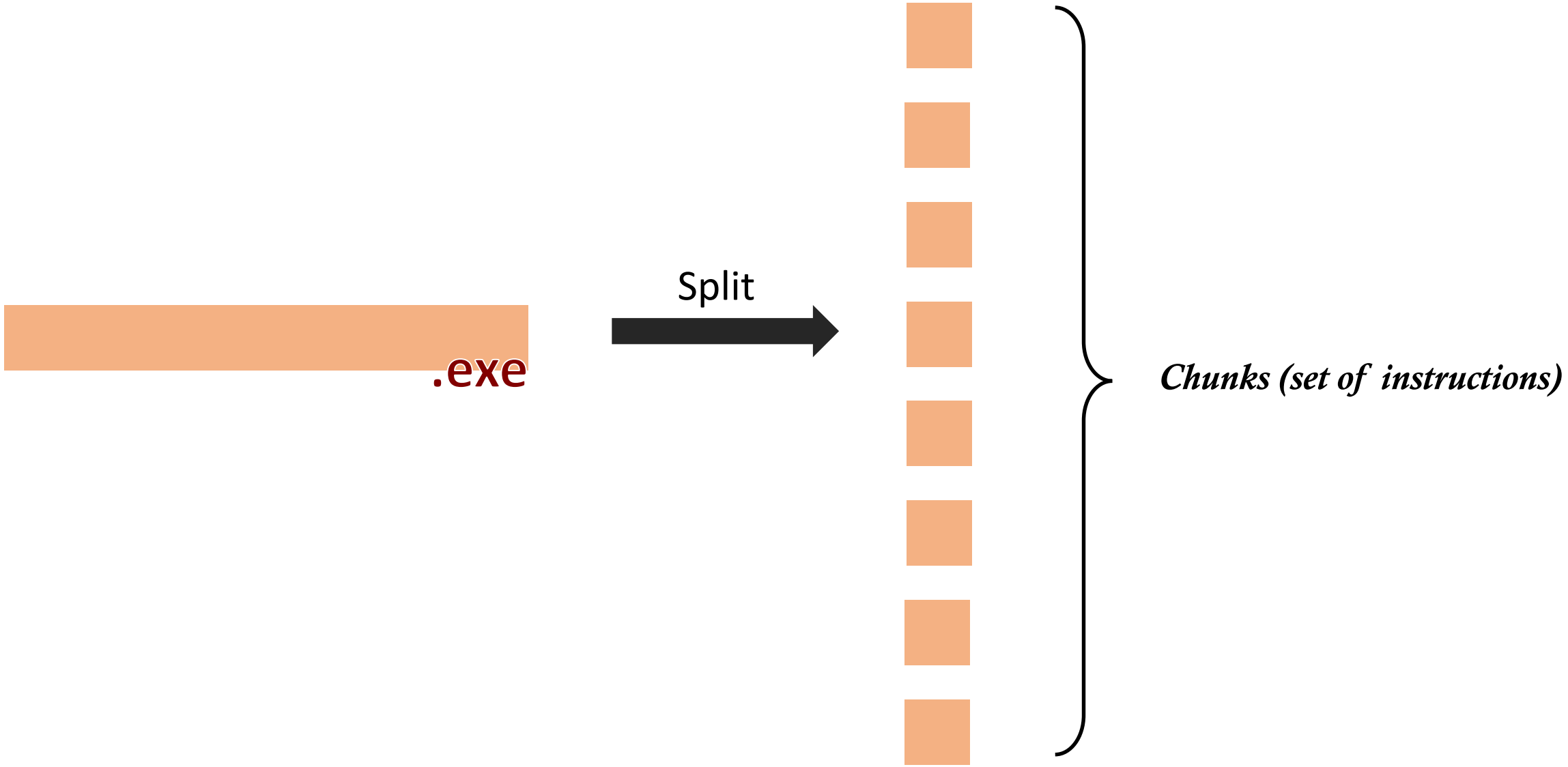


D-TIME
Distributed Threadless Independent Malware Execution

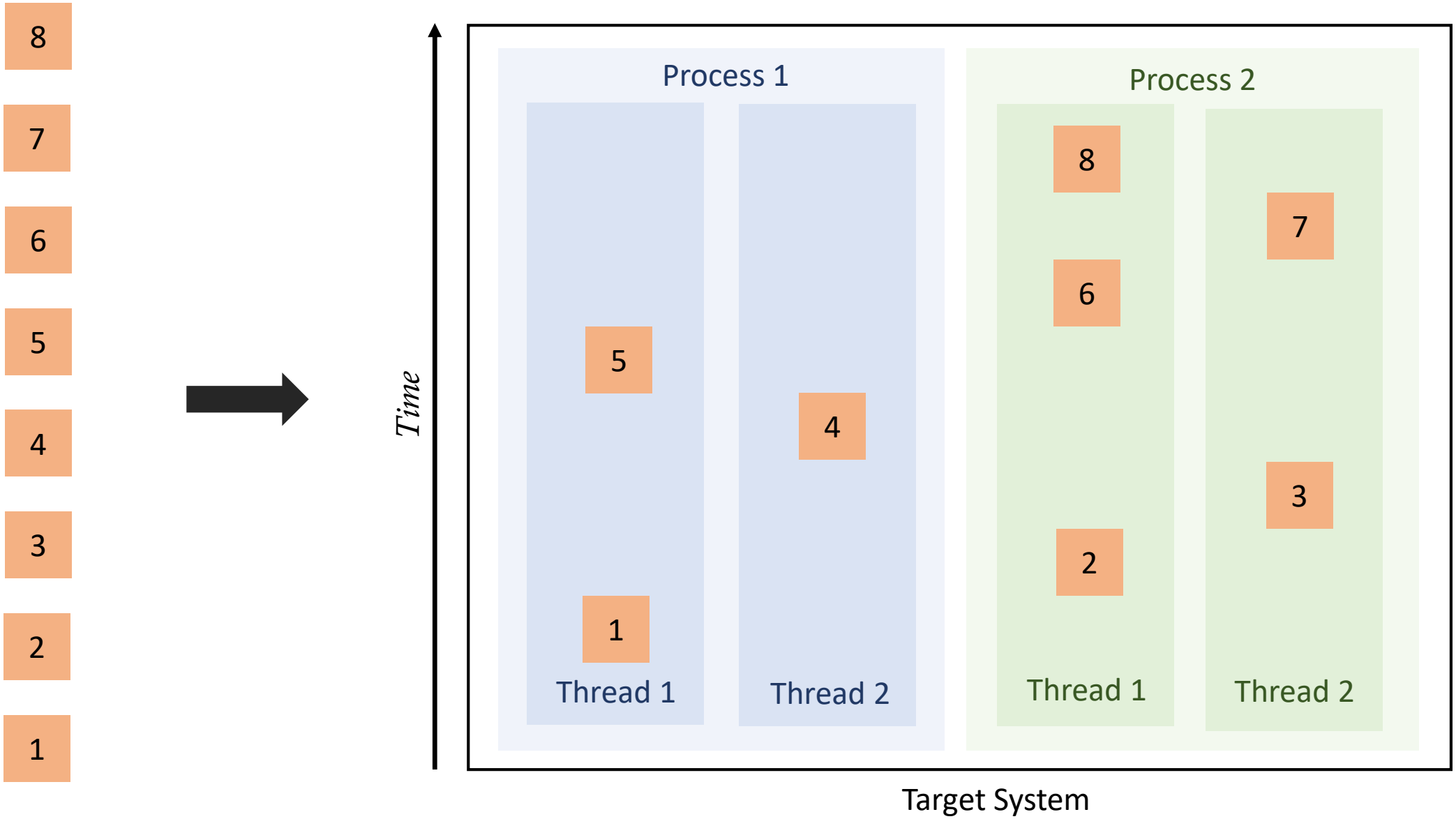
D-TIME

D-TIME

D-TIME



D-TIME

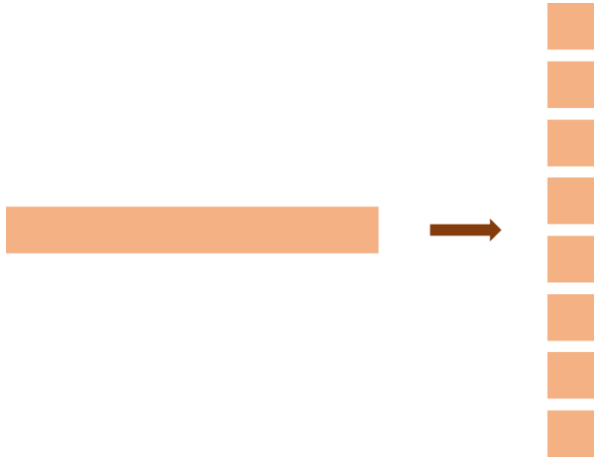

8
7
6
5
4
3
2
1
Time
Process 1
5
1
Thread 1
4
Thread 2
Process 2
8
6
2
Thread 1
7
3
Thread 2
Target System

