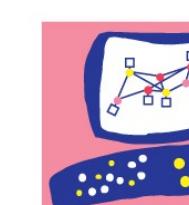
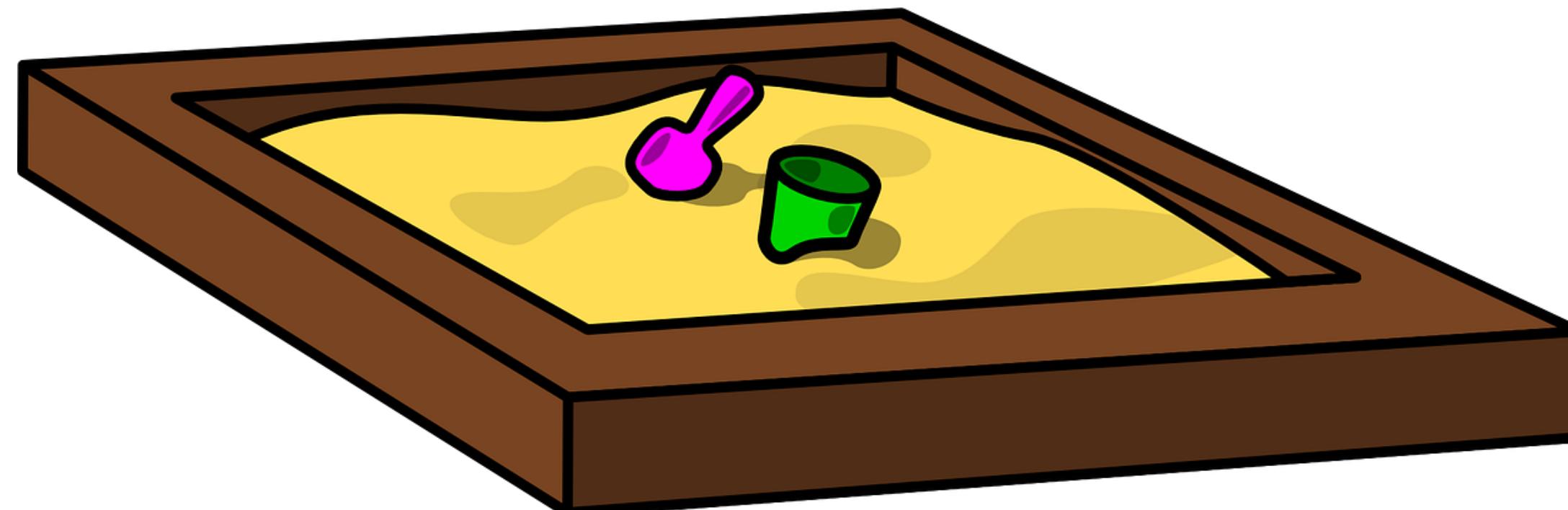


Defeating Sandbox Evasion

How to Increase Successful Emulation Rate in Your
Virtual Environment



Check Point®
SOFTWARE TECHNOLOGIES LTD.

Who?



Alexander Chailytko

Team Leader

alexanderc@checkpoint.com



Stanislav Skuratovich

Malware Researcher

stanislavsk@checkpoint.com

What it is all about?

Cuckoo Sandbox

- Detection/evasion techniques
- Proposed fixes

Virtual Environment

- Detection techniques
- Proposed fixes

InviZzzible

- Contains detection/evasion techniques for different environments
- Configurable through JSON files
- Developed for assessment of internal (your) virtual systems

