VULNERABILITY IS A LUCKY BUG

BY ARTEM SHISHKIN

DISCLAIMER

I am presenting the contents of this presentation in my personal capacity. The views expressed are solely my own and do not necessarily reflect the views of Intel Corporation or its affiliates.

WHOAMI

- Security researcher at Intel Corporation
- Intel STORM team member
- Windows kernel enthusiast

SPOT THE POTENTIAL BUG

```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

HERE IT IS

```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

HOW LUCKY THIS BUG IS SUPPOSED TO BE TO BECOME A SECURITY ISSUE?

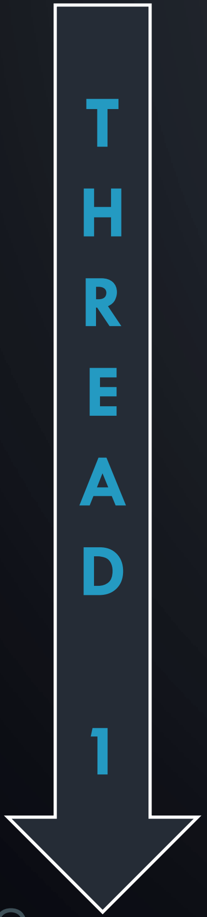
SINGLE THREADED ENVIRONMENT

- Not exploitable
  - Well, who is going to attack the value otherwise?

SINGLE THREADED ENVIRONMENT

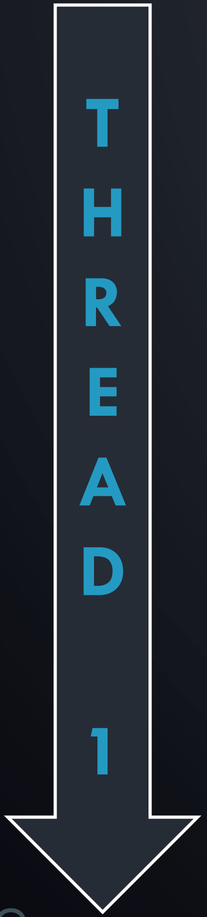
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

SINGLE THREADED ENVIRONMENT



```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

SINGLE THREADED ENVIRONMENT



```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

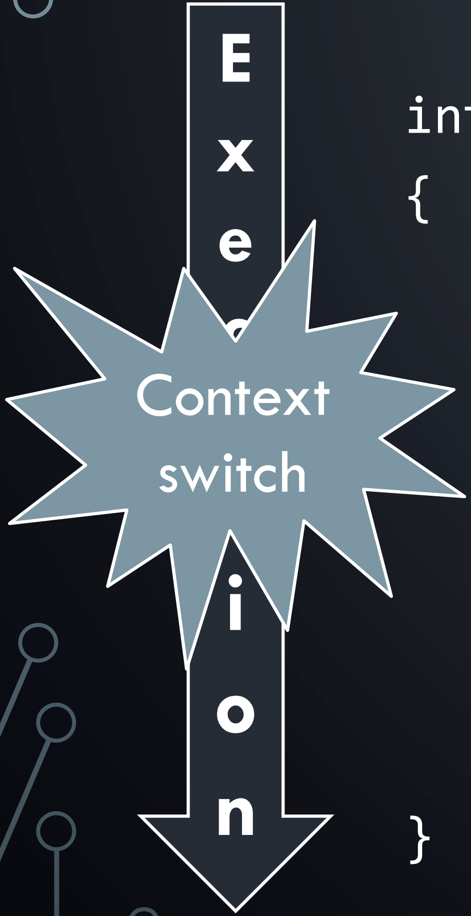
SINGLE THREADED ENVIRONMENT

```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT

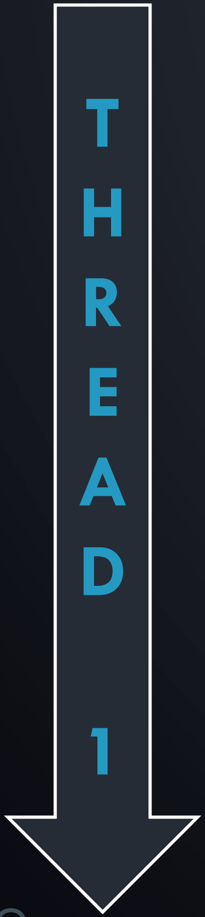
- Code is executed within the threads
- Thread is a basic scheduling unit on Windows
- Thread quantum is large (measured in milliseconds)
  - The race window is too small compared to the quantum duration
- Not exploitable in this case
  - Could be in a case with scheduling interruption in the race window

MULTITHREADED ENVIRONMENT



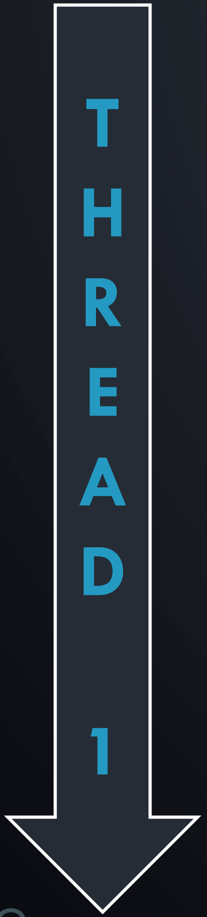
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



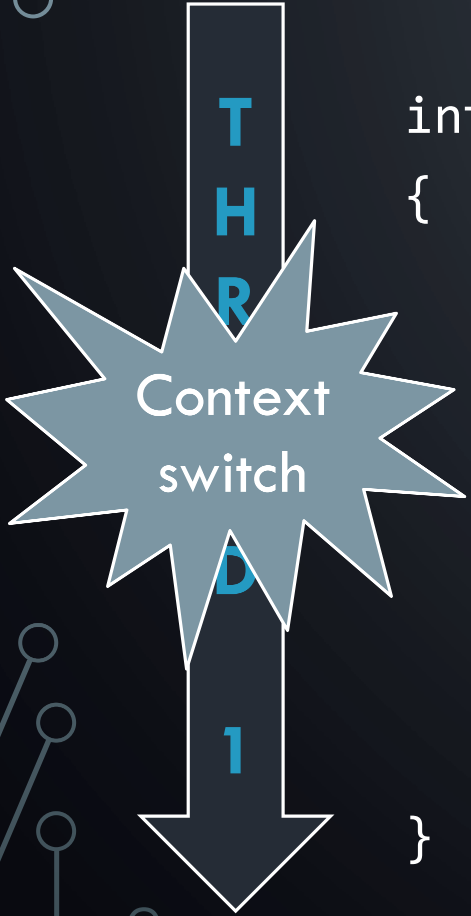
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



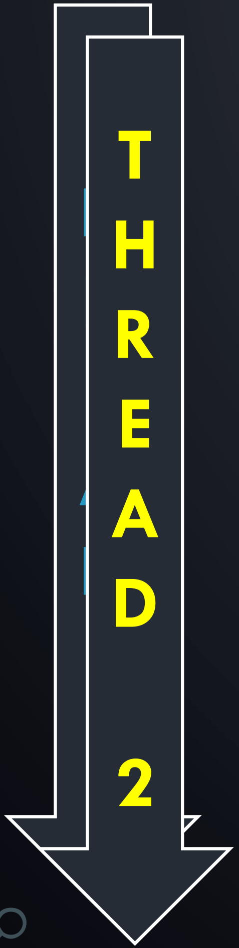
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



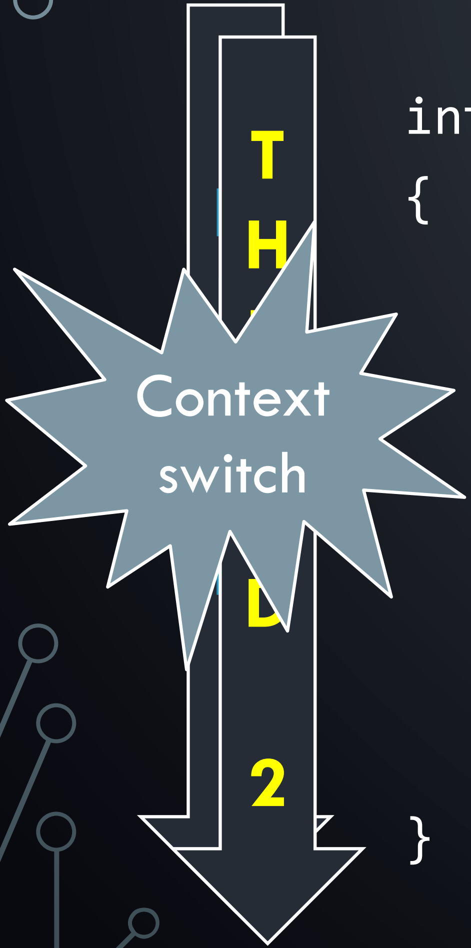
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT

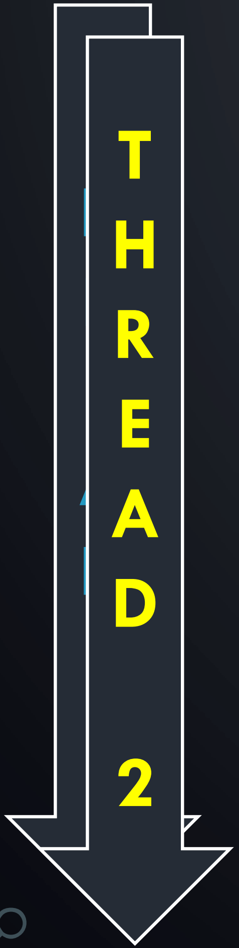