Difficulties of Distribution

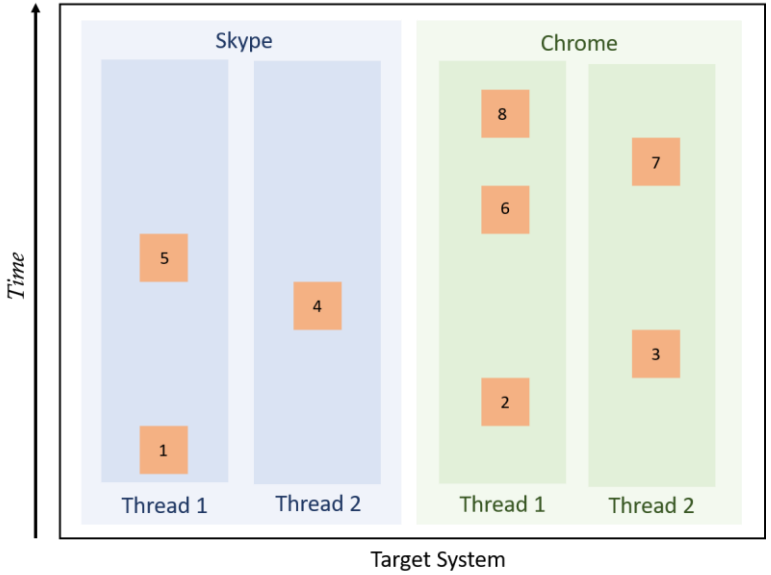
- Sequence of Chunks
- Context

D-TIME

Offline Phase



Online Phase



D-TIME – Offline Phase

```
      ...                  ;
    cmp [a], [b]           ; if(a==b){
    jne a_unequal_b        ;
    mov [a], 0             ;     a = 0
a_unequal_b:               ; }
    mov [c], [d]           ; c = d
      ...                  ;
```

Splitting technique adopted from MalWASH (WOOT'16)

D-TIME – Offline Phase

Chunk 1

```
  ...                    ;
cmp [a], [b]             ; if(a==b){
jne  a_unequal_b         ;
```

Chunk 2

```
mov [a], 0               ;     a = 0
```

Chunk 3

```
mov [c], [d]             ; c = d
  ...                    ;
```


Splitting technique adopted from MalWASH (WOOT'16)

D-TIME – Offline Phase

Chunk 1

```
...                     ;
cmp [a], [b]            ; if(a==b){
jne a_unequal_b         ;
mov ebx, 2
jmp END
jmp a_unequal_b:
mov ebx, 3
```

Chunk 2

```
mov [a], 0              ;     a = 0
mov ebx, 3
```

Chunk 3

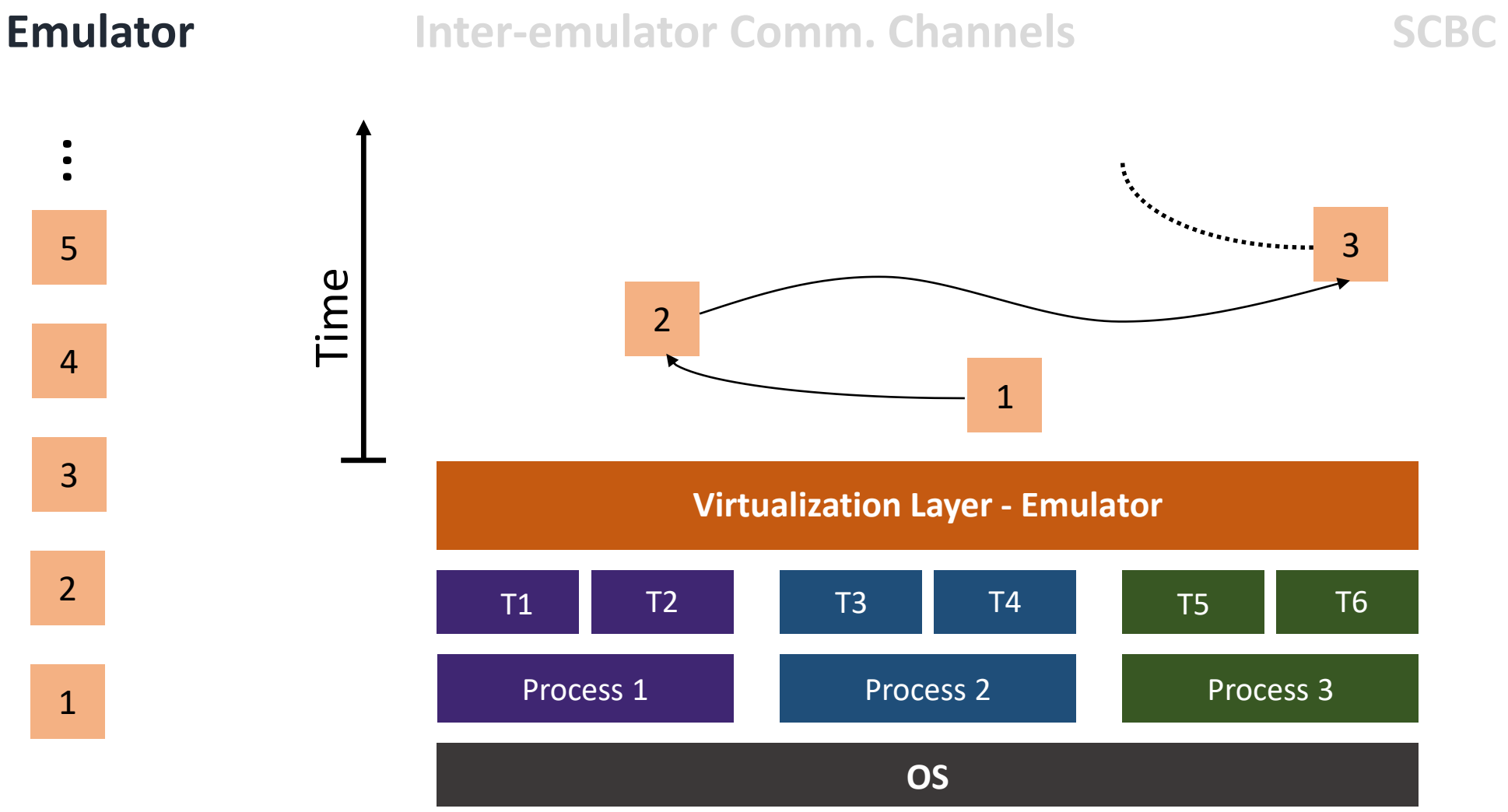
```
mov [c], [d]            ; c = d
...                     ;
```

Splitting technique adopted from MalWASH (WOOT'16)

D-TIME – Online Phase

Emulator **Inter-emulator Comm. Channels** **SCBC**

D-TIME – Online Phase

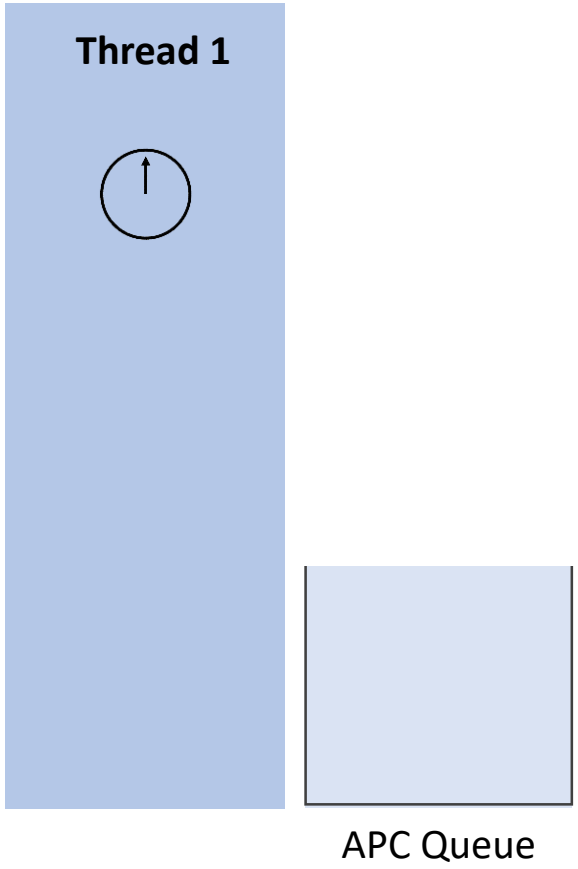


D-TIME – Online Phase

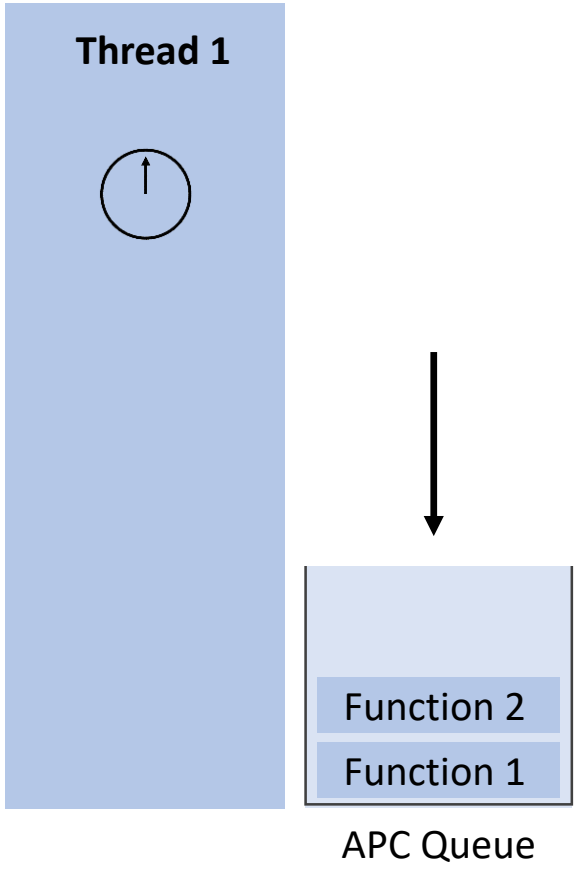
Emulator Inter-emulator Comm. Channels SCBC