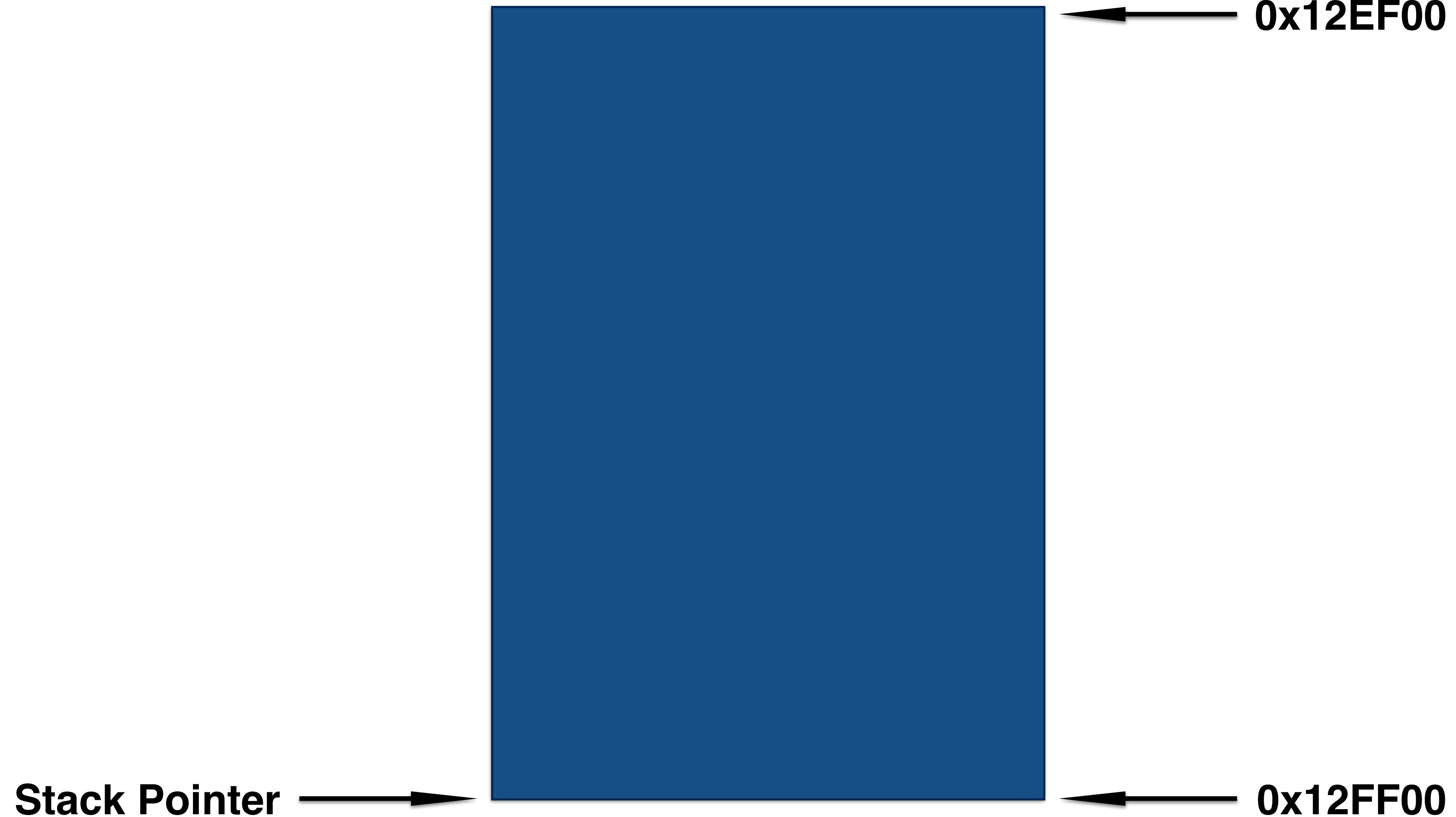
Cuckoo Sandbox



Info

Cuckoo: Monitor

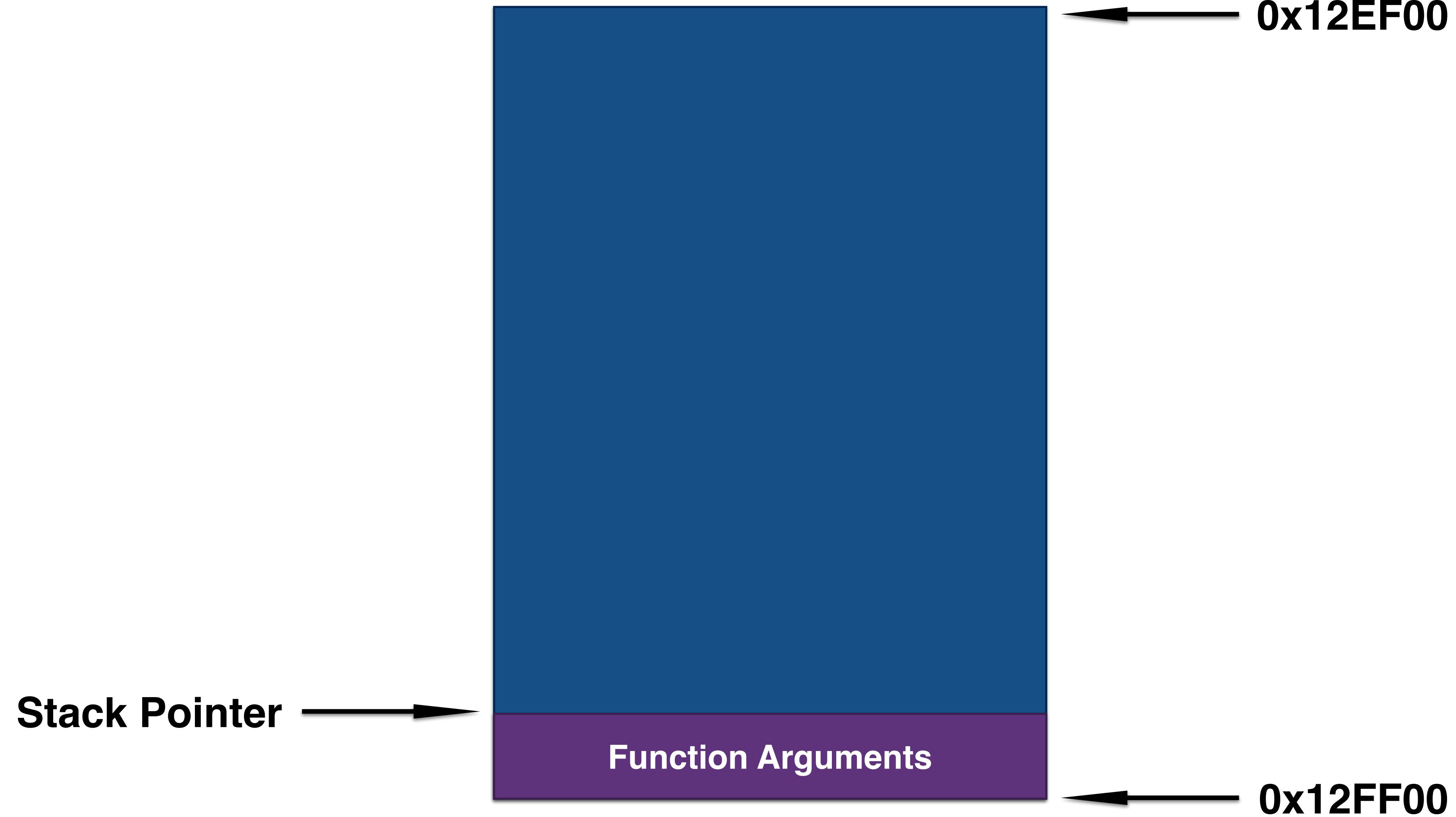
Unbalanced Stack



Info

Cuckoo: Monitor

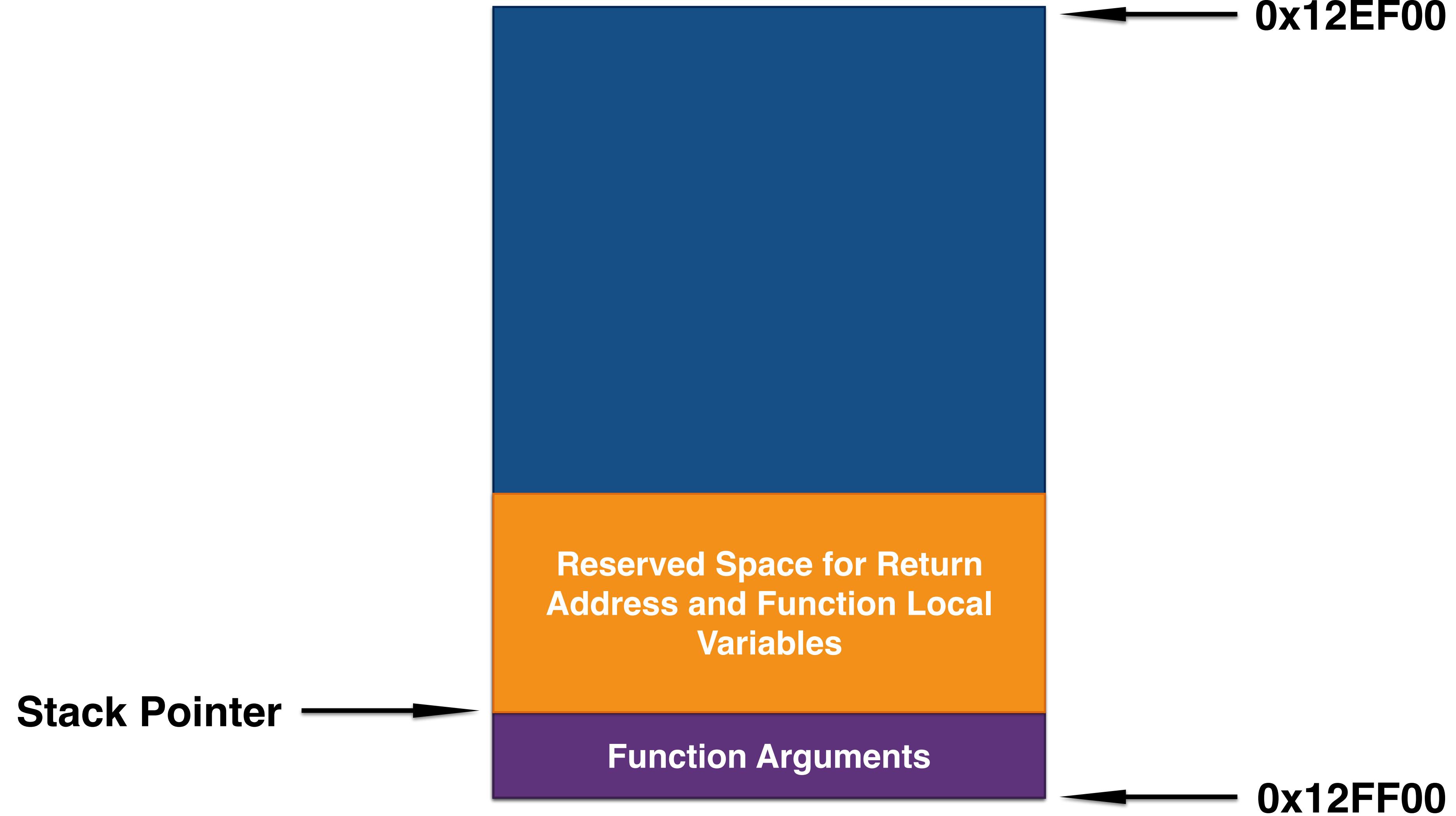
Unbalanced Stack



Info

Cuckoo: Monitor

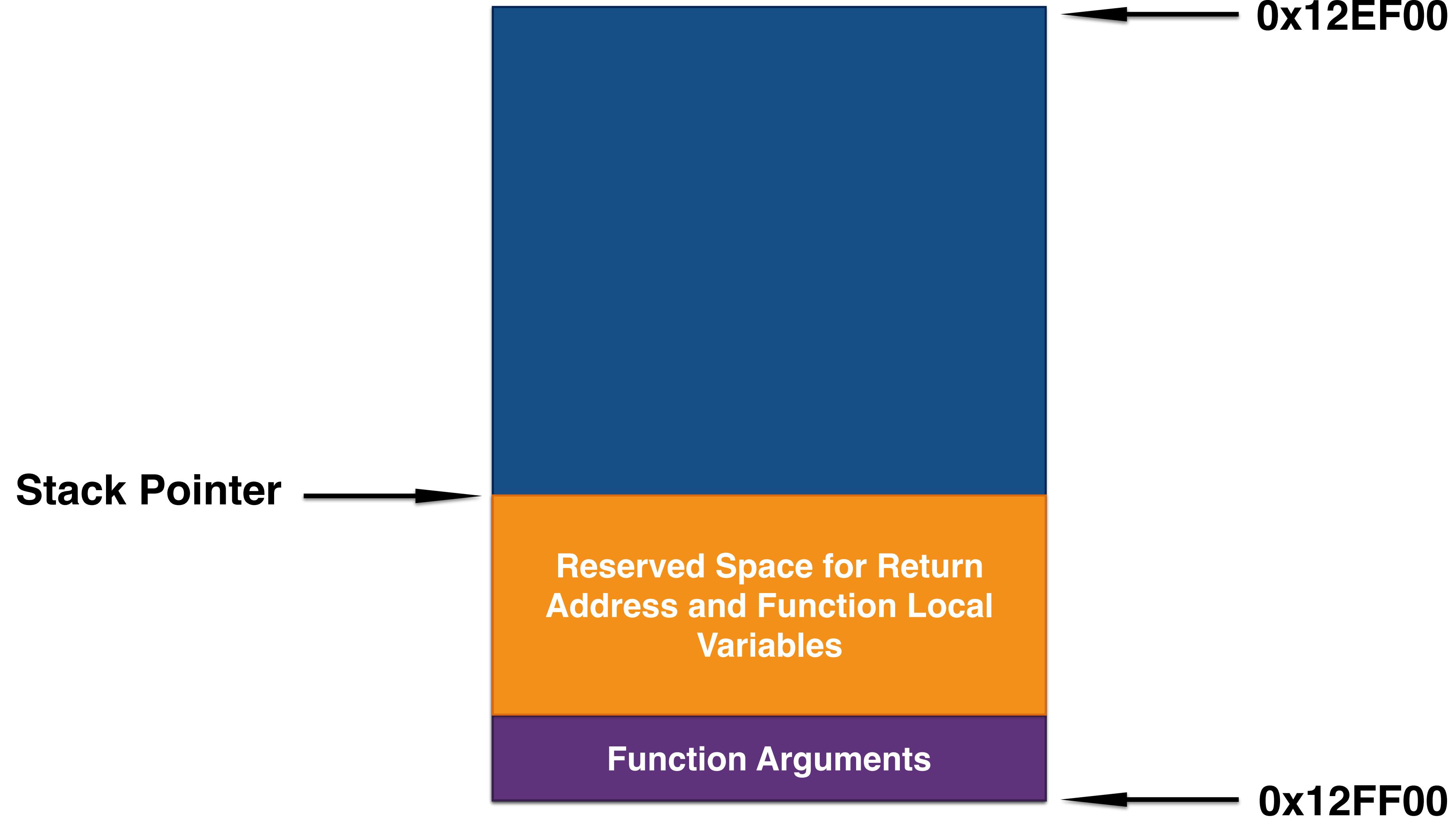