```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT

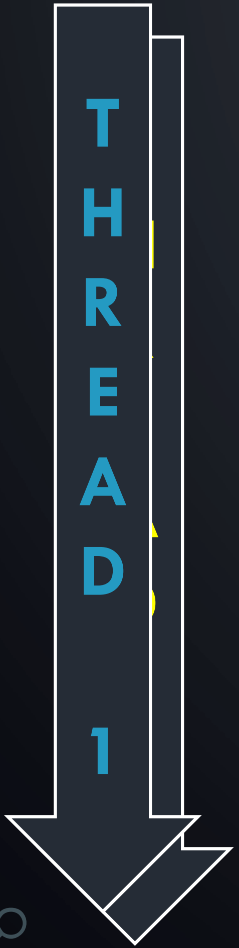
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



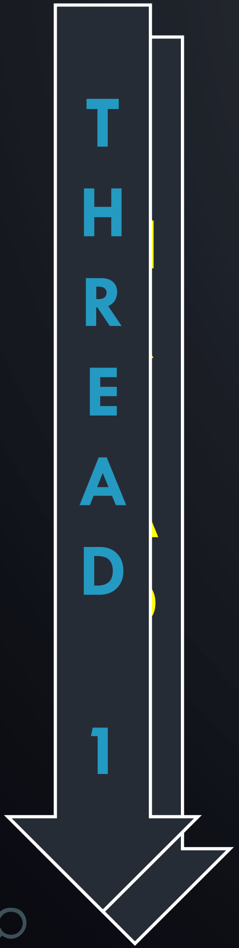
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT



```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED ENVIRONMENT

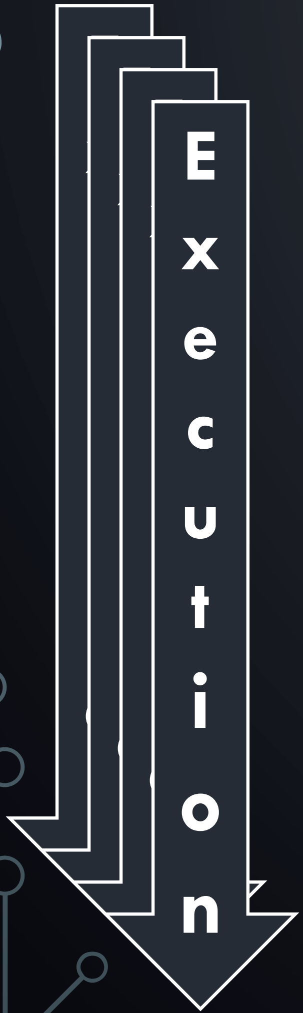


```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    sleep(3000);
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED MULTICORE ENVIRONMENT