- A Function
- Gets executed in benign thread's context
- Creates the virtualization layer
- Execute a chunk
- Maintaining
  - ✓ Sequence of chunks
  - ✓ Continuity of state (stack, registers, heap)
- Ensures completeness of malware

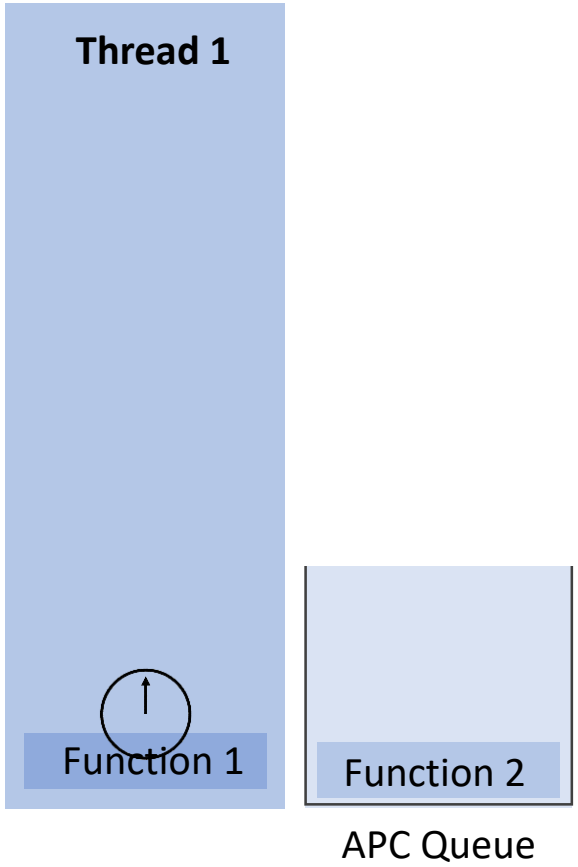
APC
(Asynchronous Procedural Call)



APC
(Asynchronous Procedural Call)



APC
(Asynchronous Procedural Call)



Alertable Wait State

```
SleepEx
SignalObjectAndWait
MsgWaitForMultipleObjectsEx
WaitForMultipleObjectsEx
WaitForSingleObjectEx
```

D-TIME

Emulator Inter-emulator Comm. Channels SCBC

Steps:

1. Connect to Comm. Channel
2. Retrieve
3. Statefing

Online Phase: A series of emulator executions,

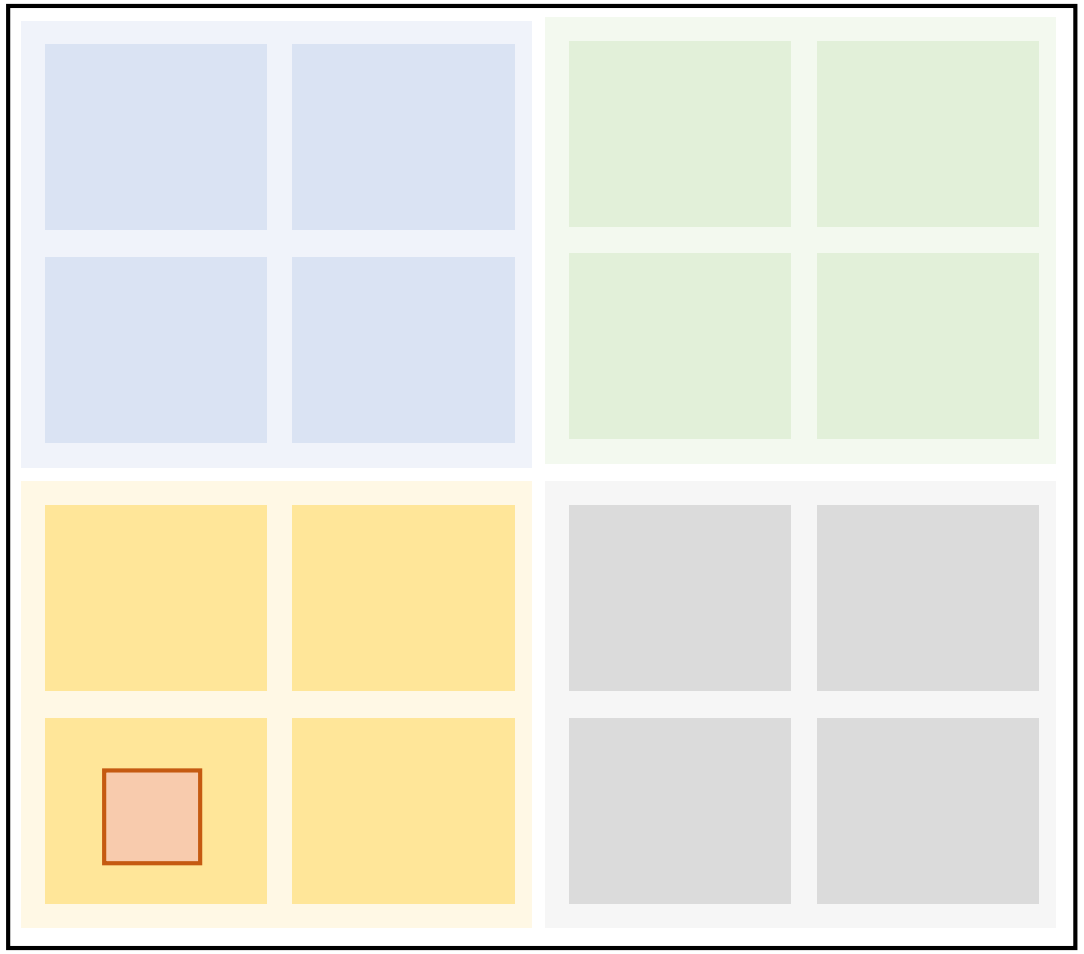
Emulators

- Spawn in different threads
- Executing one chunk each
- Collectively completing the sequence.