Unbalanced Stack



Info

Cuckoo: Monitor

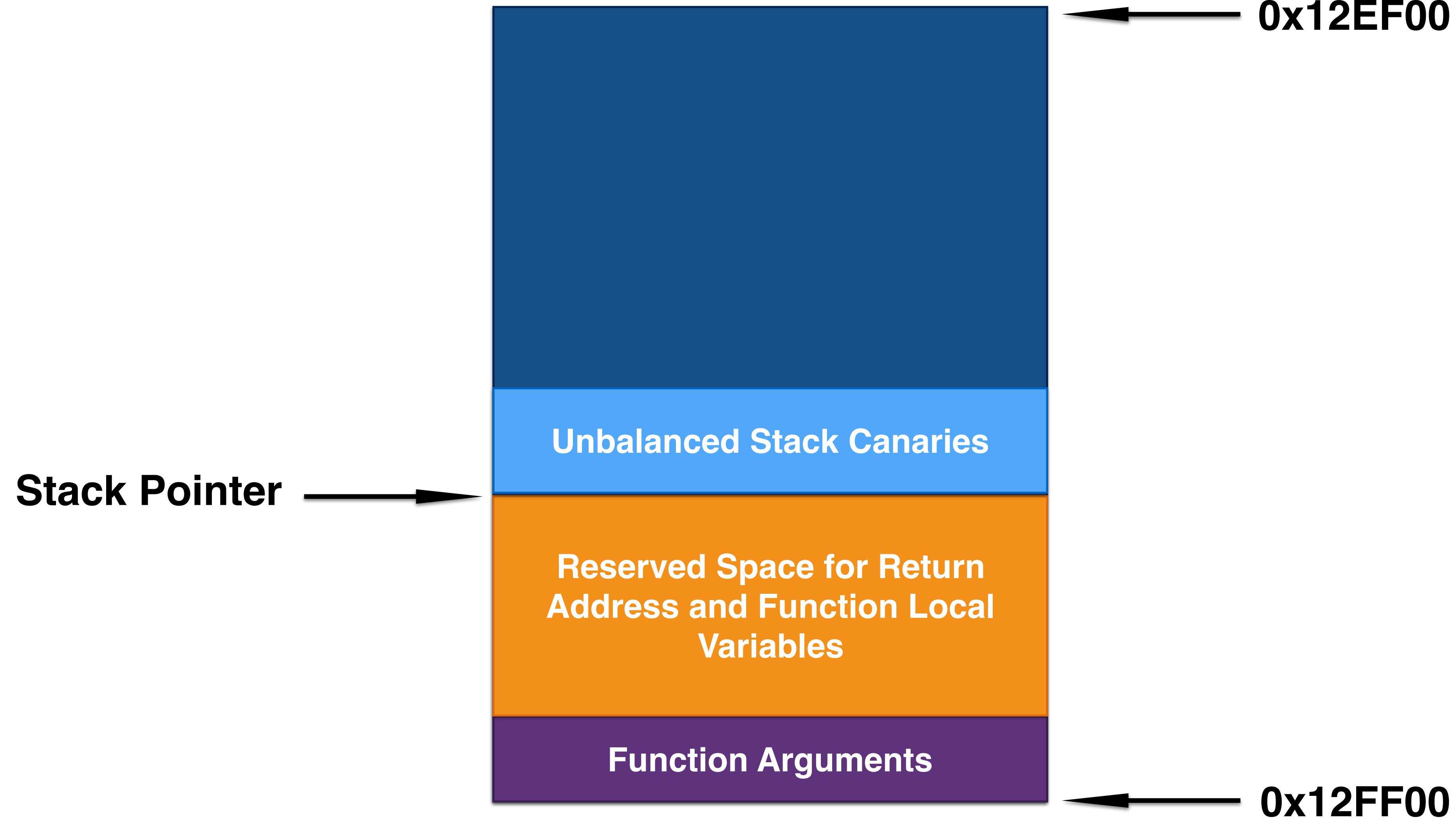
Unbalanced Stack



Info

Cuckoo: Monitor

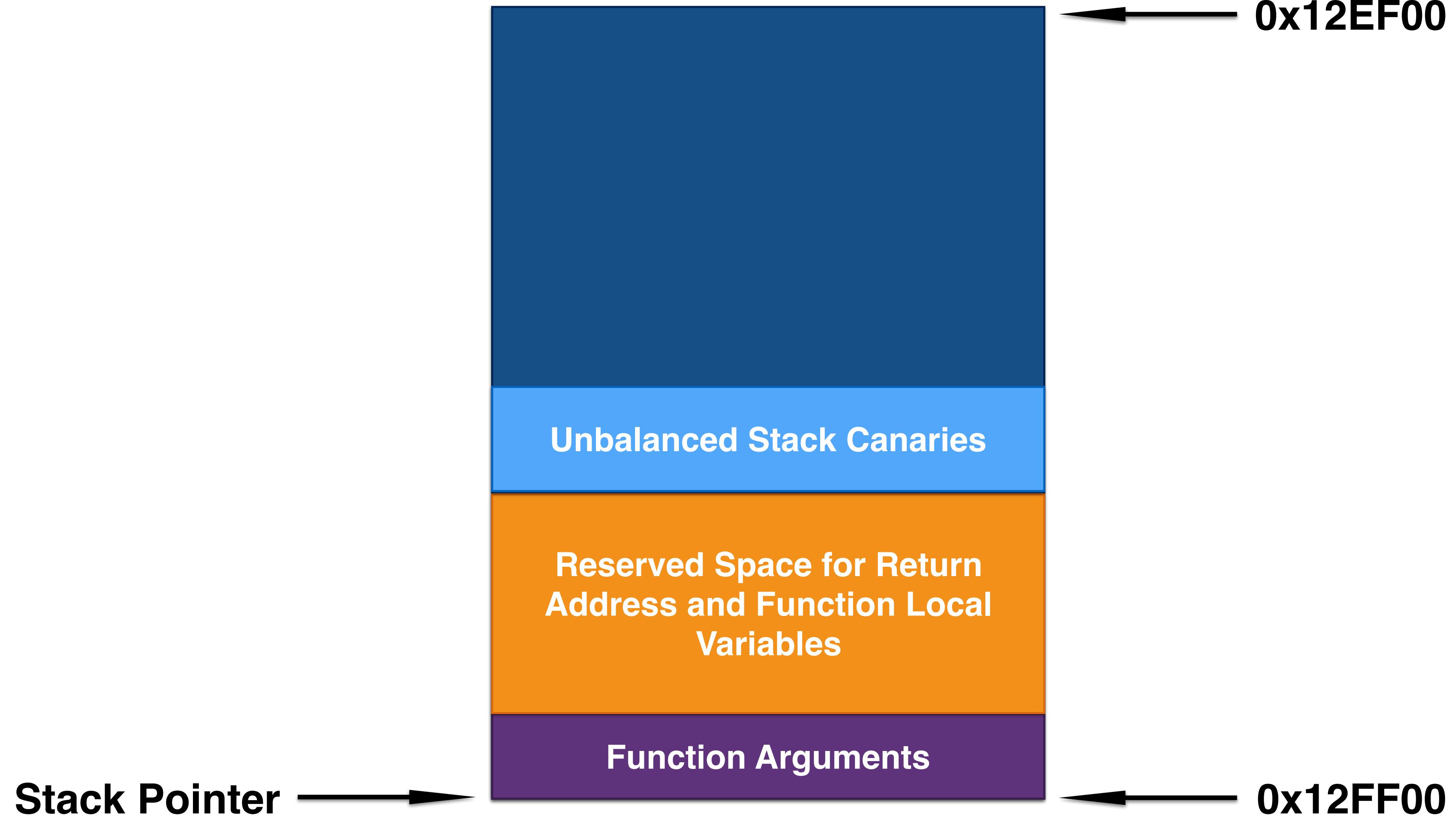
Unbalanced Stack



Info

Cuckoo: Monitor

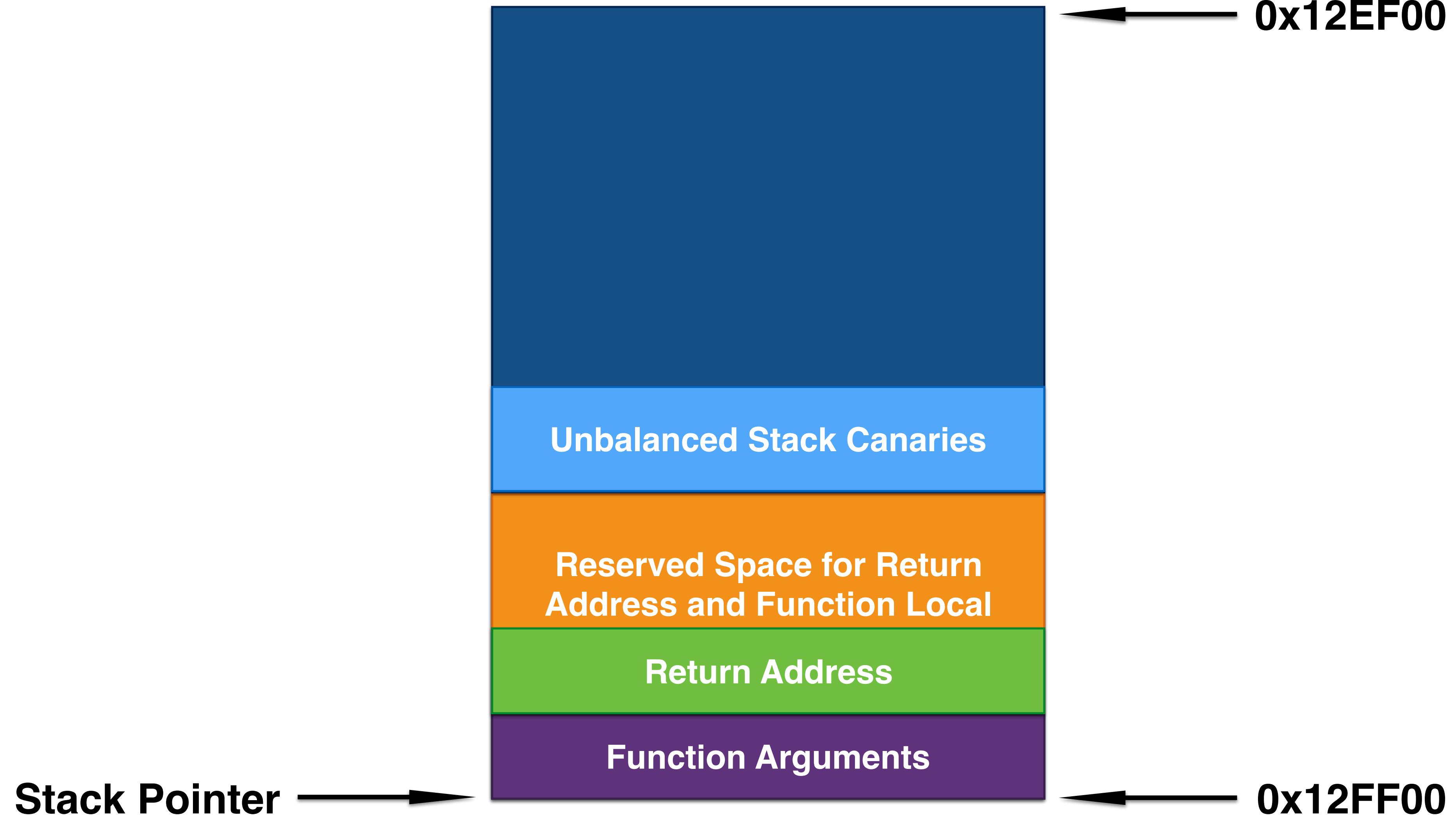
Unbalanced Stack



Info

Cuckoo: Monitor