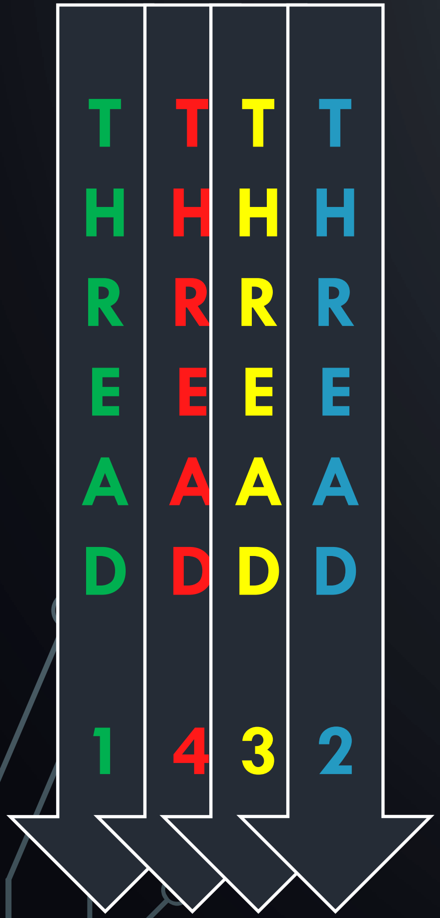
- CPU cores execute code simultaneously
- So we can schedule thread execution on different CPU cores
- So, exploitable?

MULTITHREADED MULTICORE ENVIRONMENT



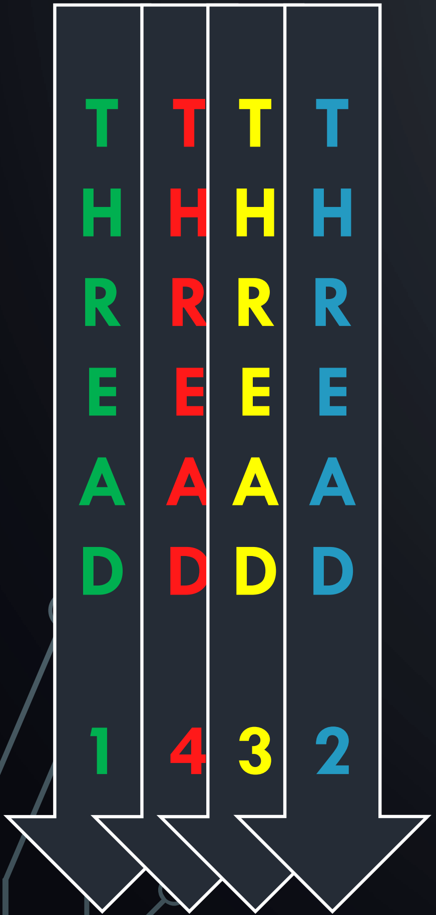
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED MULTICORE ENVIRONMENT



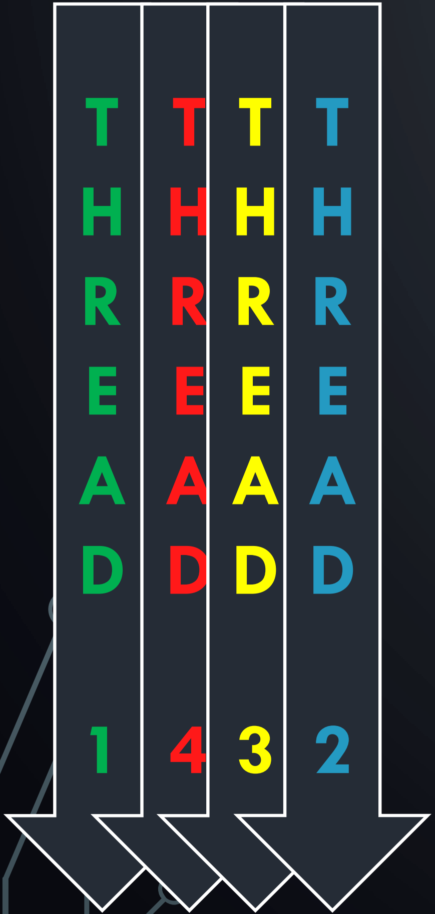
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED MULTICORE ENVIRONMENT



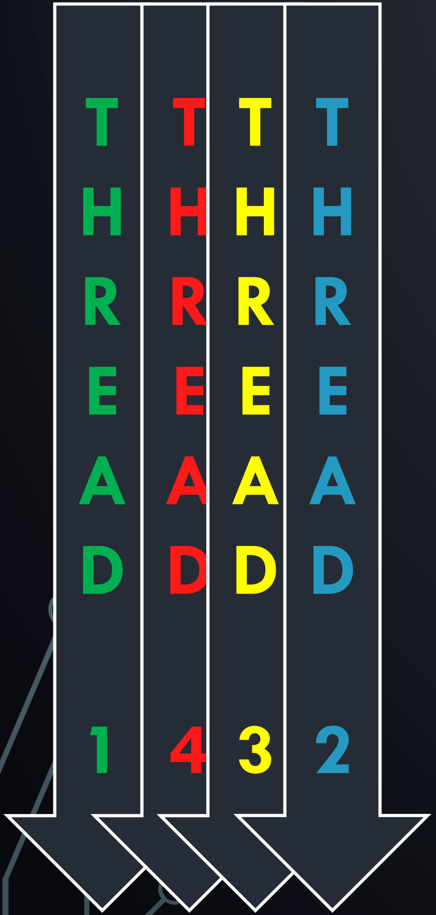
```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED MULTICORE ENVIRONMENT