5. Regenerate itself

D-TIME

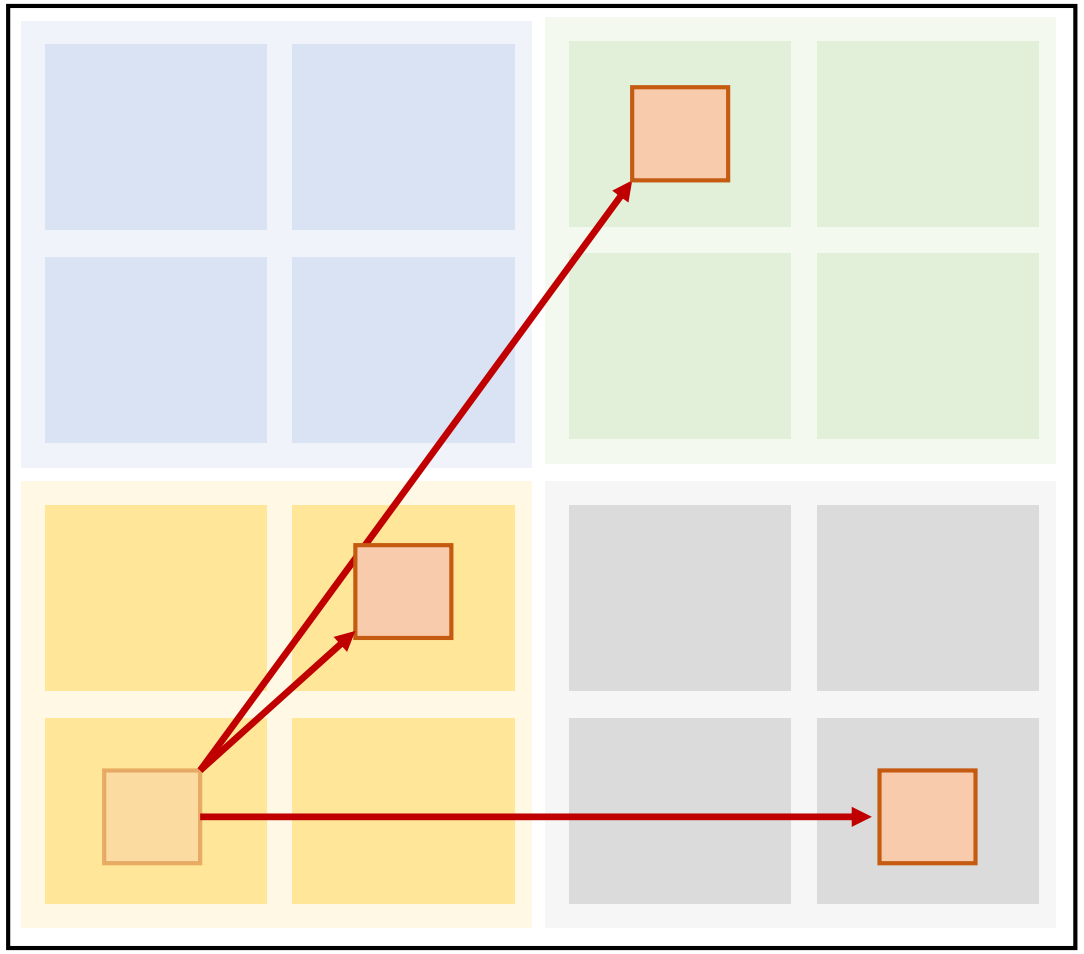
Emulator Regeneration



Target System

D-TIME

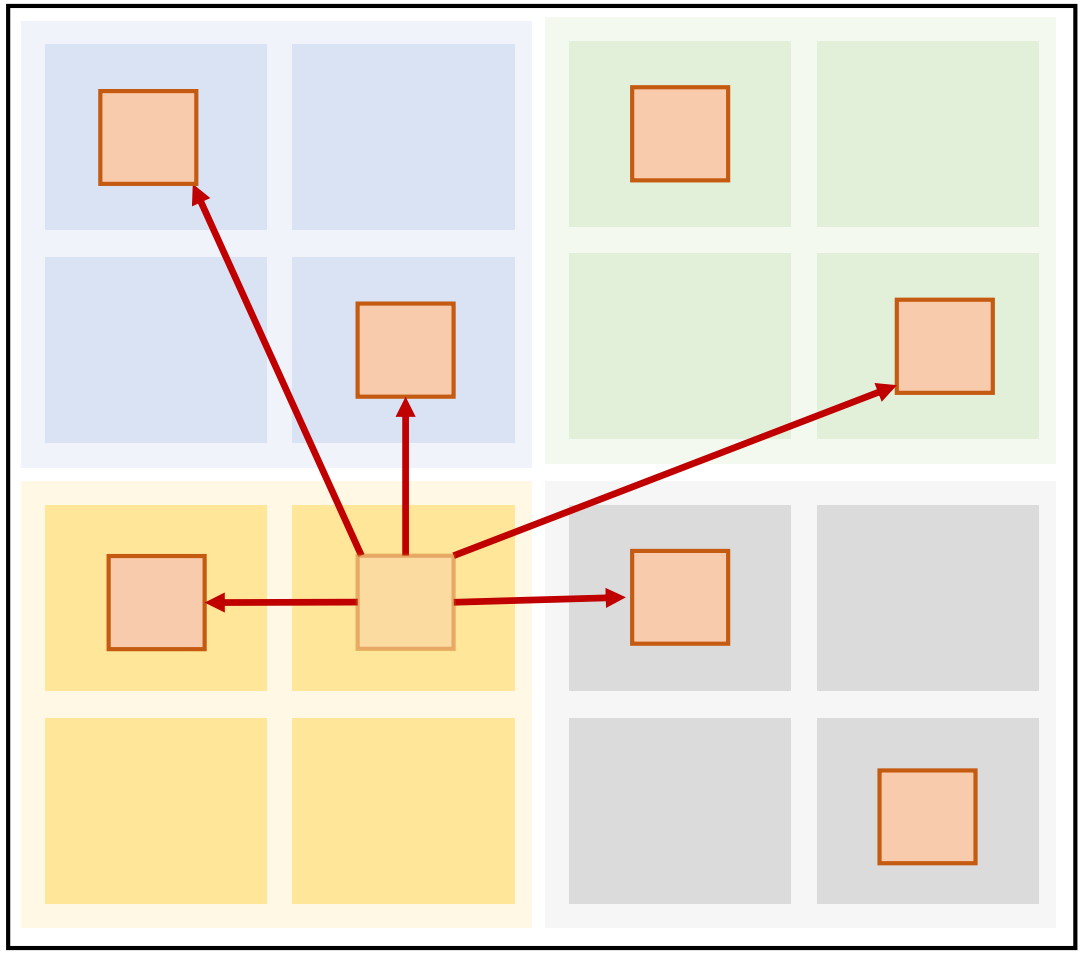
Emulator Regeneration



Target System

D-TIME

Emulator Regeneration



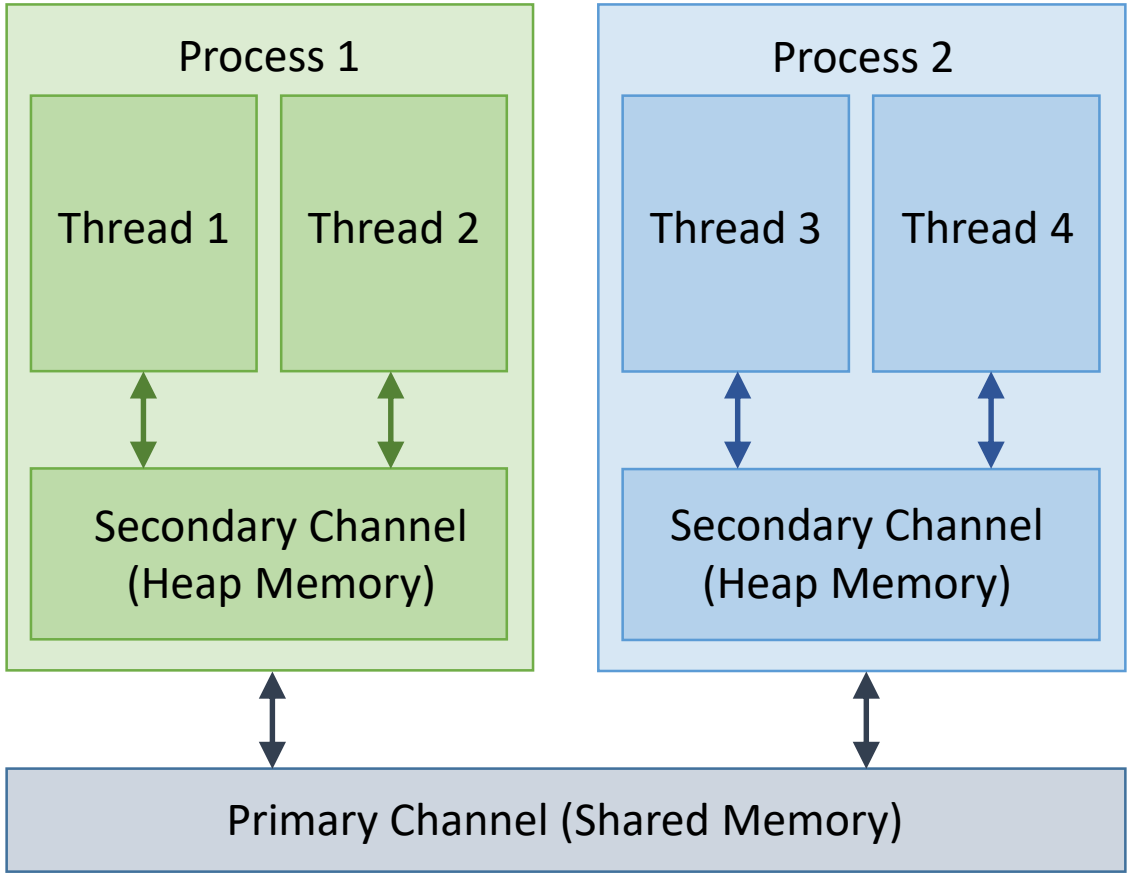
Target System

D-TIME

Emulator

Inter-emulator Comm. Channels

SCBC

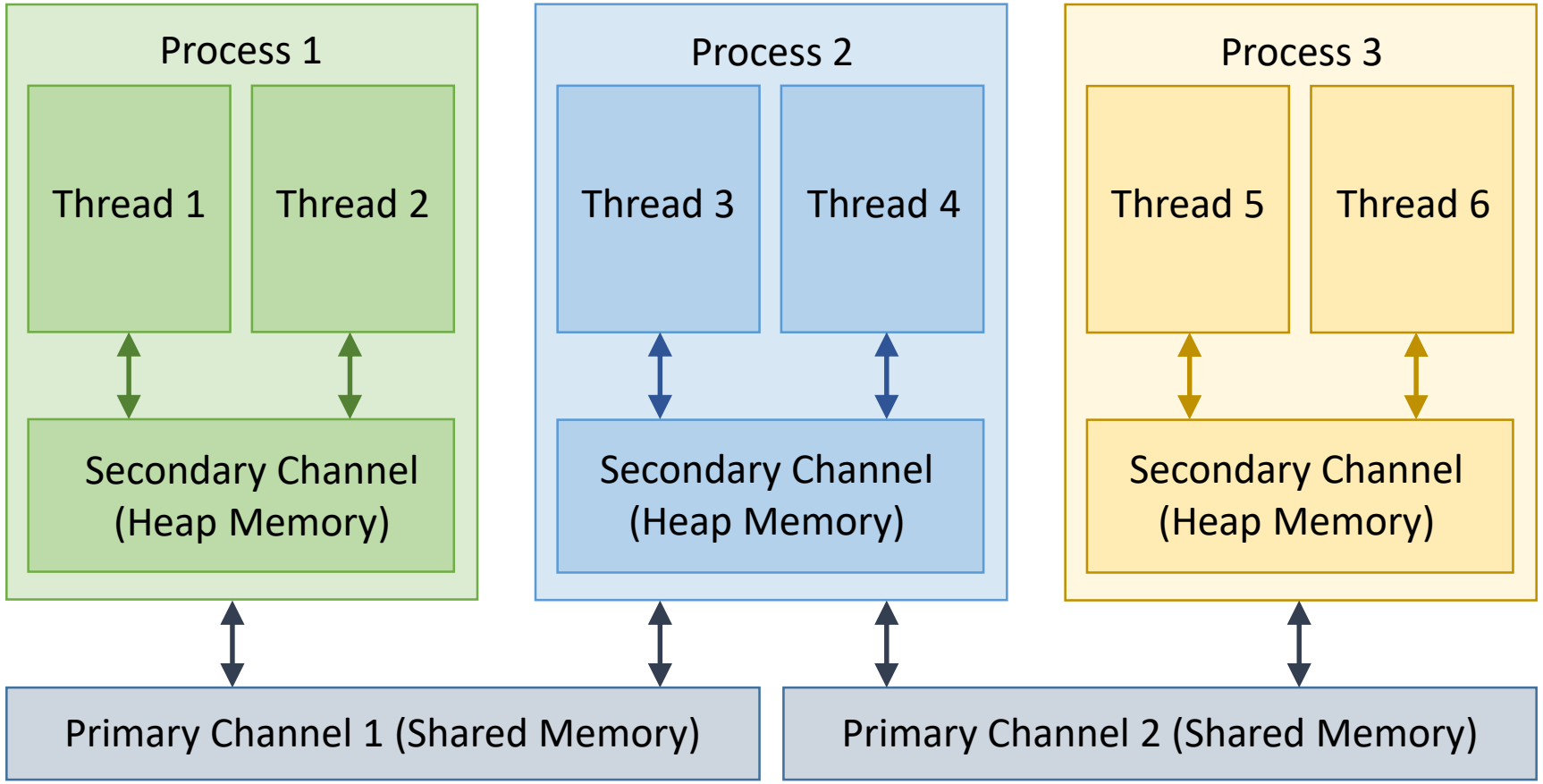


D-TIME

Emulator

Inter-emulator Comm. Channels

SCBC



Partially adopted from MalWASH (WOOT'16)

D-TIME

Emulator **Inter-emulator Comm. Channels** SCBC

Heap and Shared Memory

1. Commonly used ⟶ Hard to classify as malign
2. Once allocated/attached, lives until
   a) De-allocated by application or
   b) Process exit

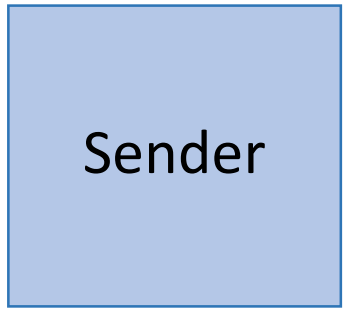
D-TIME