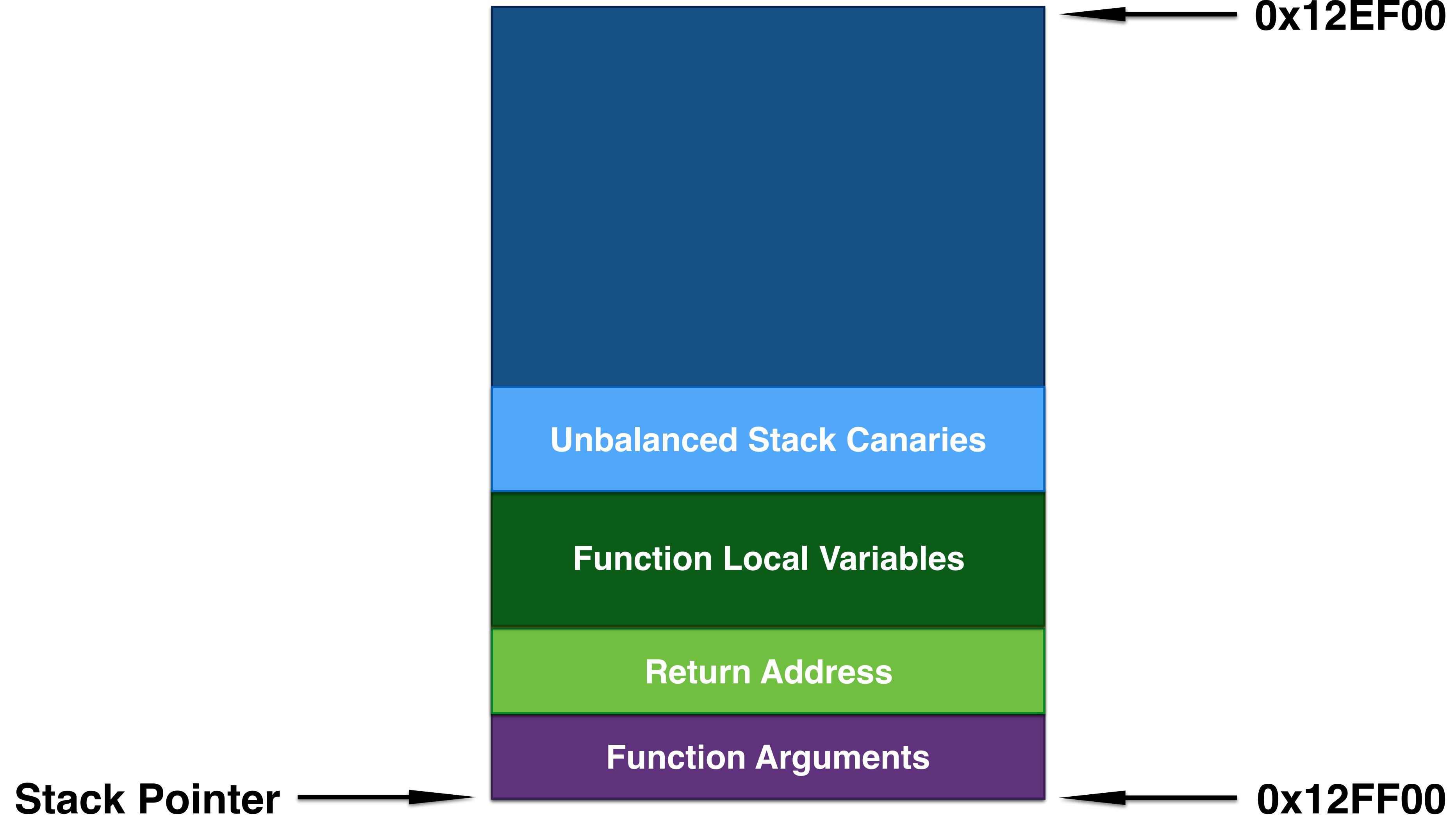
Unbalanced Stack



Info

Cuckoo: Monitor

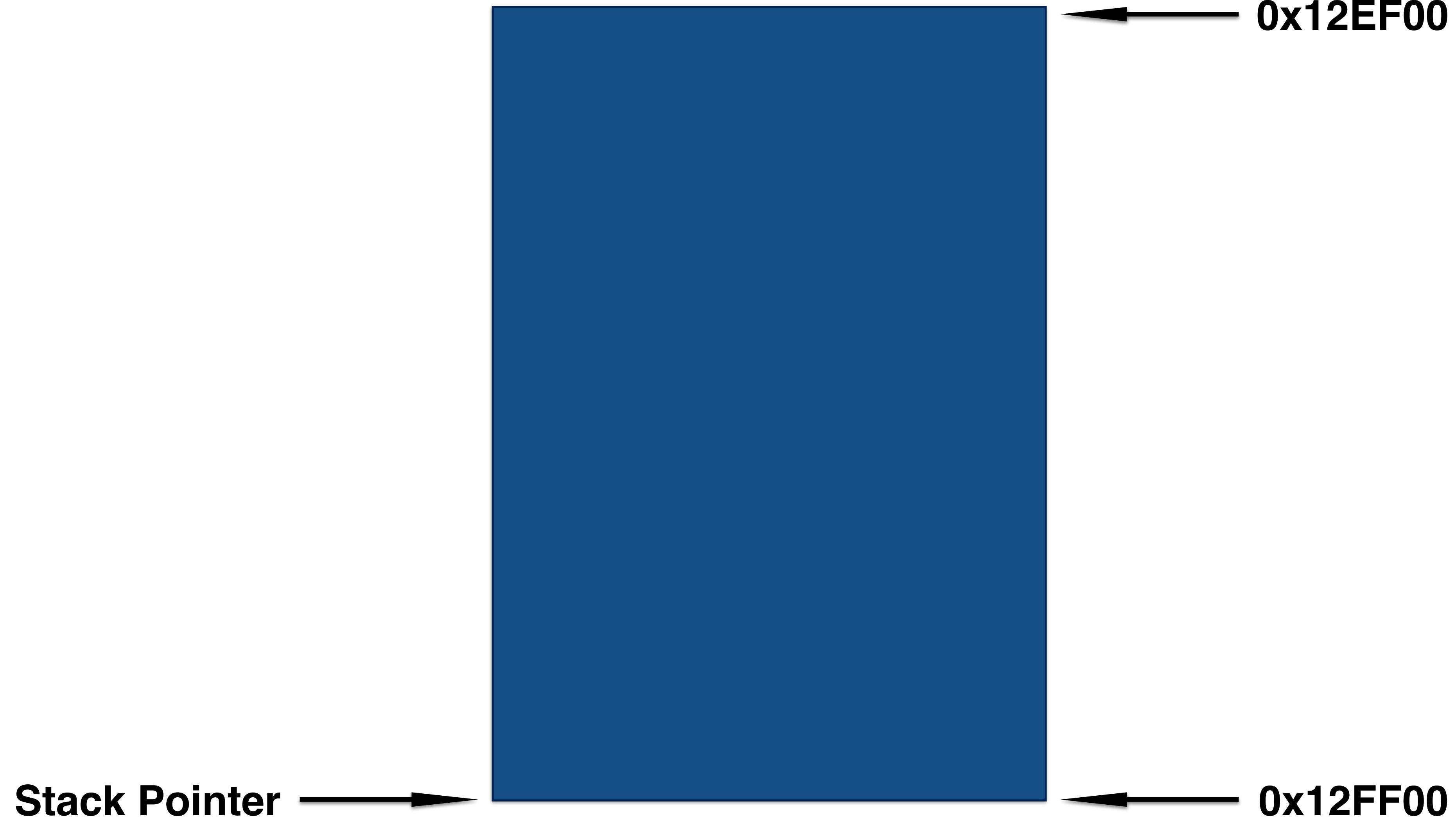
Unbalanced Stack



Detection

Cuckoo: Monitor

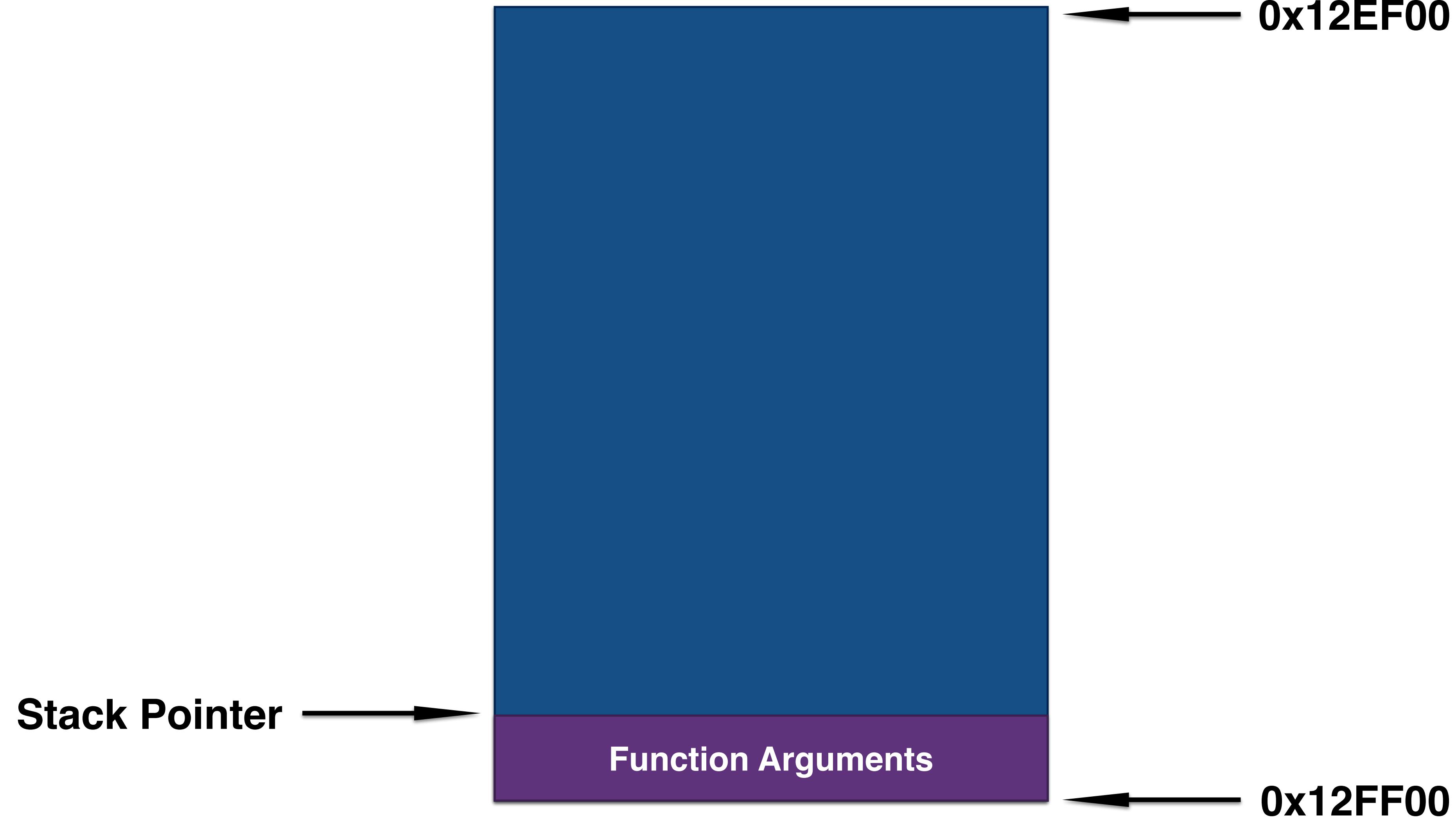
Unbalanced Stack



Detection

Cuckoo: Monitor

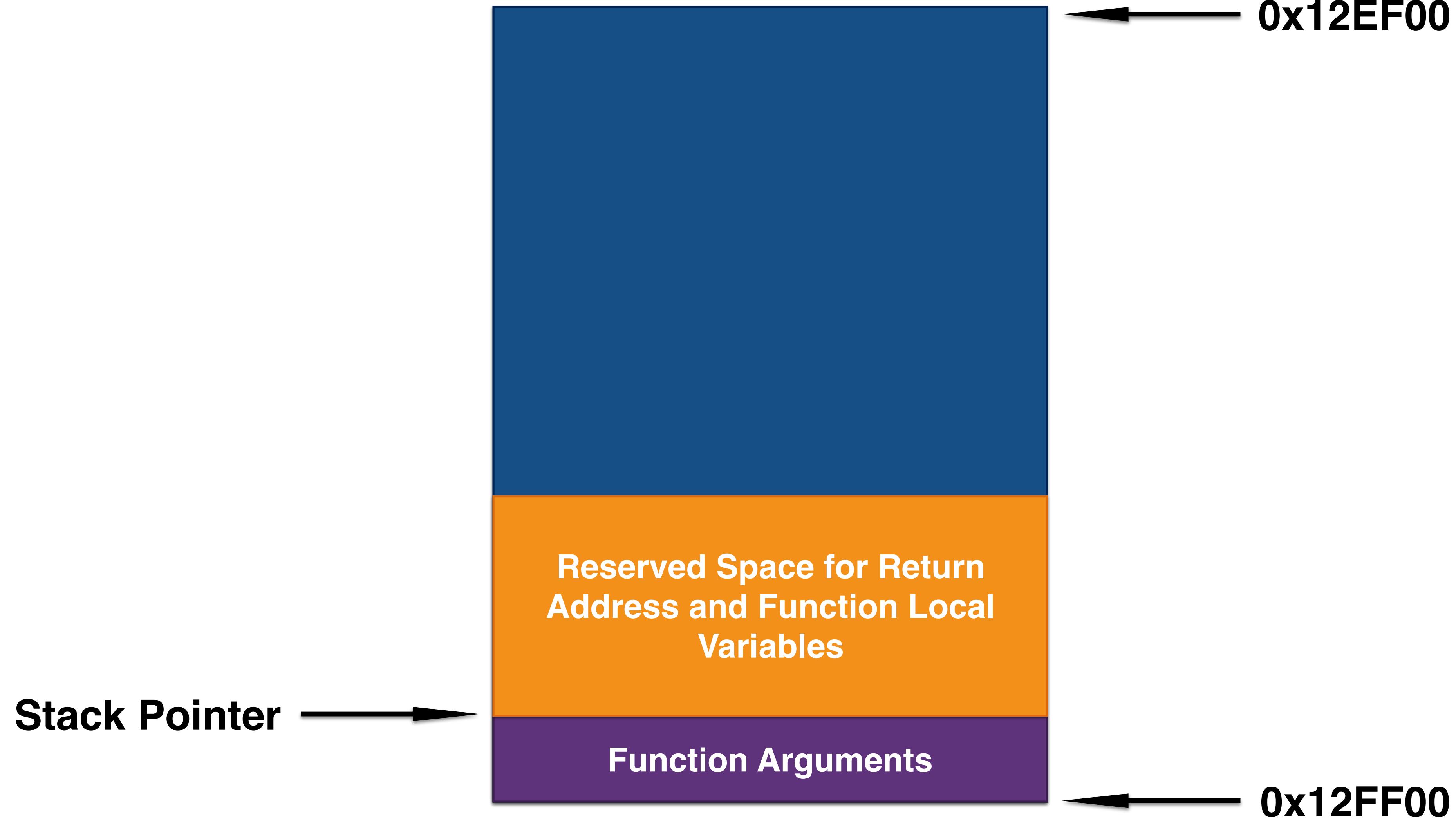