```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

MULTITHREADED MULTICORE ENVIRONMENT

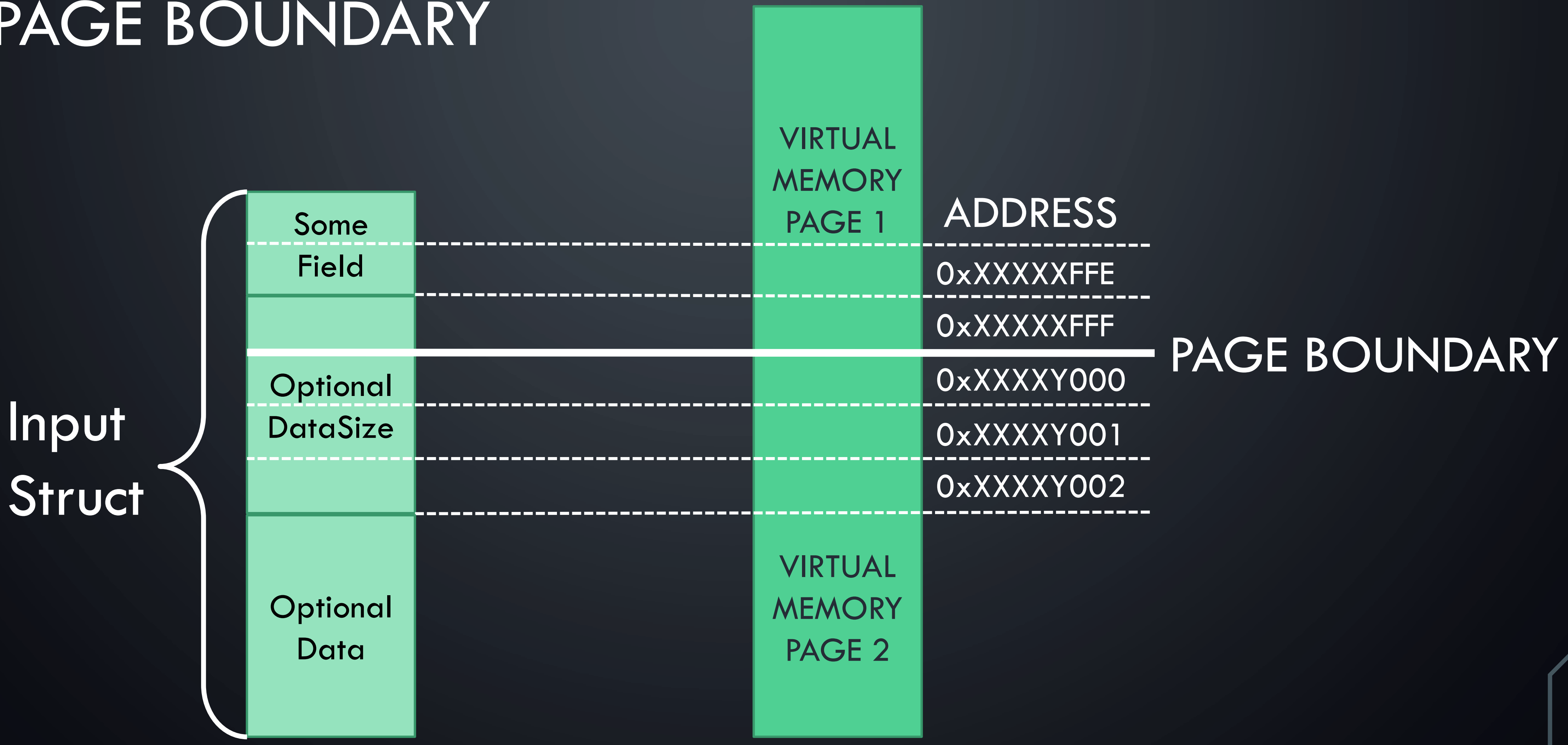


```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    if (pInputStruct->OptionalDataSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData,
        pInputStruct->OptionalDataSize);

    return 0;
}
```

RACE COMPETITION

- Thread 1 executes the vulnerable code
- Thread 2 is trying to modify the controlled data in the structure
- Slow down the first thread
  - Place the data on a page boundary for the first thread
  - Keep flipping the attacked value in the second thread
  - Flip only the part that is spread on a single page