Emulator **Inter-emulator Comm. Channels** SCBC

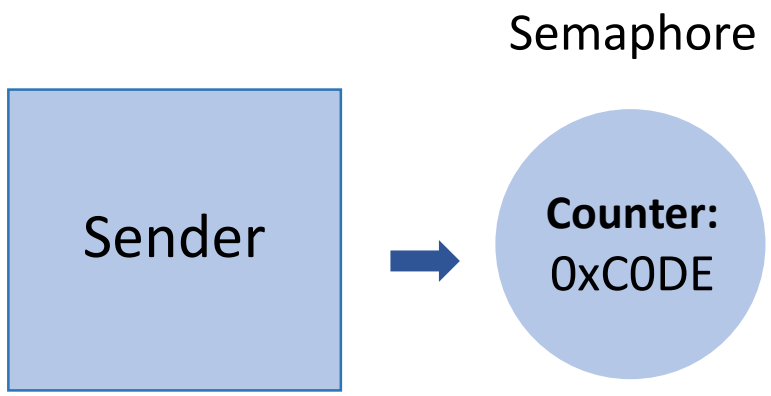
Unknown location!

1. First emulator comes up
   a) Attaches the shared memory
   b) Executes a chunk
   c) Exit
2. The second emulator comes up
   ✓ Has access rights to shared memory
   ? Doesn't know the starting address

SCBC
(Semaphore based Covert Broadcasting Channel)



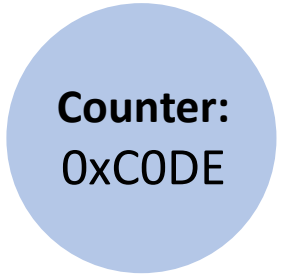
SCBC
(Semaphore based Covert Broadcasting Channel)



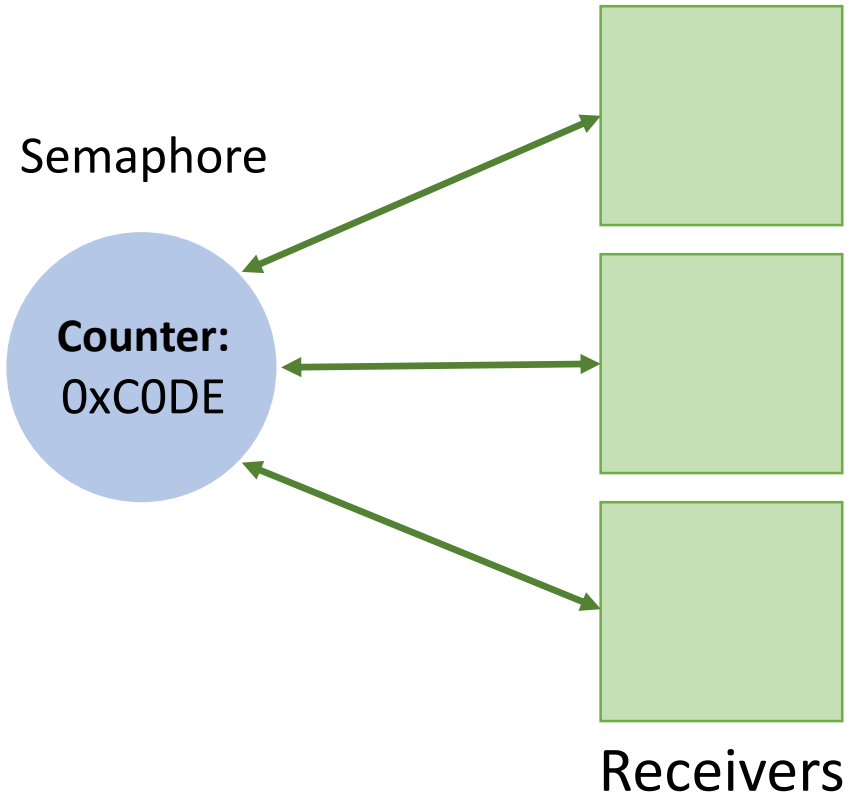
Creates a semaphore and set
semaphore-counter = data

SCBC
(Semaphore based Covert Broadcasting Channel)

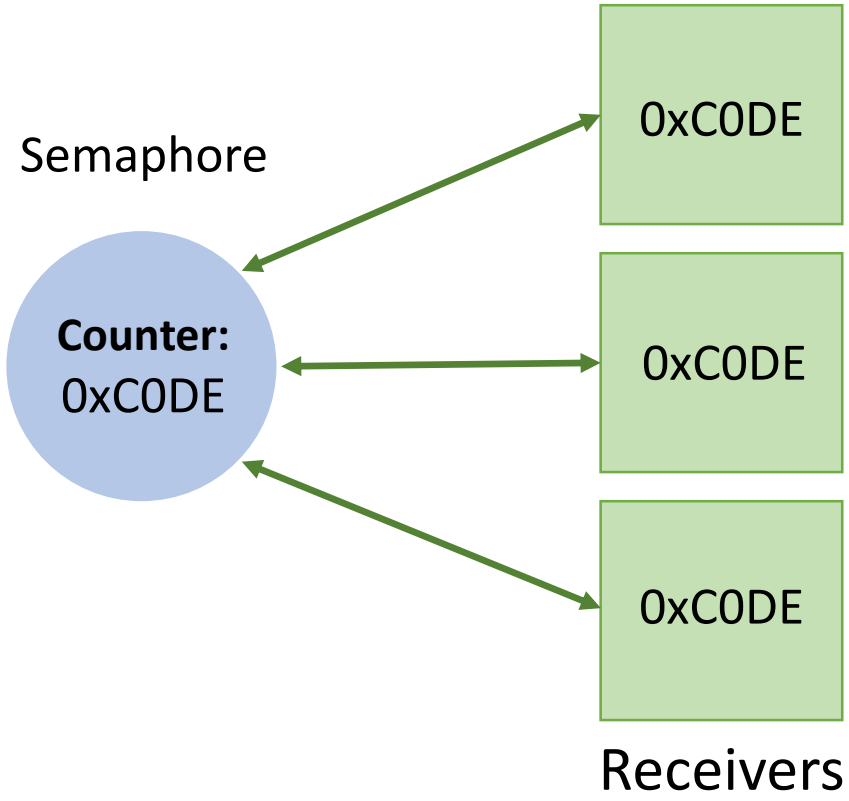
Semaphore



SCBC
(Semaphore based Covert Broadcasting Channel)



SCBC
(Semaphore based Covert Broadcasting Channel)