Unbalanced Stack



Detection

Cuckoo: Monitor

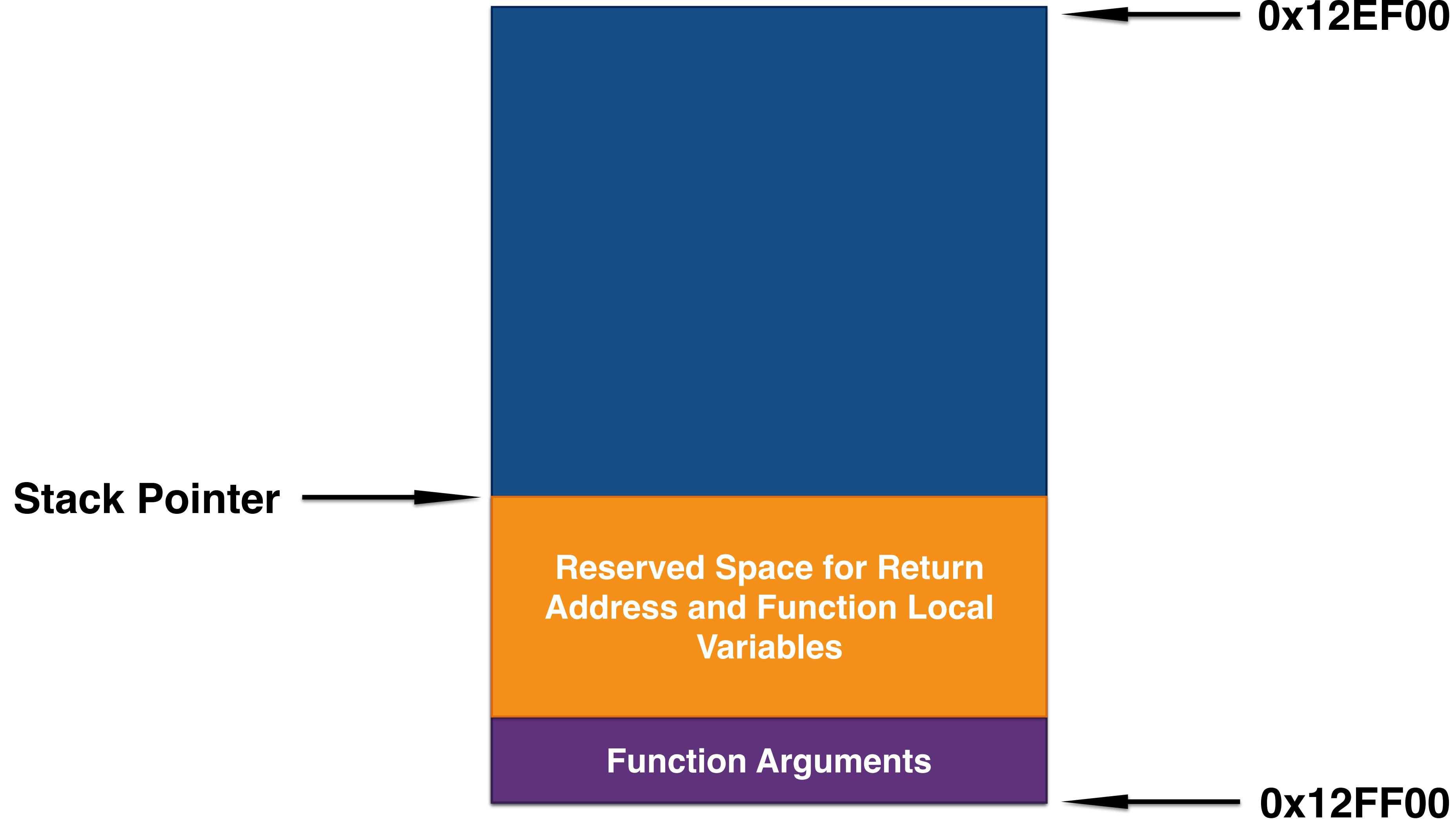
Unbalanced Stack



Detection

Cuckoo: Monitor

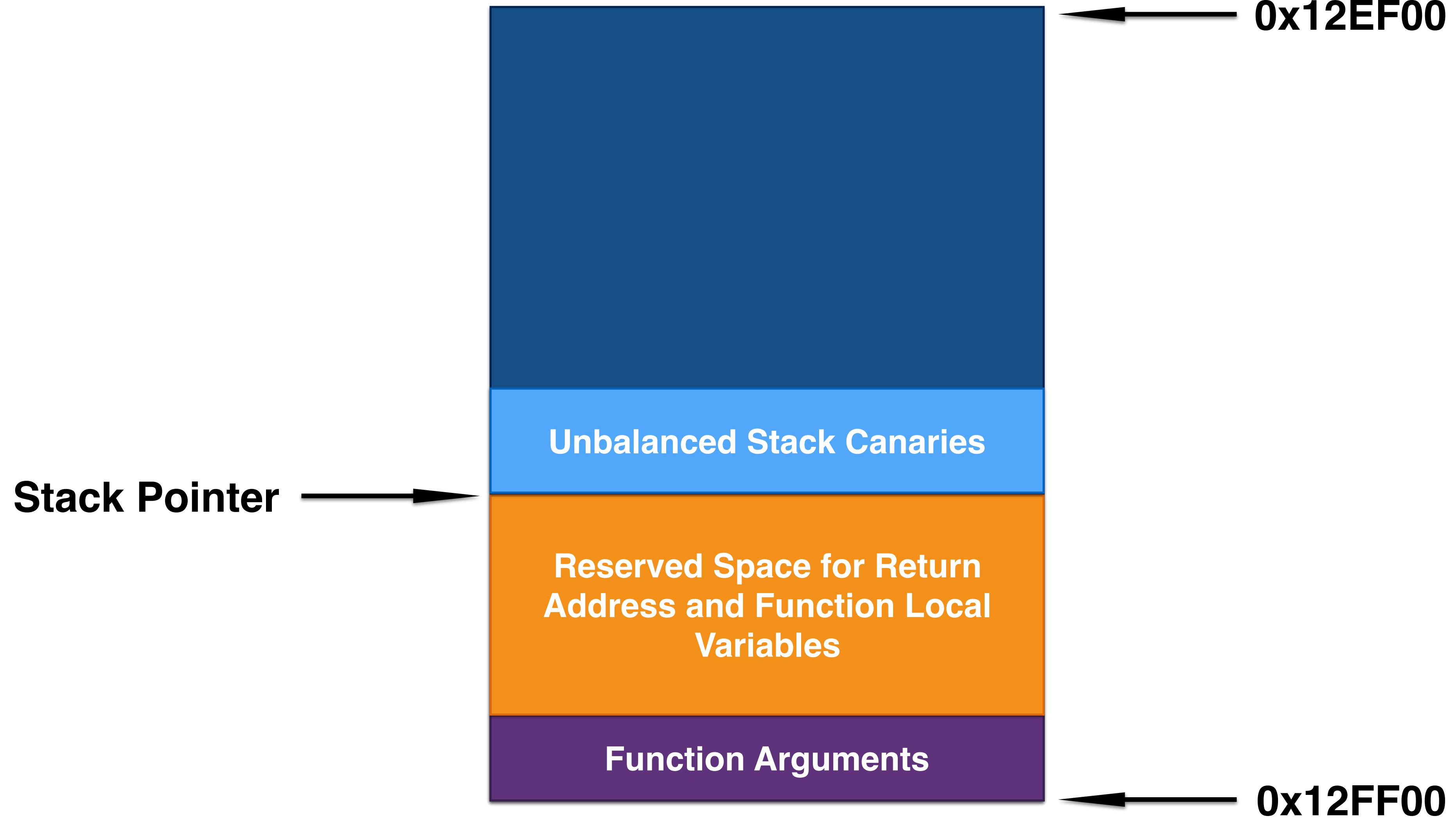
Unbalanced Stack



Detection

Cuckoo: Monitor

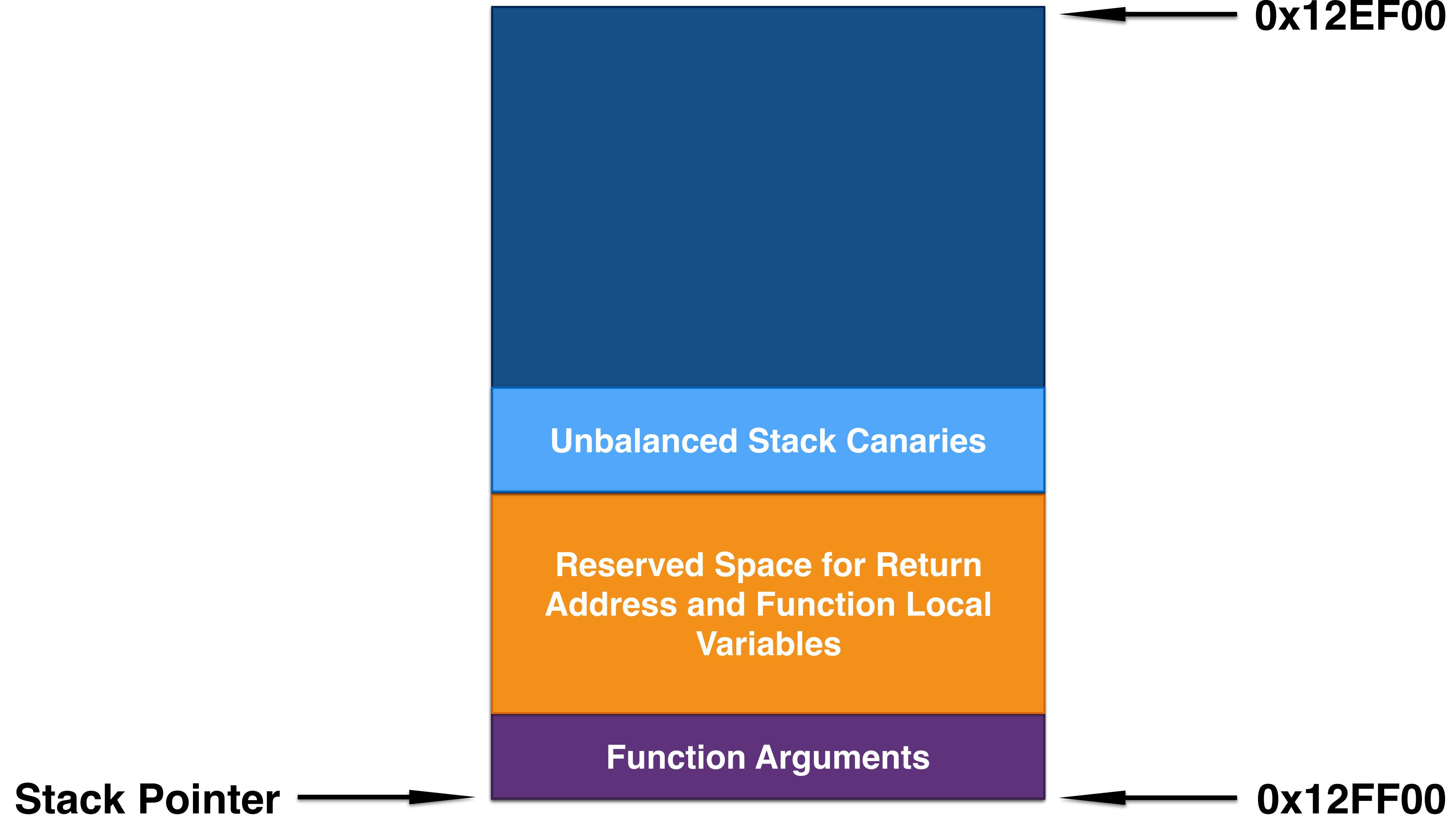
Unbalanced Stack