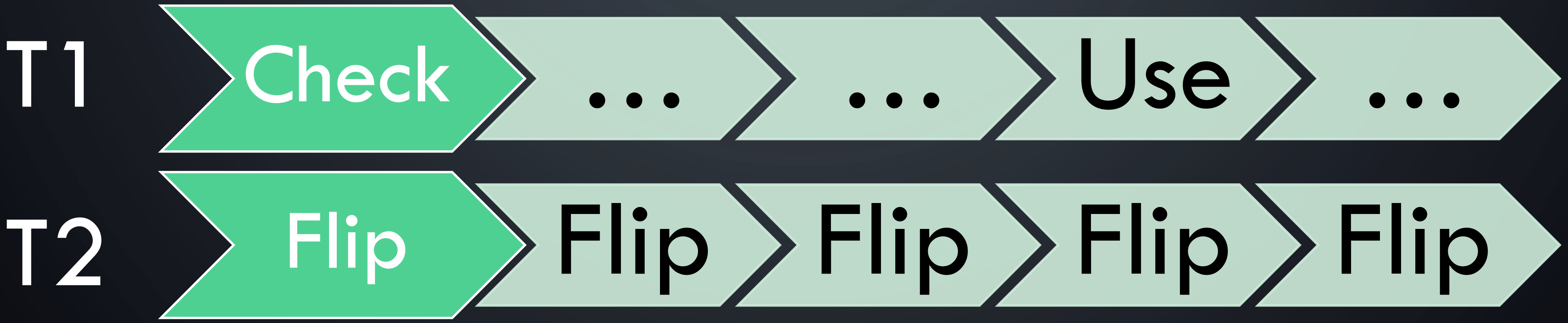
PAGE BOUNDARY



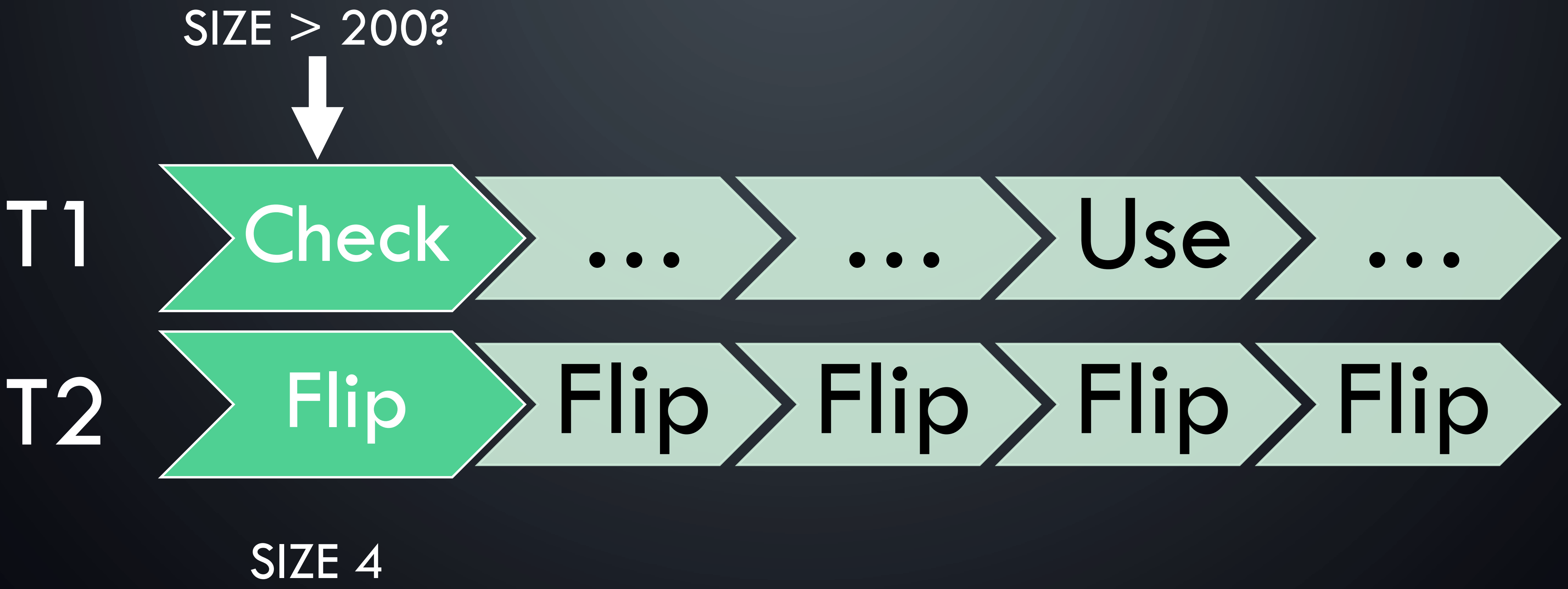
MEMORY LAYOUT

```
0: kd> dt someapp!_INPUT_TYPE 0x00000207`f0590ffb
   +0x000 SomeField              : 2
   +0x004 OptionalDataSize   : 4
   +0x008 OptionalData         : [1]  "A“

0: kd> db 0x00000207`f0590ff0
00000207`f0590ff0   00 00 00 00 00 00 00 00-00 00 00 02 00 00 00 04  ........
00000207`f0591000  00 00 00 41 41 41 41 41-41 41 41 41 41 41 41 41
...AAAAAAAAAAAAA
00000207`f0591010  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
00000207`f0591020  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
```