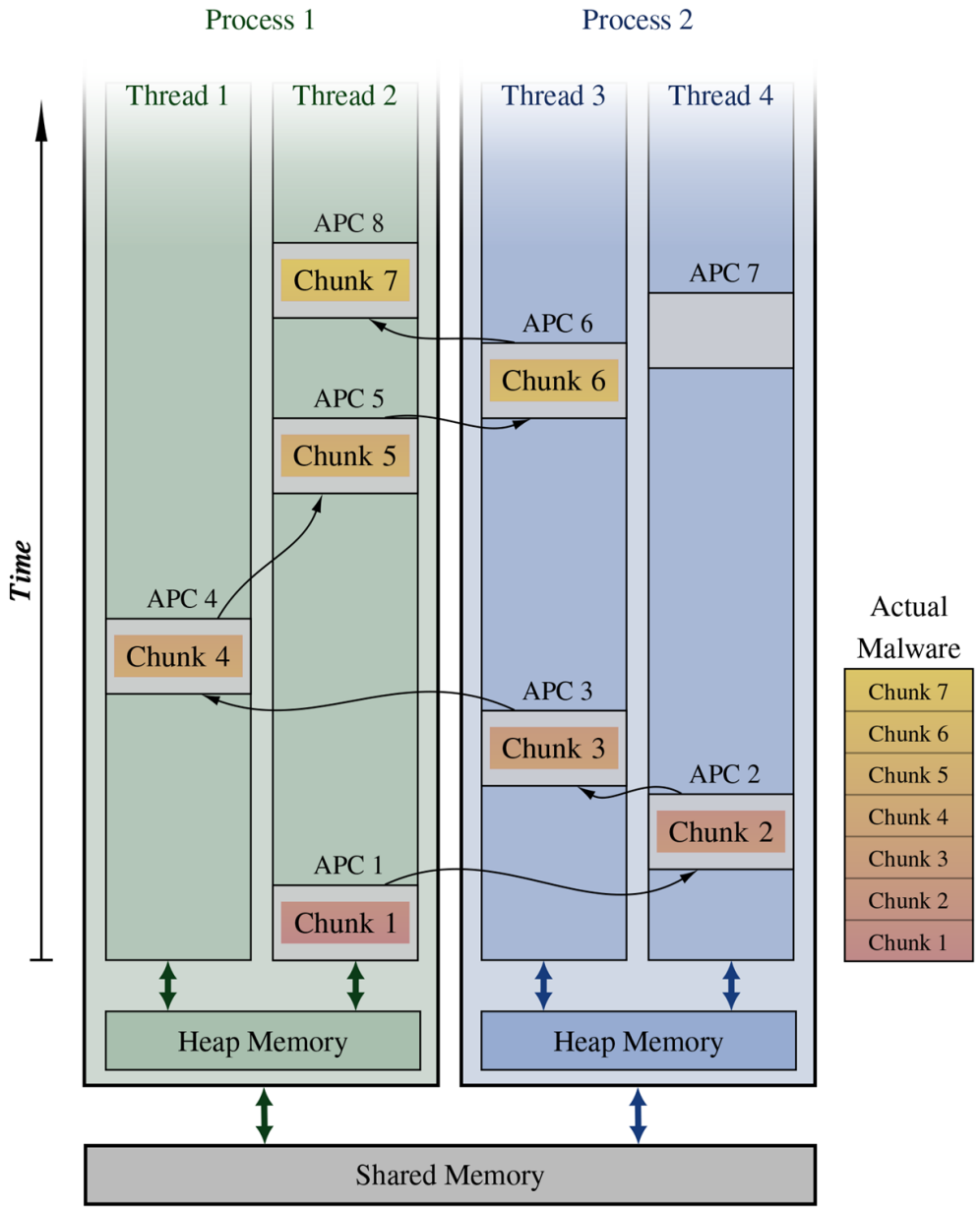
Works in Linux as well as Windows

SCBC
(Semaphore based Covert Broadcasting Channel)