Detection

Cuckoo: Monitor

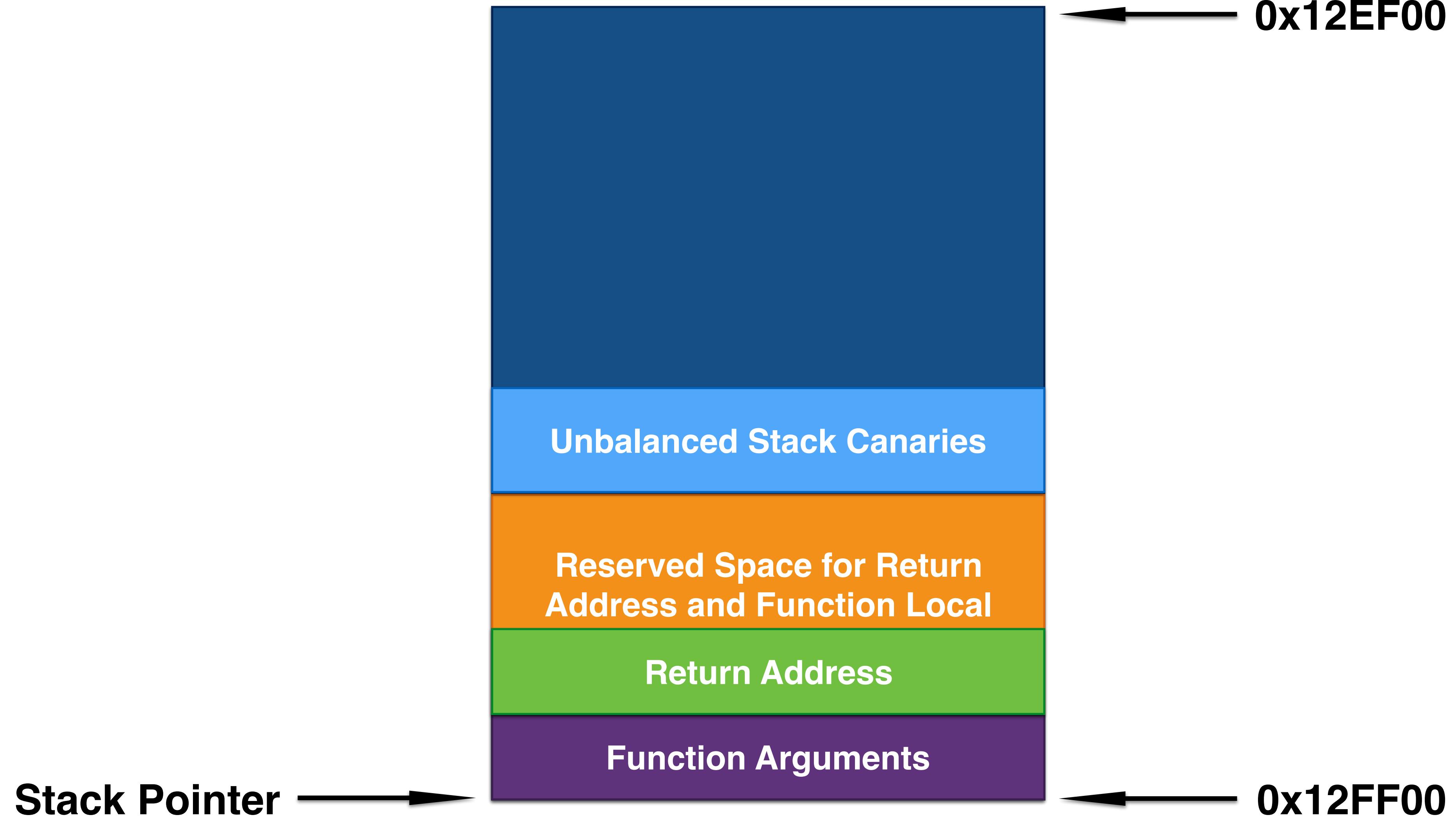
Unbalanced Stack



Detection

Cuckoo: Monitor

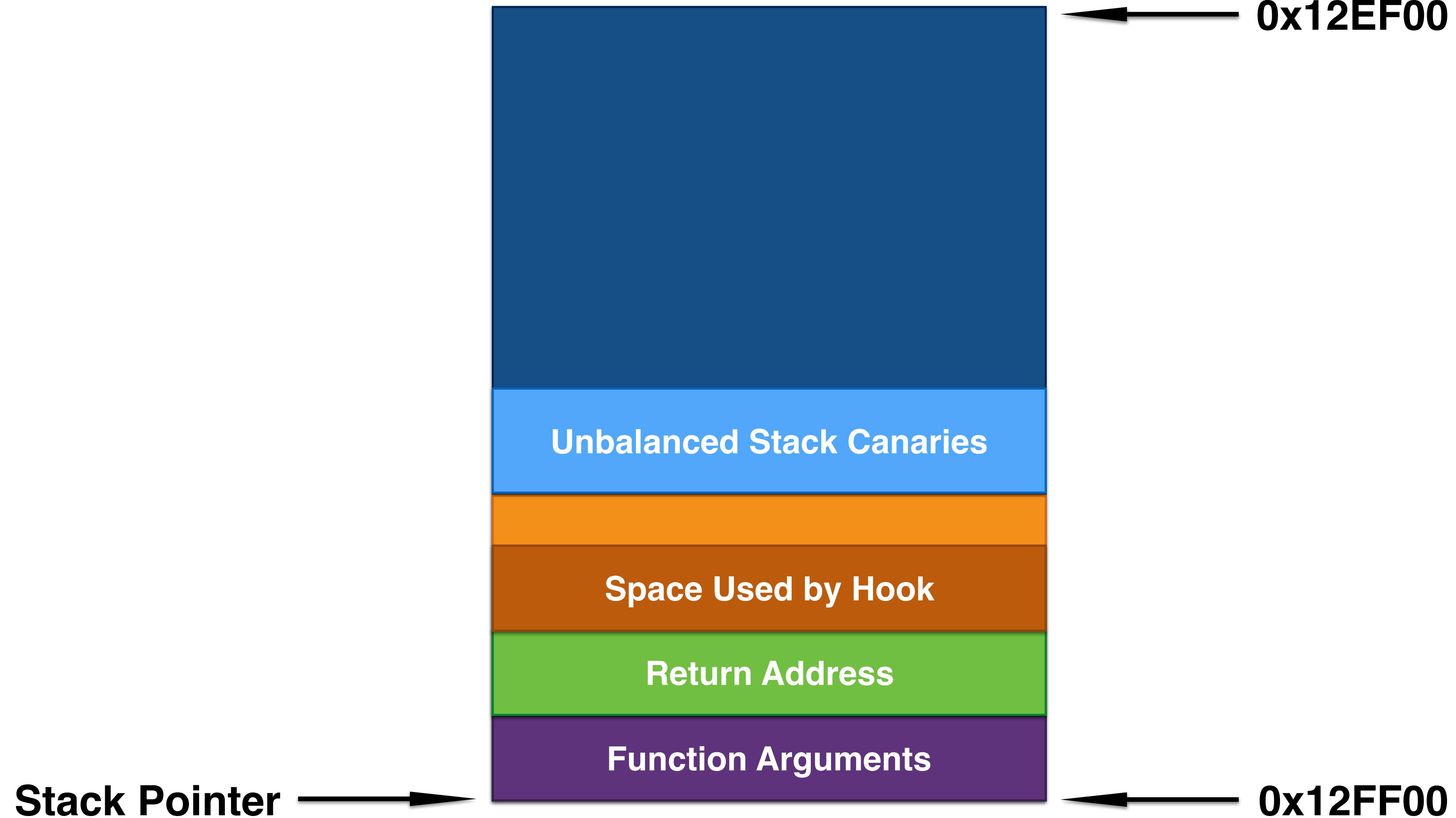
Unbalanced Stack



Detection

Cuckoo: Monitor

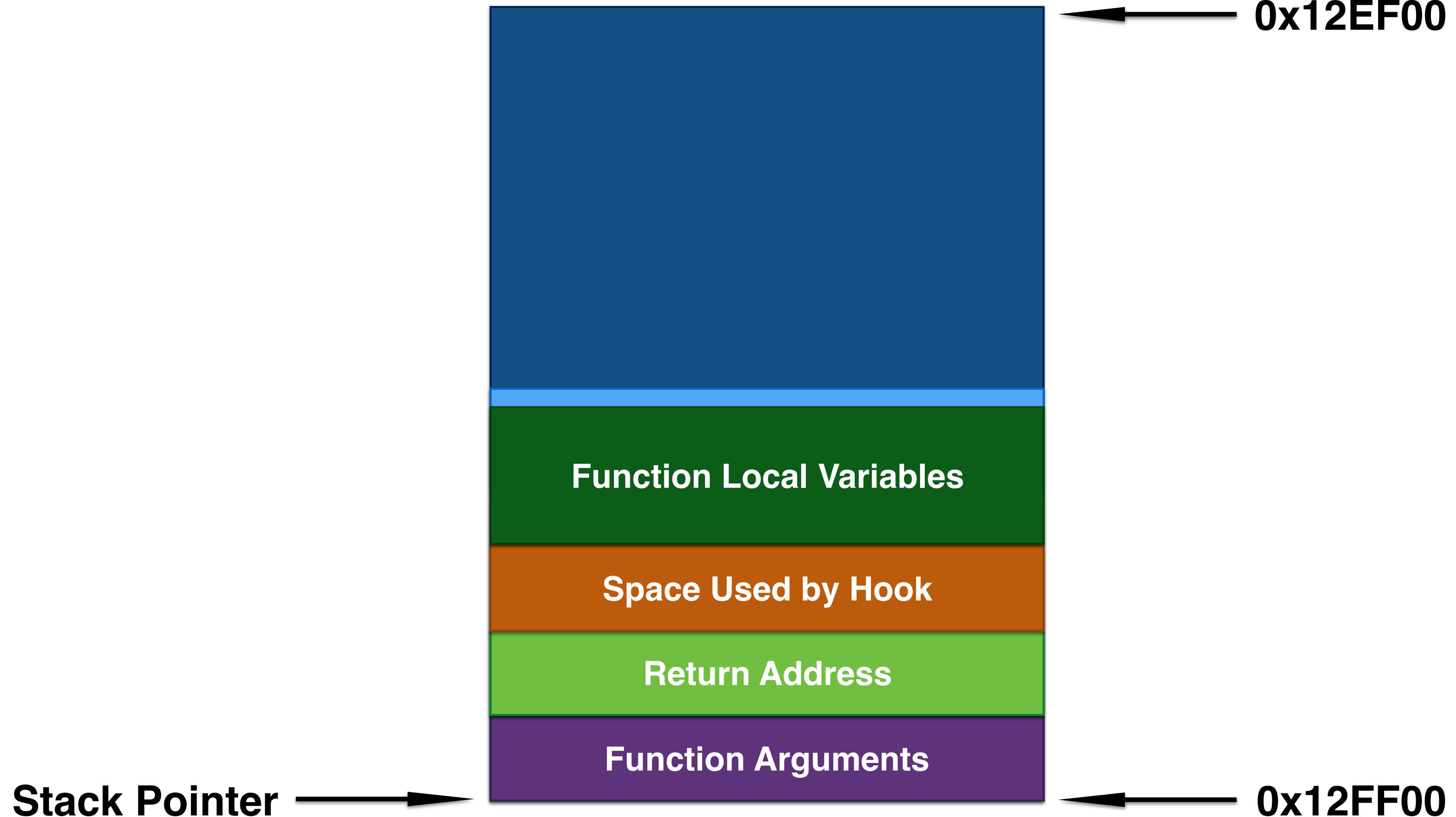
Unbalanced Stack



Detection

Cuckoo: Monitor