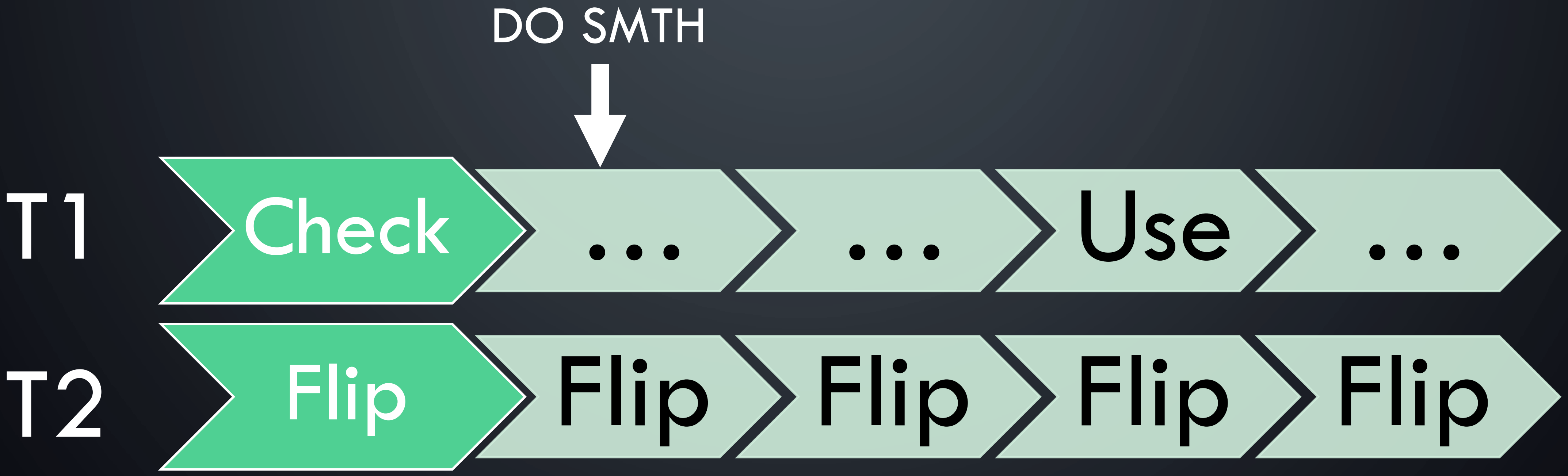
RACE COMPETITION



RACE COMPETITION

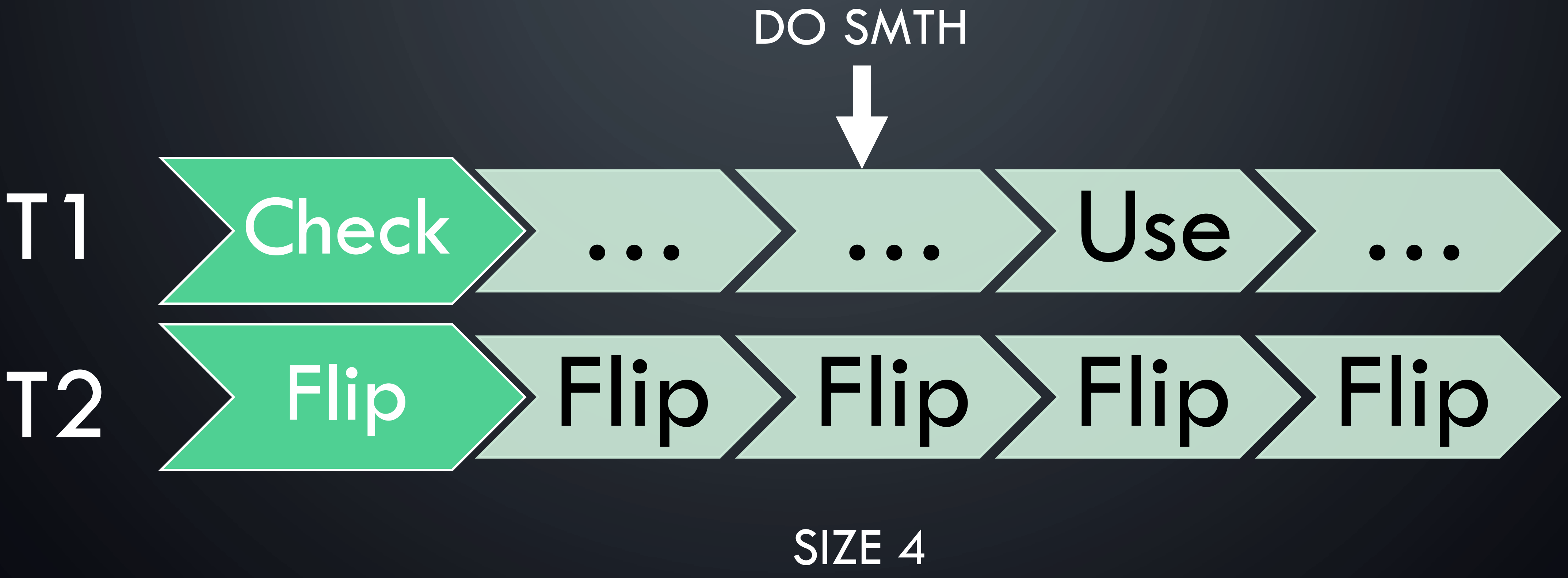


RACE COMPETITION

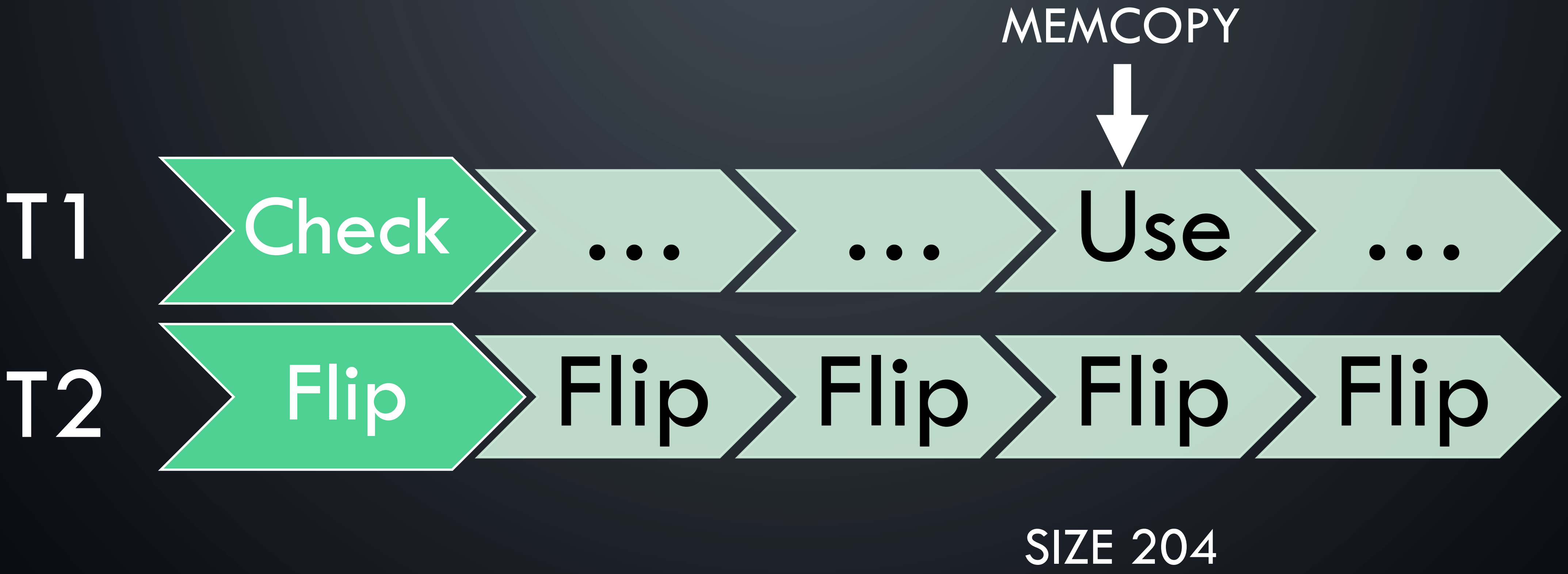


SIZE 204

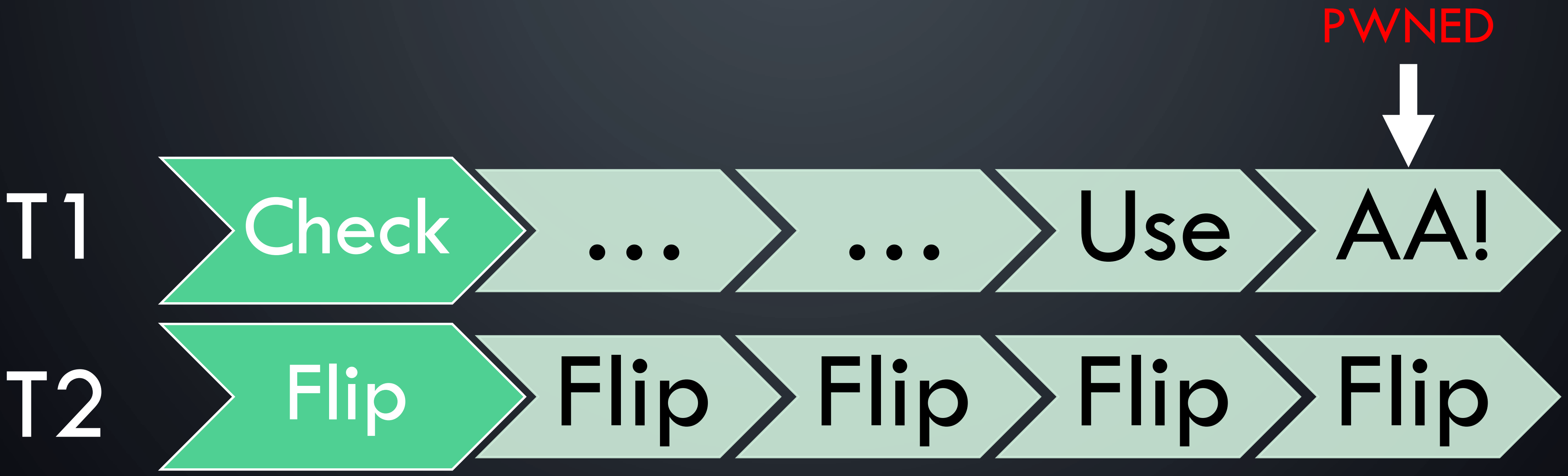
RACE COMPETITION



RACE COMPETITION

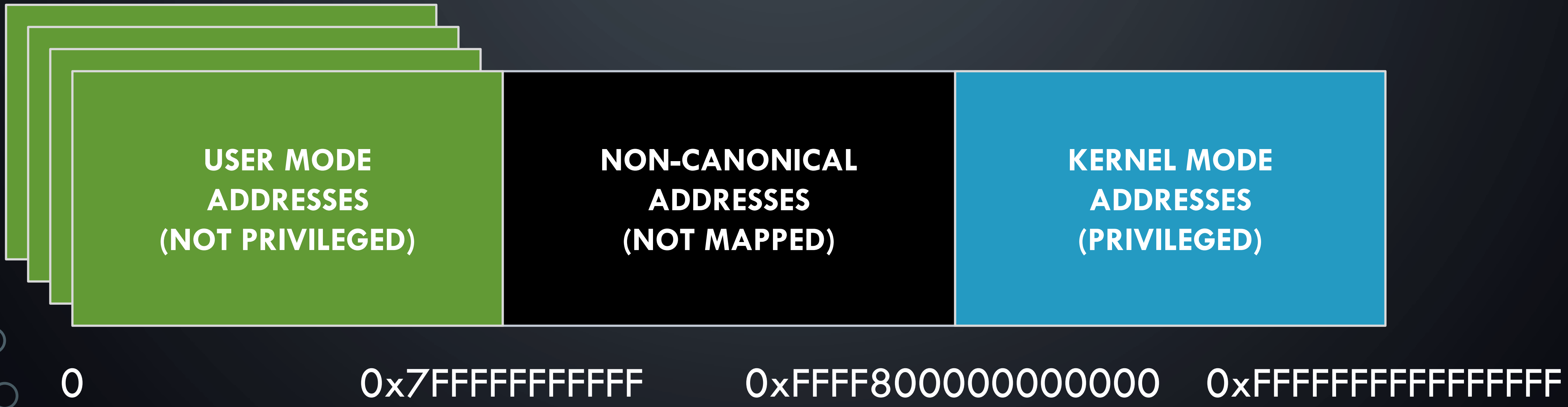


RACE COMPETITION



SIZE 4

WINDOWS VIRTUAL ADDRESS SPACE LAYOUT (X64)


USER MODE ADDRESSES (NOT PRIVILEGED)
NON-CANONICAL ADDRESSES (NOT MAPPED)
KERNEL MODE ADDRESSES (PRIVILEGED)
0
0x7FFFFFFFFFF
0xFFFF800000000000
0xFFFFFFFFFFFFFFFF

DEMO

LET’S WINDBG THIS

WINNER WINNER CHICKEN DINNER

- This doesn’t lead to the code execution anyway
- Chaining with some other bug needed
- Every exploitation path is unique
  - Depends on the environment
  - Depends on input limitation
  - Depends on reachability

CODE EXEC GAINED

- What’s the impact?
  - Privilege escalation
    - Remote code execution
    - Guest to admin
    - User to kernel
  - Trust boundary violation
    - Sensitive data leak
  - Media impact

BUT IT HAS TO BE FOUND FIRST

DETECTION PRINCIPLES

- Detect consecutive memory accesses
- Static
  - Lack of runtime info
- Dynamic
  - Coverage issues

DYNAMIC DETECTION APPROACH

- This is an example for Windows kernel vulnerabilities
- Using the full system emulation
  - Bochs (bochspwn)
  - Simics
- Using the live system
  - SMAPwn

BOCHSPWN

- By Mateusz "j00ru" Jurczyk and Gynvael Coldwind
  - https://research.google.com/pubs/archive/42189.pdf
- Slow but reliable
- Open source
  - https://github.com/googleprojectzero/bochspwn
- Can also try info leaks