- High Integrity
- Broadcasting
- Persistent
- As Convenient as a shared integer

Converts semaphore to a shared integer

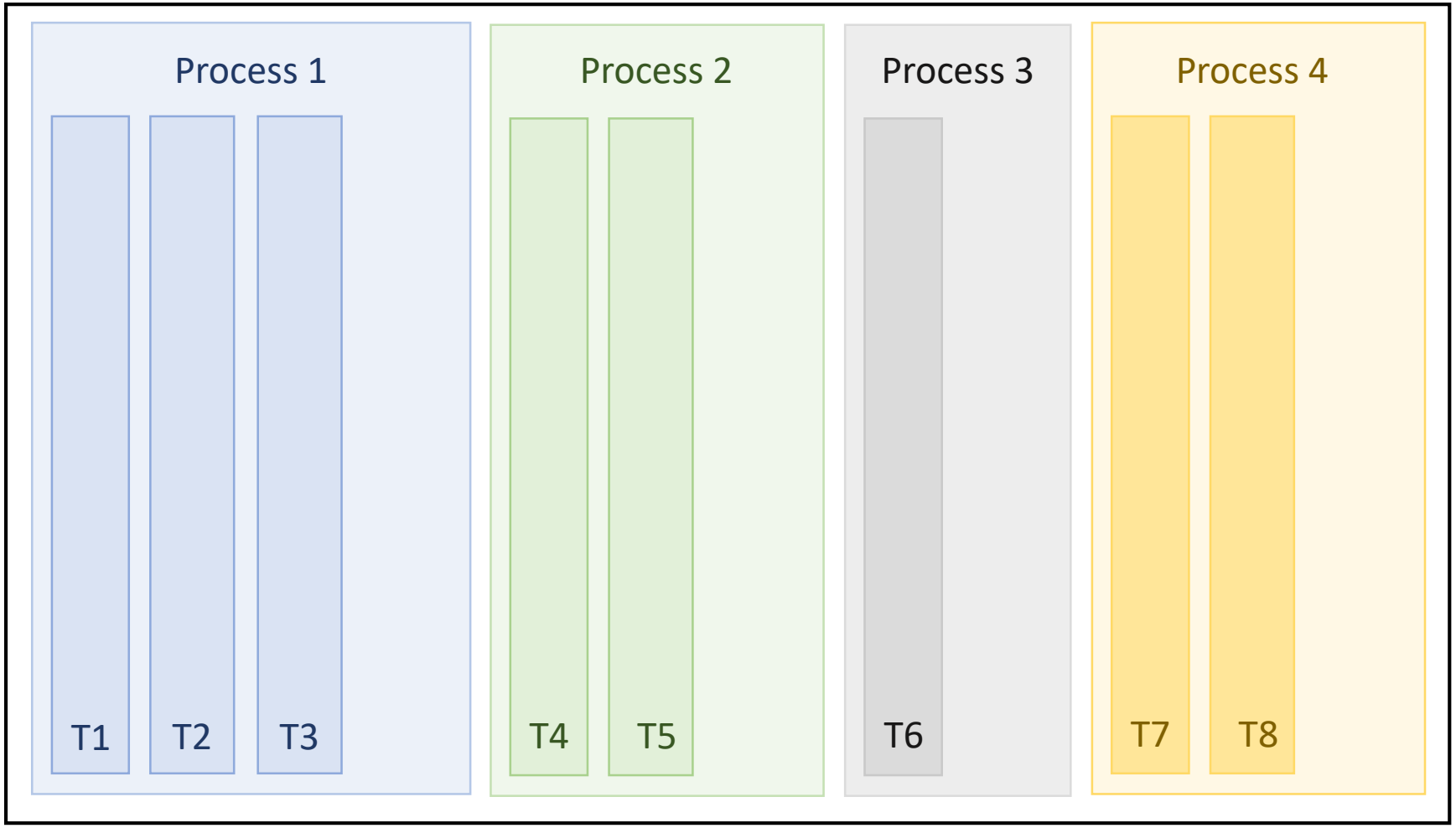
D-TIME Execution



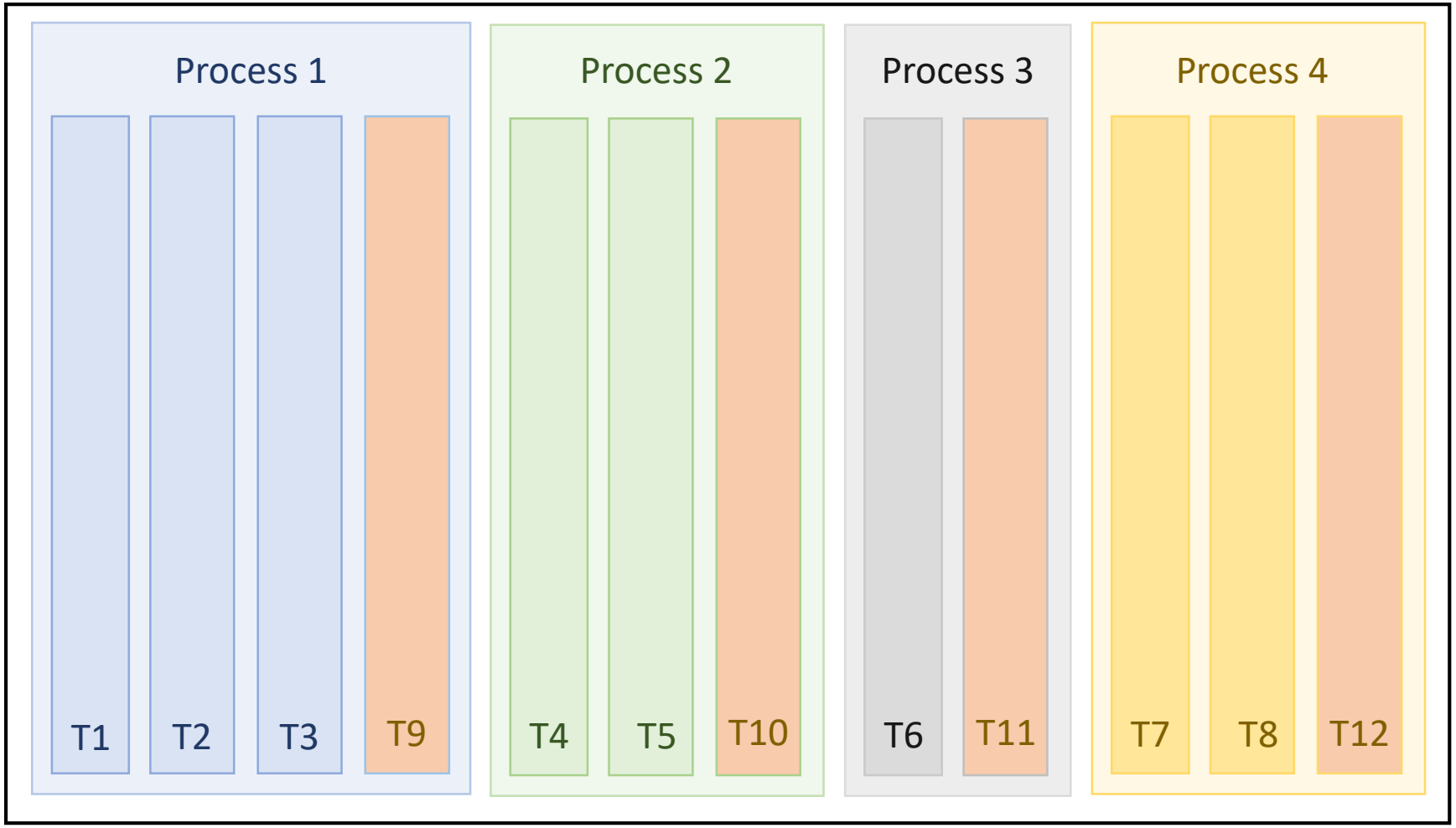
D-TIME - Summary

- Evades Detection by Distribution
- Performs Threadless execution
- Independent of Victim Process
- Completely Decentralized System
- High resilience

MalWASH – Phase 2

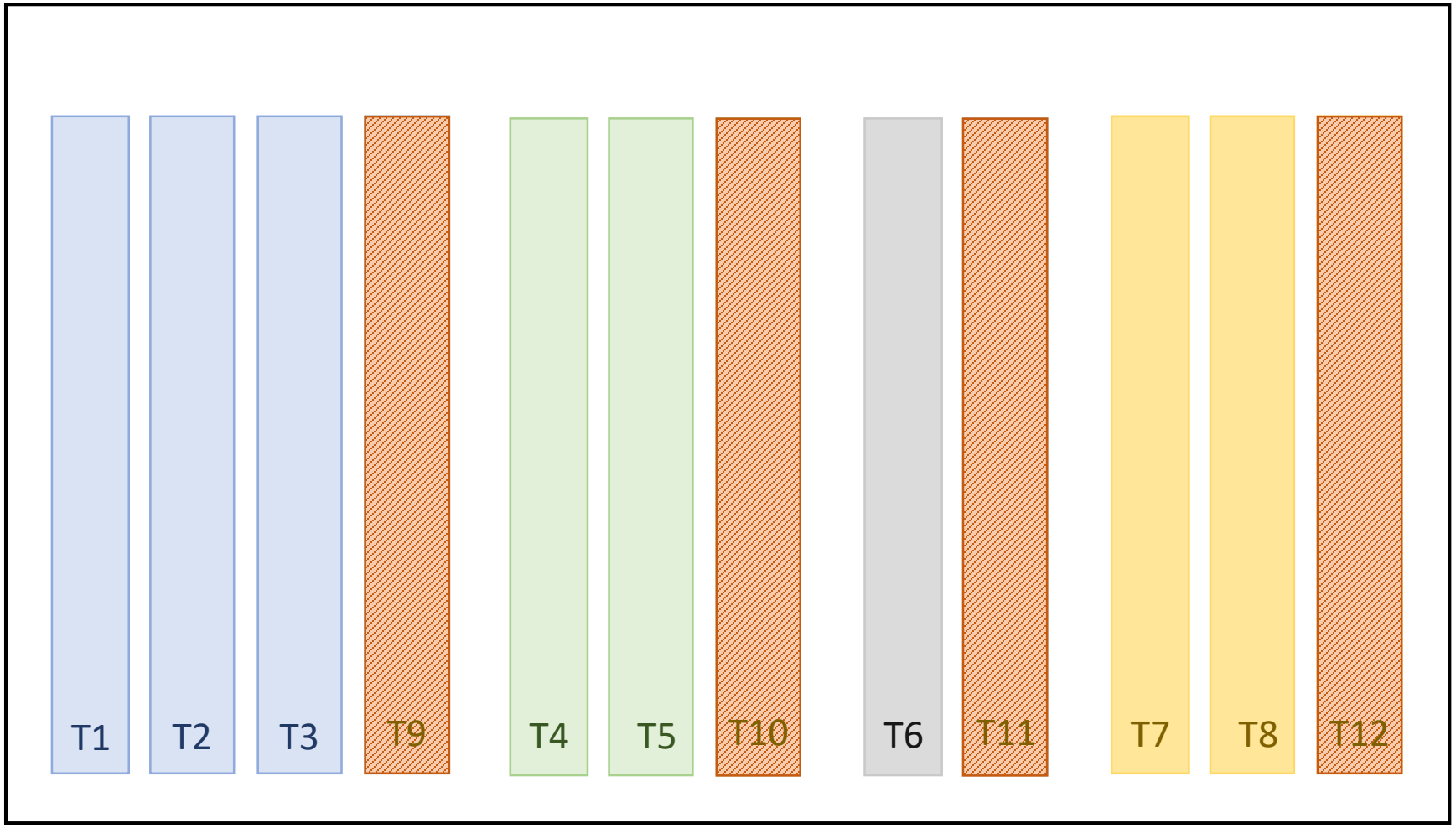


MalWASH – Phase 2



Creates threads in benign processes

MalWASH – Phase 2



Created threads are well exposed!

MalWASH vs D-TIME

	MalWASH	D-TIME
Needs to Create Threads	YES	NO
Has Dedicated Threads	YES	NO
Needs Administrative Privilege	YES	NO
Resilience	-	IMPROVED
Communication mechanism	-	IMPROVED