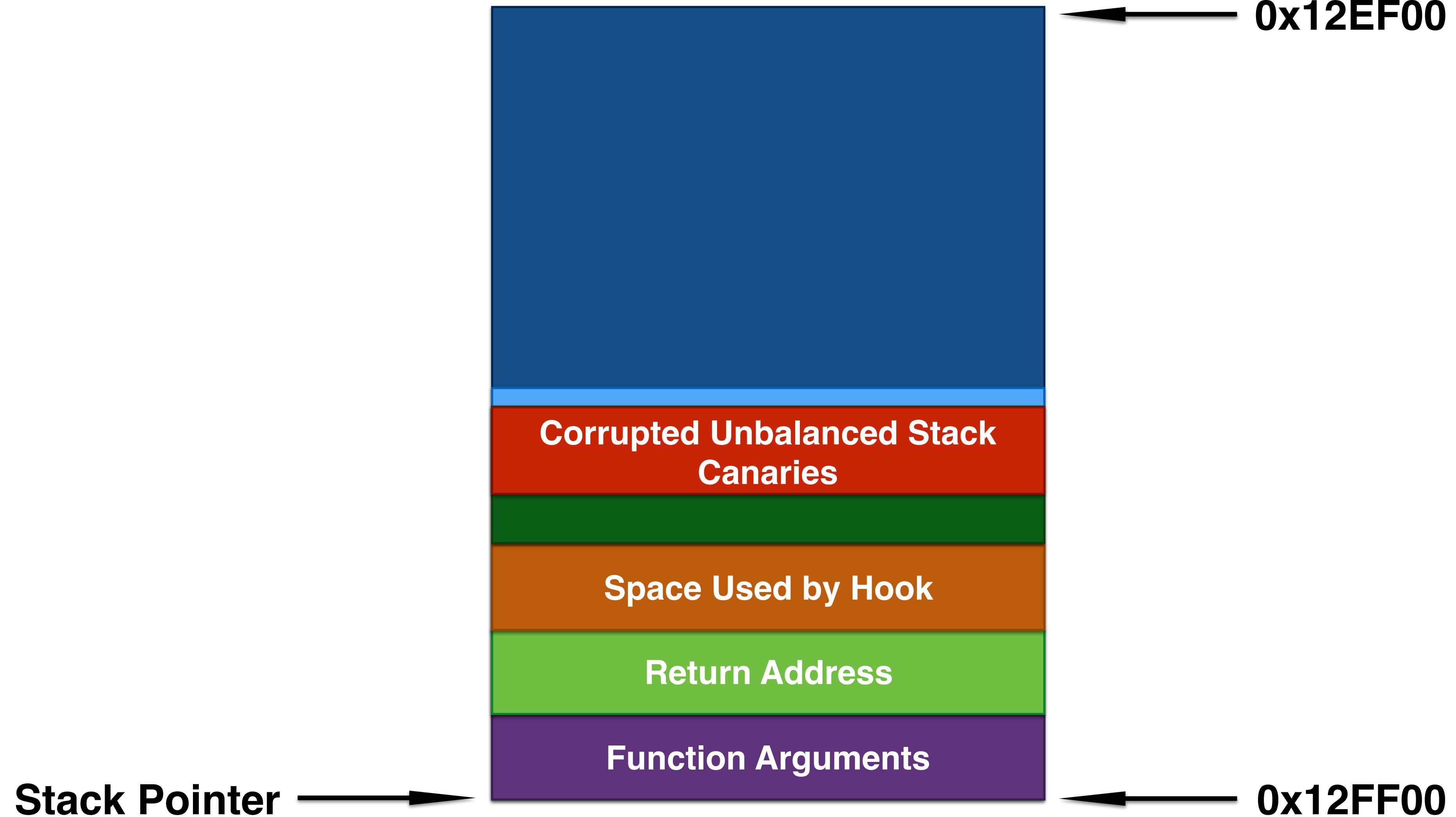
Unbalanced Stack



Detection

Cuckoo: Monitor

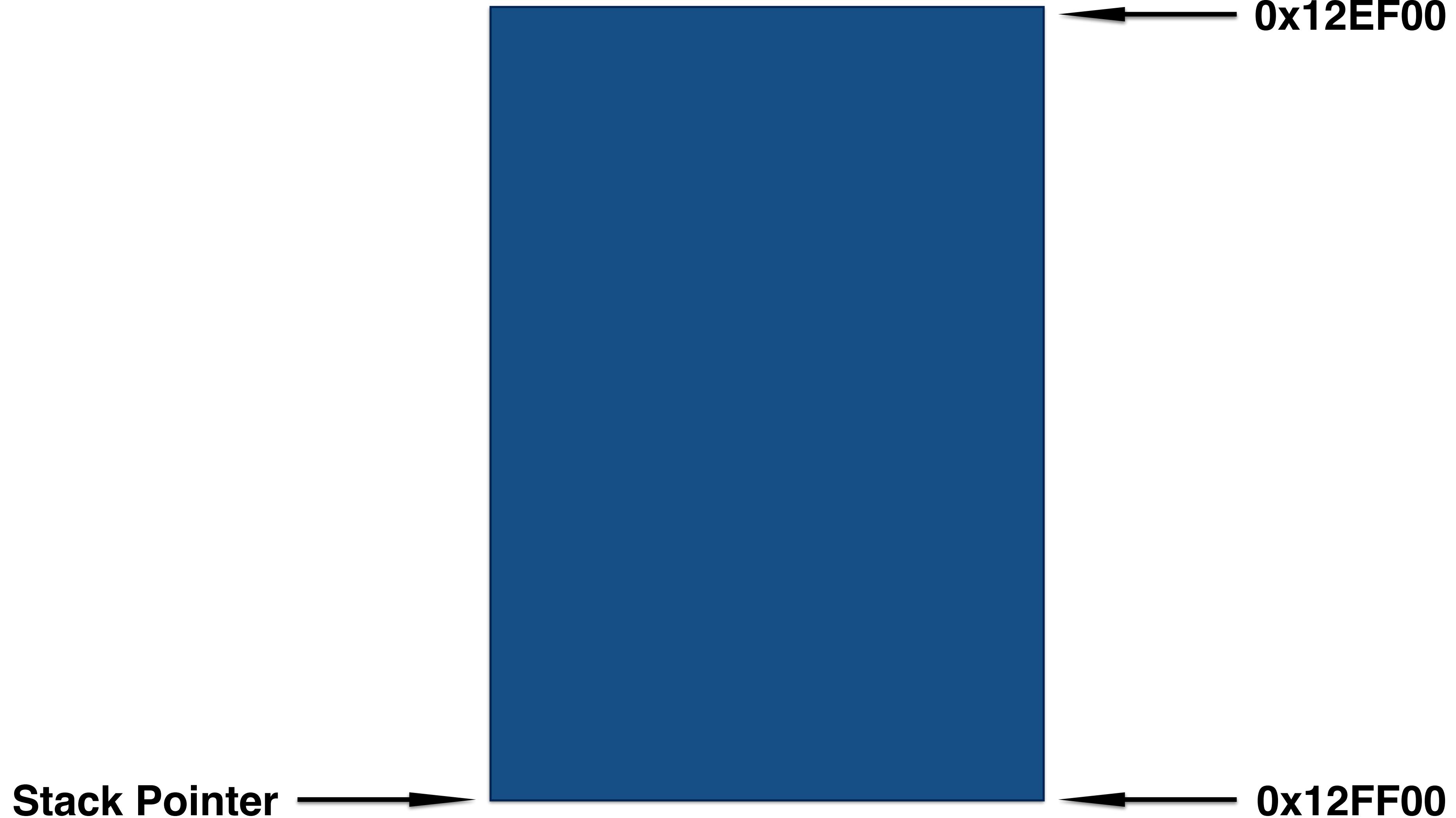
Unbalanced Stack



Fix

Cuckoo: Monitor

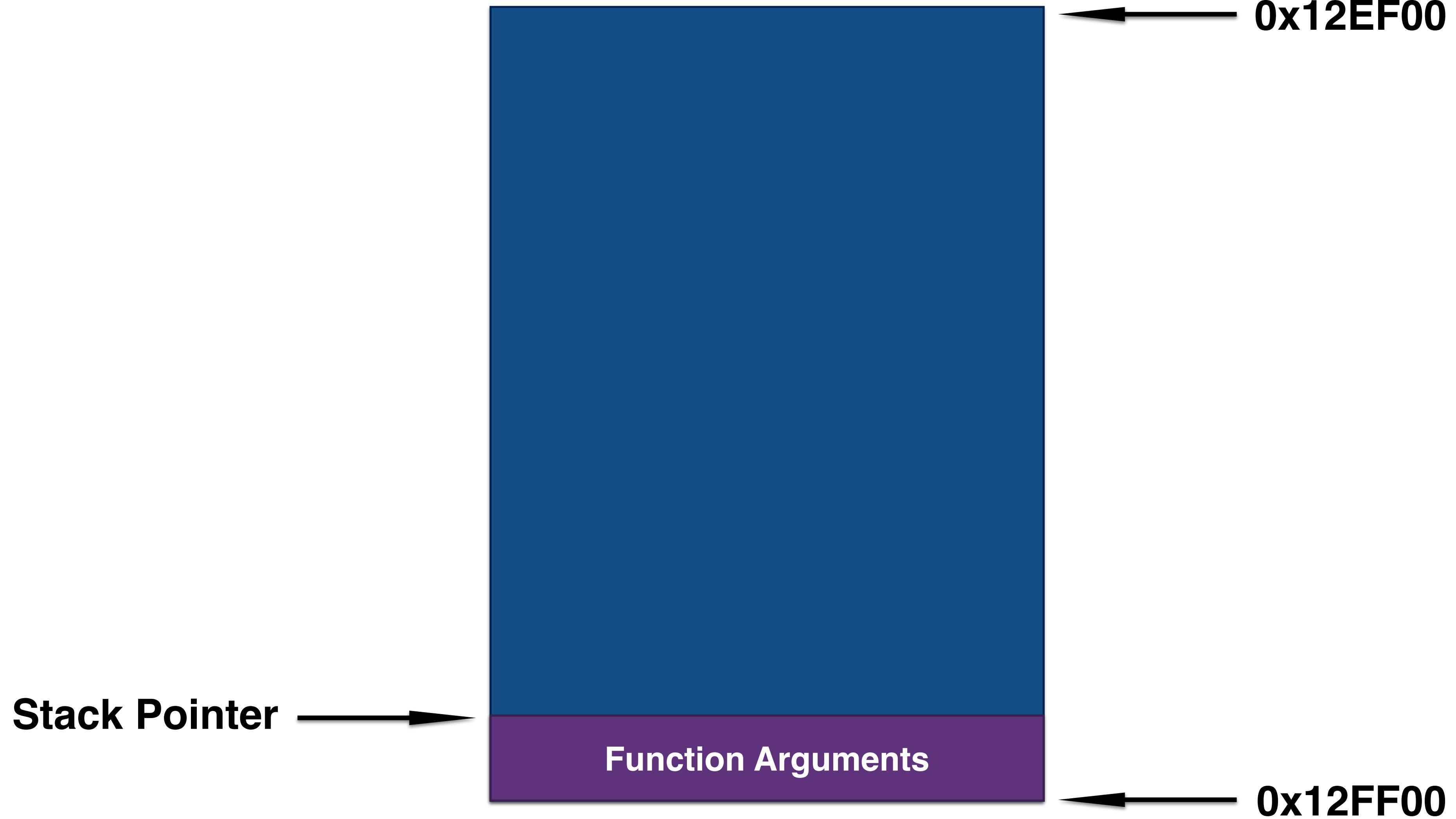
Unbalanced Stack



Fix

Cuckoo: Monitor

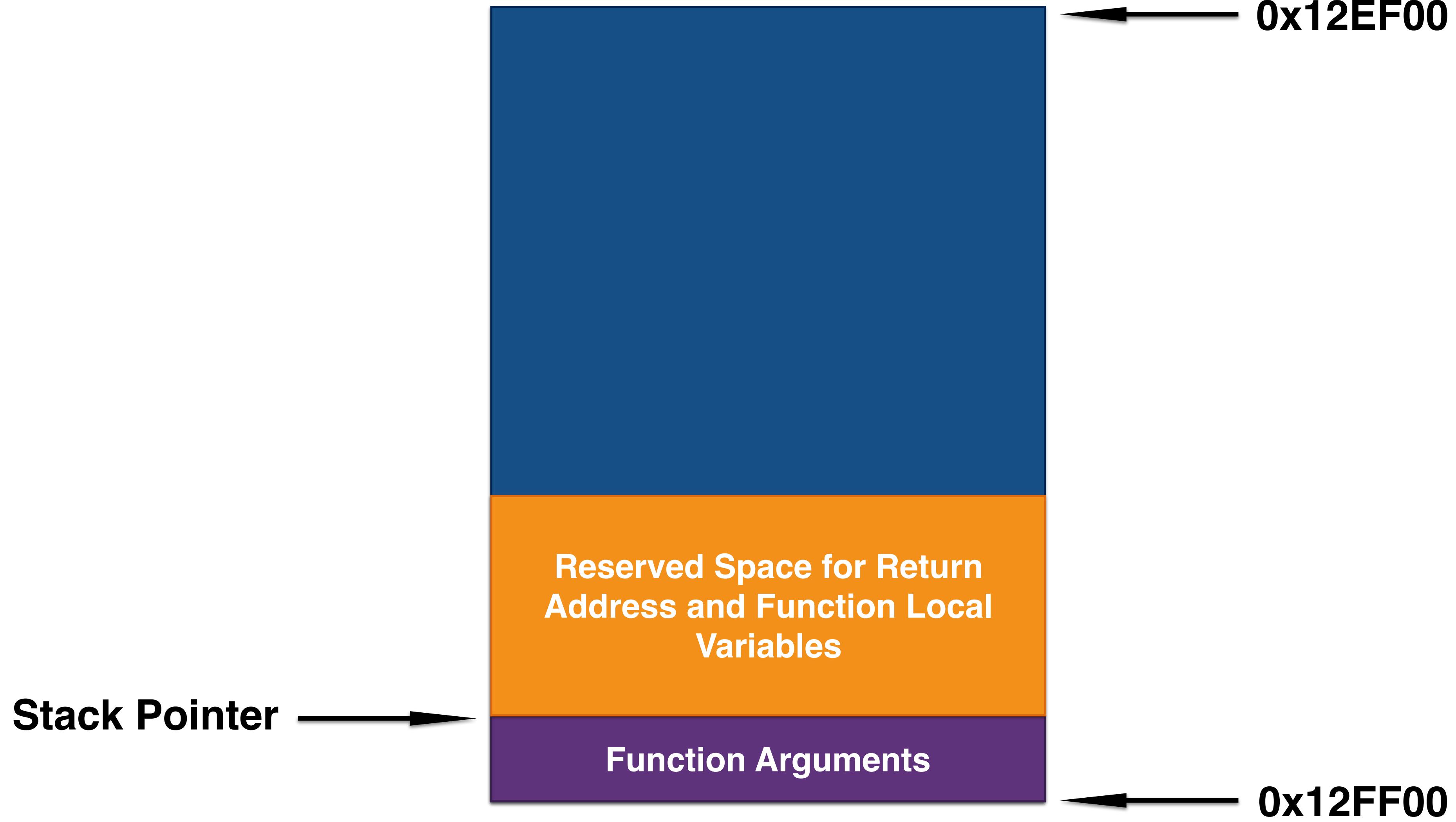