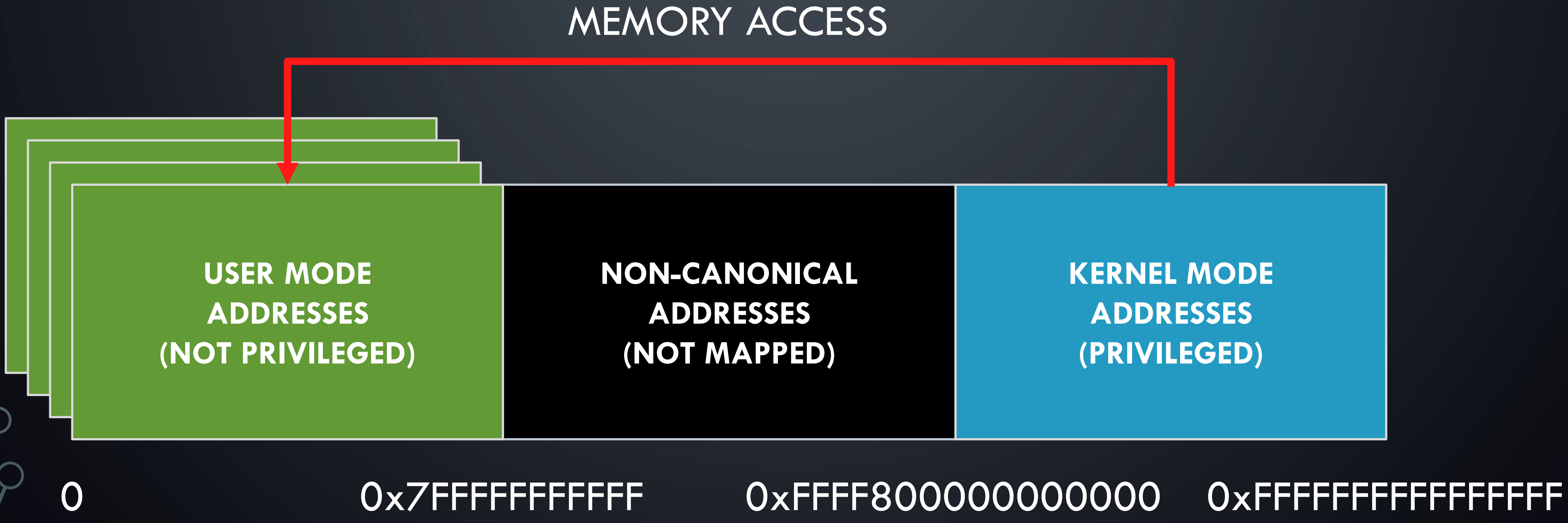
SIMICS BASED TRACKER

- BSDaemon and NadavCh did that using Simics
  - https://github.com/rrbranco/poc_gtfo/blob/master/pocorgtfo15.pdf
- Simics is not a public tool

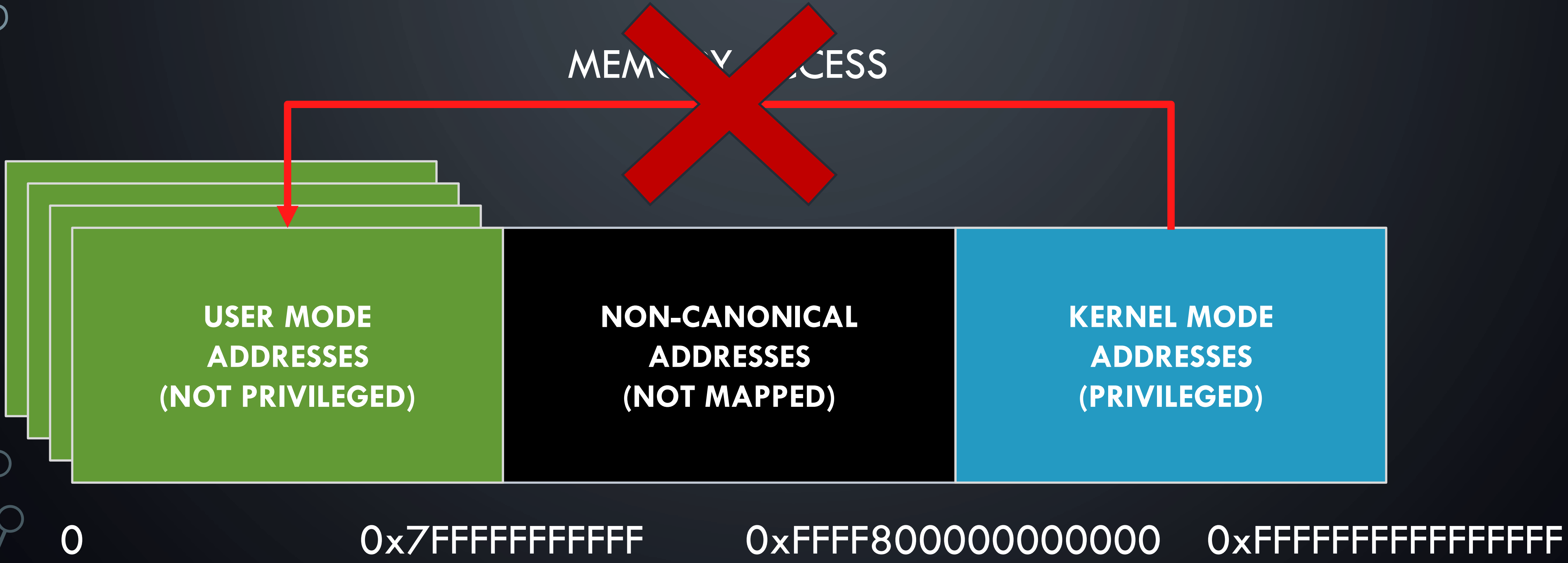
SMAPWN WINDBG PLUGIN

- By me
- Thoroughly described at H2HC 2018
  - https://github.com/h2hconference/2018
- The plugin is still not released
  - But can be easily reproduced

SUPERVISOR MODE ACCESS PREVENTION OFF


MEMORY ACCESS
USER MODE ADDRESSES (NOT PRIVILEGED)
NON-CANONICAL ADDRESSES (NOT MAPPED)
KERNEL MODE ADDRESSES (PRIVILEGED)
0
0x7FFFFFFFFFFF
0xFFFF800000000000
0xFFFFFFFFFFFFFFFF

SUPERVISOR MODE ACCESS PREVENTION ON


MEM   Y   CESS
USER MODE ADDRESSES (NOT PRIVILEGED)
NON-CANONICAL ADDRESSES (NOT MAPPED)
KERNEL MODE ADDRESSES (PRIVILEGED)
0
0x7FFFFFFFFFFF
0xFFFF800000000000
0xFFFFFFFFFFFFFFFF

DEMO

WINDBG AGAIN

SUMMARY

- The bug has to be found
- The bug has to be reachable
- The bug has to be controlled
- The environment has to be aligned
- Security premise must be broken
- Someone must care about it

THE FIX

```
int CaptureInput(PINPUT_TYPE pInputStruct)
{
    int OptionalSize = pInputStruct->OptionalDataSize;
    if (OptionalSize > MAX_OPTIONAL_DATA_SIZE)
        return -EINVAL;
    memcpy(pTarget, pInputStruct->OptionalData, OptionalSize);

    return 0;
}
```

THANKS AND HAPPY FISHING!