Evaluation & Results

Key-logger Offline

Key-logger Online

Backdoor

Ransomware

Screenshot malware

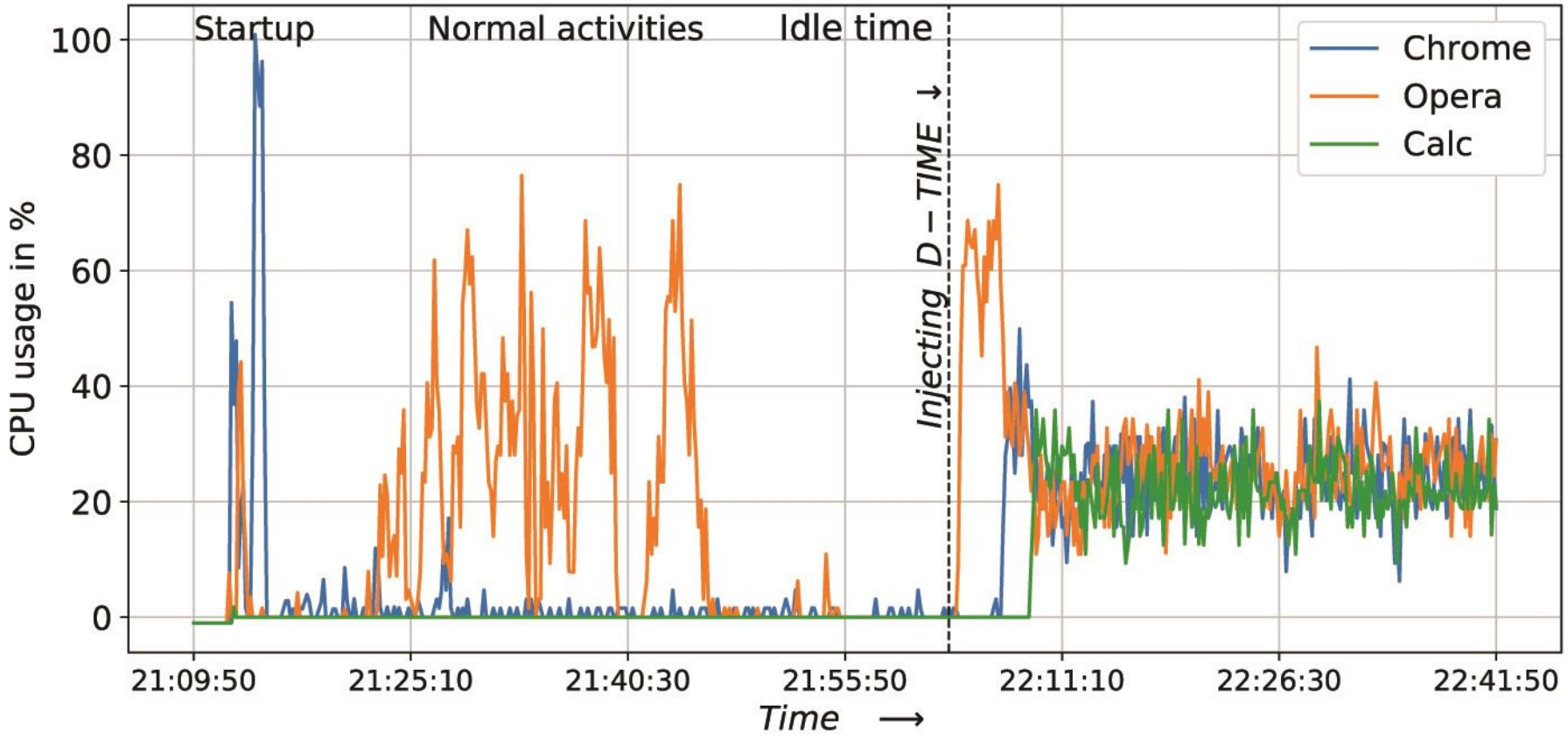
Window Monitor

Evaluation & Results

1. BitDefender
2. Norton
3. Kaspersky
4. WEBROOT
5. McAfee
6. ESET
7. Avast
8. AVG
9. Windows Defender
10. Avira

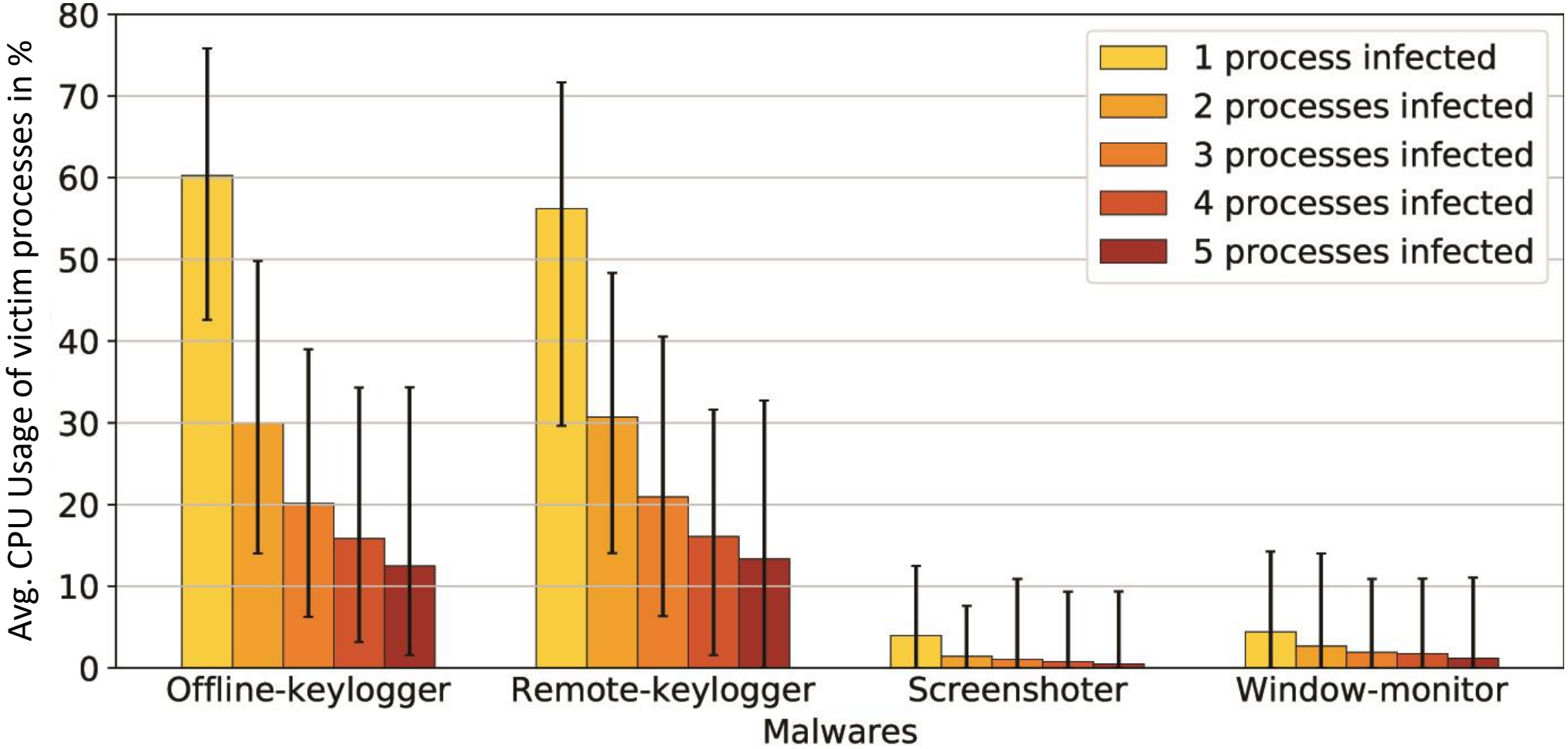
Evaluation & Results

CPU usage vs Time



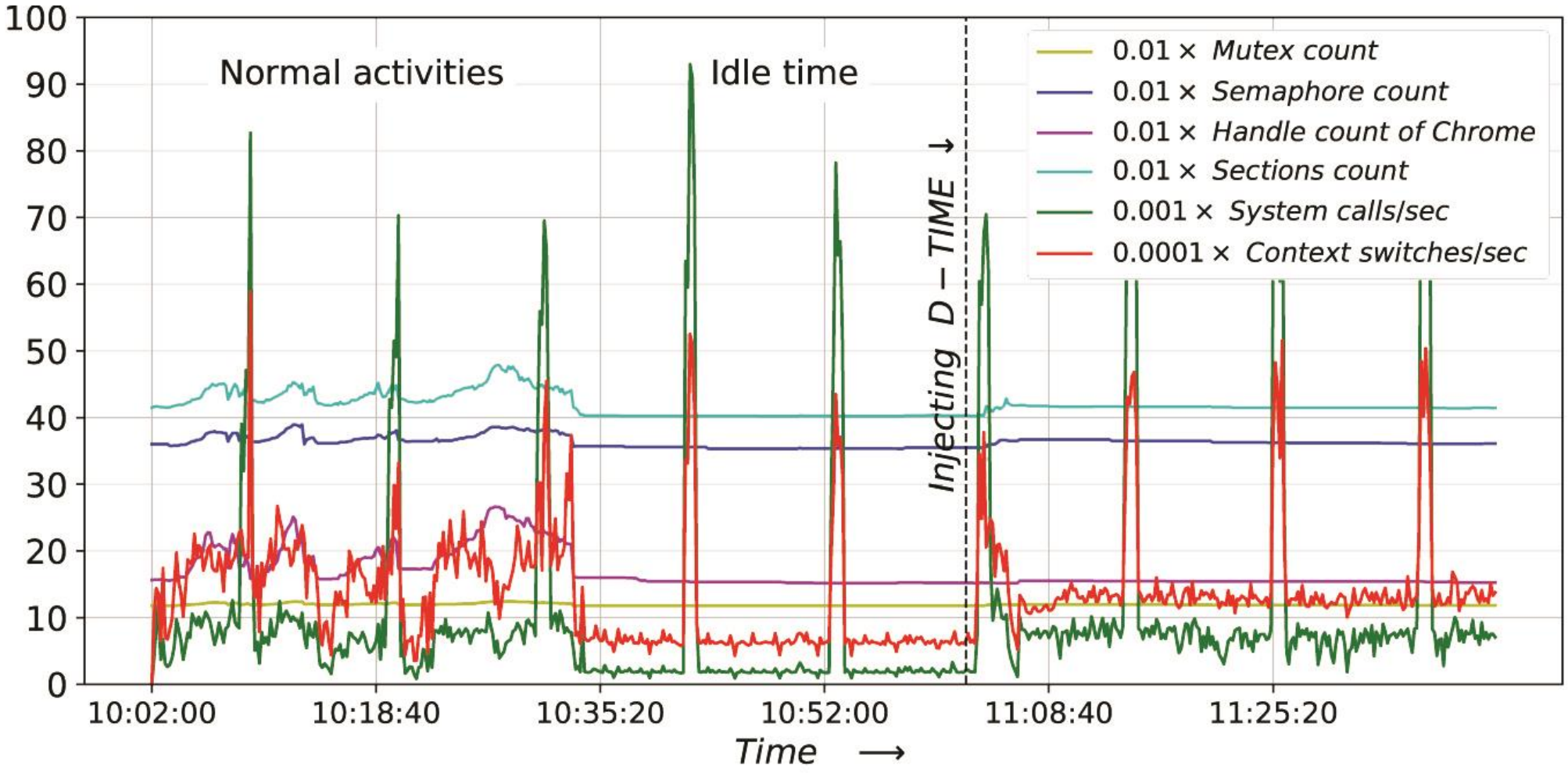
Evaluation & Results

CPU usage for different number of infected processes



Evaluation & Results

Performance Counters



Countermeasures

Prevention

- Redesign APC
- Create QPC (Queue Privilege Class)
- Limit Recursive Queuing
- Principle of minimum disclosure

Detection

- Detecting Comm. Channels
- Analyzing
  - CPU Usage
  - Context switches

Removal

- Removing all Emulators
- Destroying all Comm. Channels

Contributions

- D-TIME:
  Distributed, Decentralized, threadless malware execution framework
- SCBC (Semaphore based Covert Broadcasting Channel)
- Tested D-TIME using 6 Malware against 10 anti-virus
- Suggested counter measures

Artifact

Paper

Thank you