Unbalanced Stack



Fix

Cuckoo: Monitor

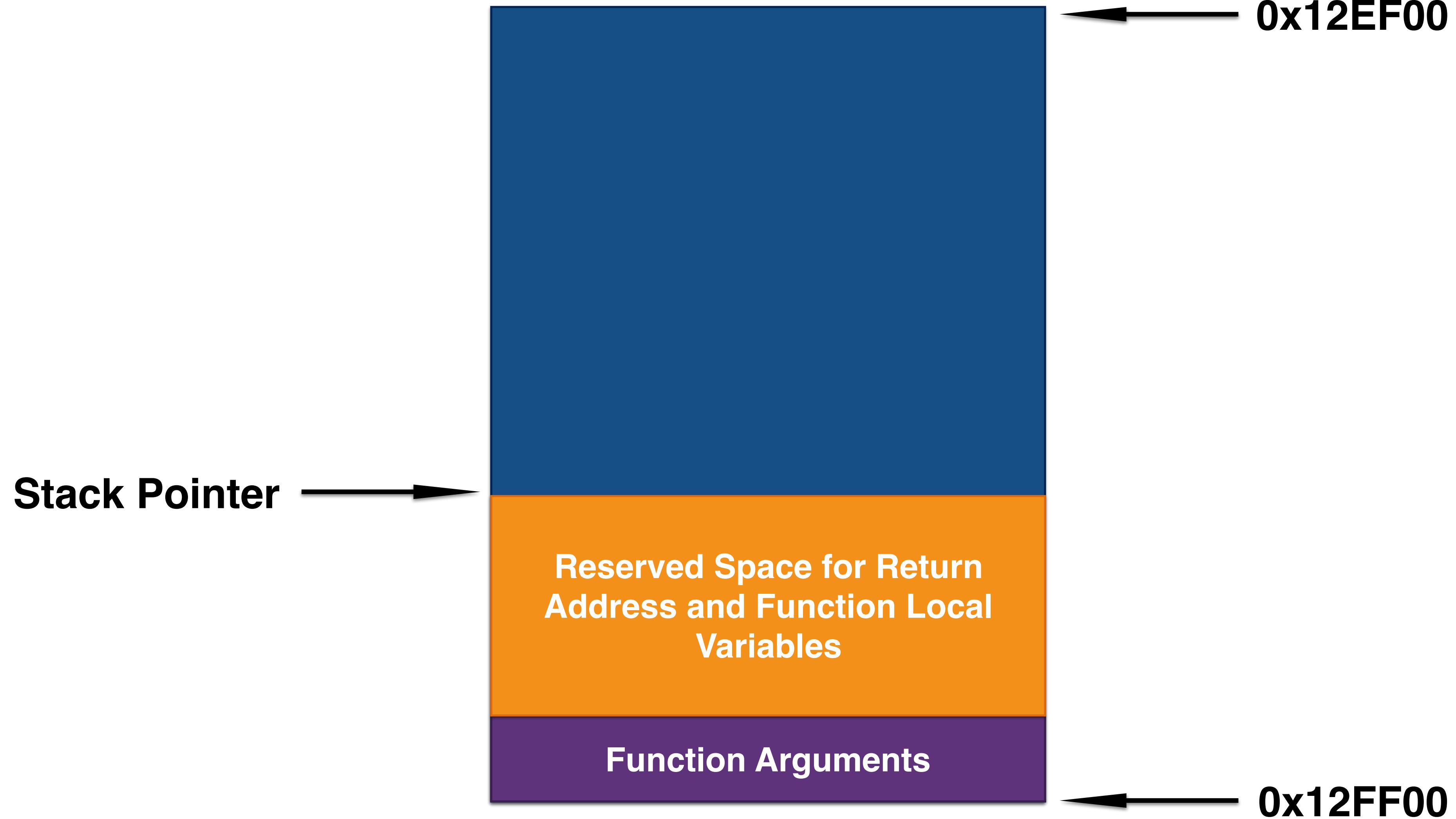
Unbalanced Stack



Fix

Cuckoo: Monitor

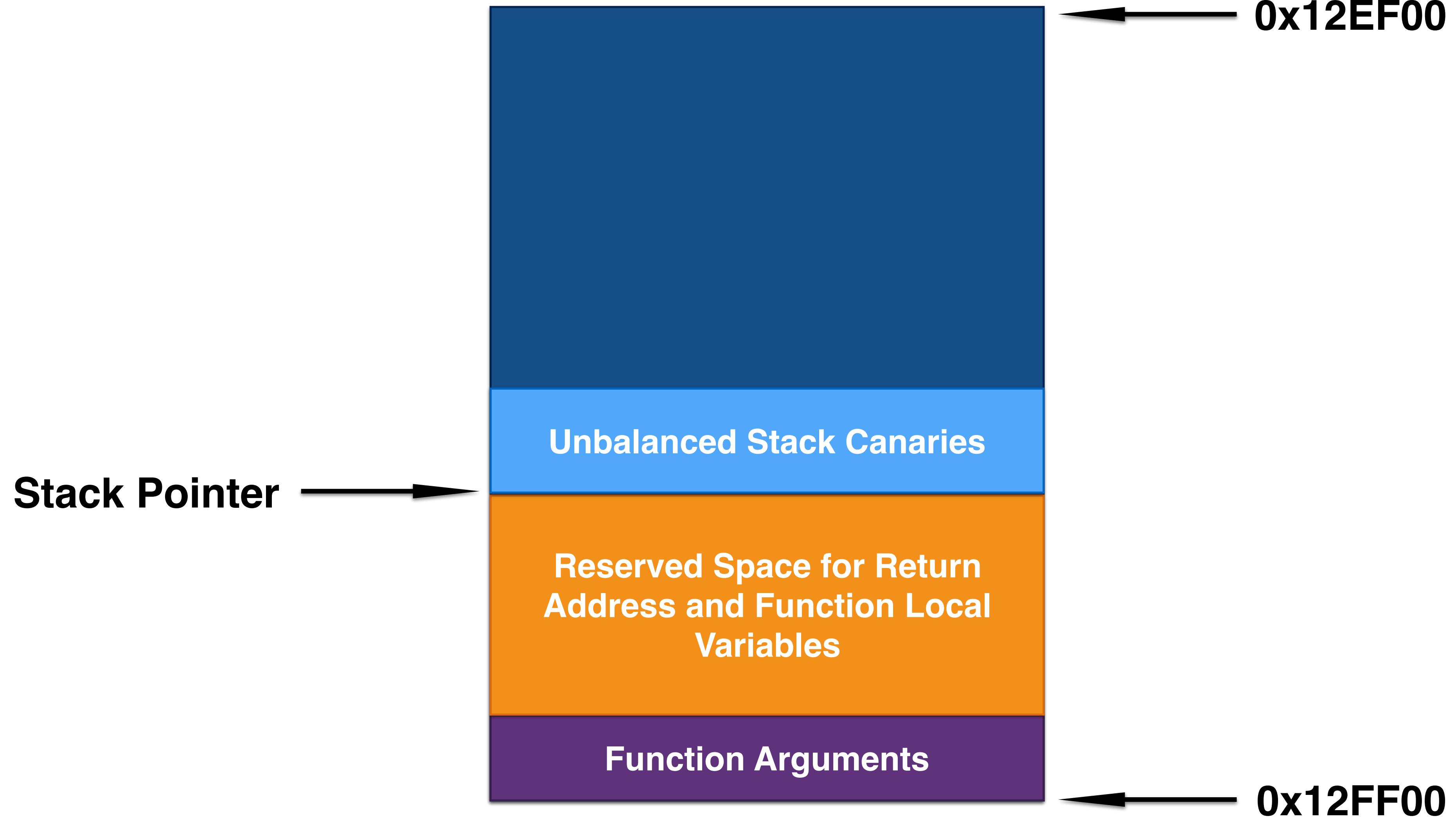
Unbalanced Stack



Fix

Cuckoo: Monitor

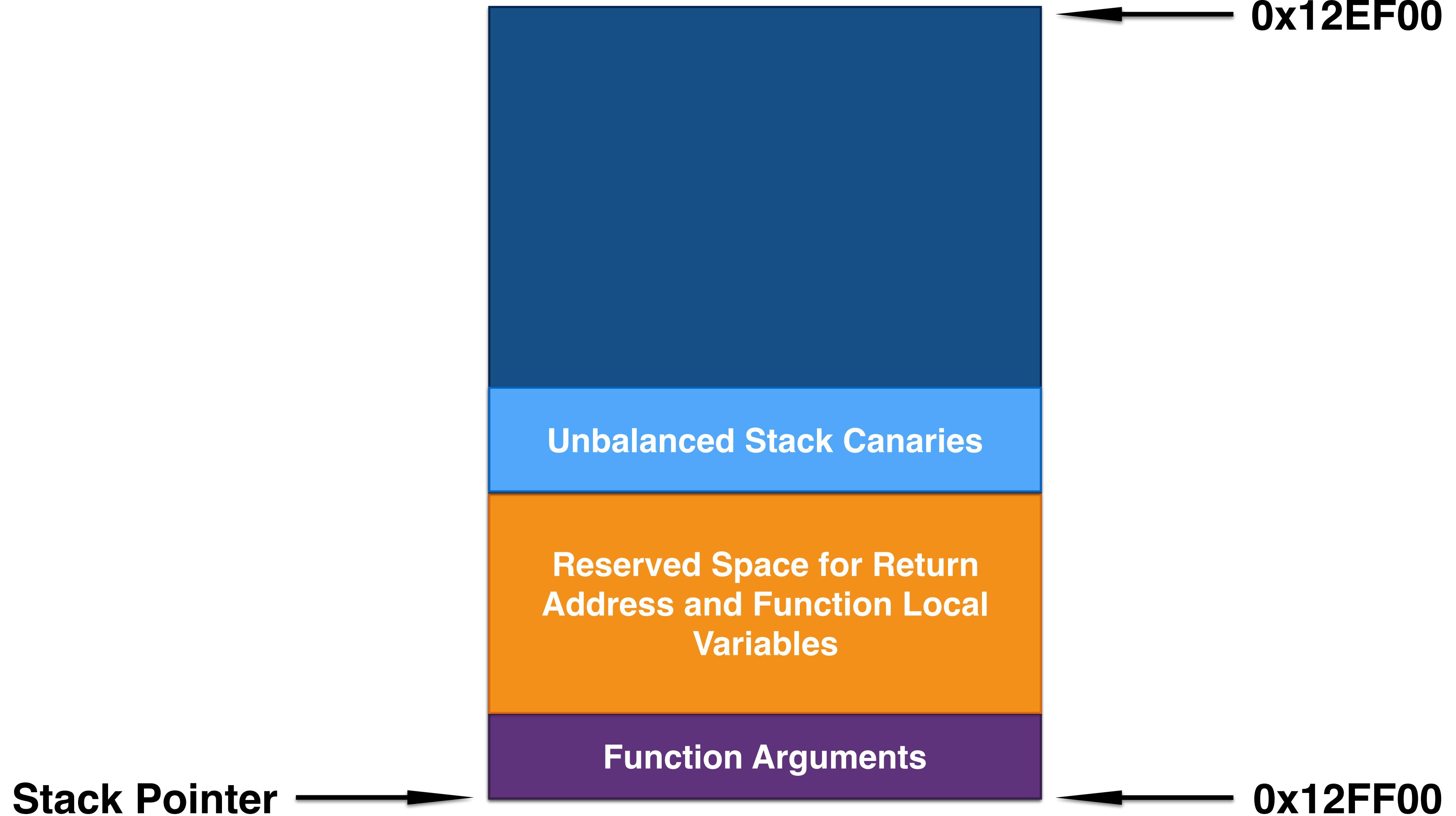
Unbalanced Stack



Fix

Cuckoo: Monitor

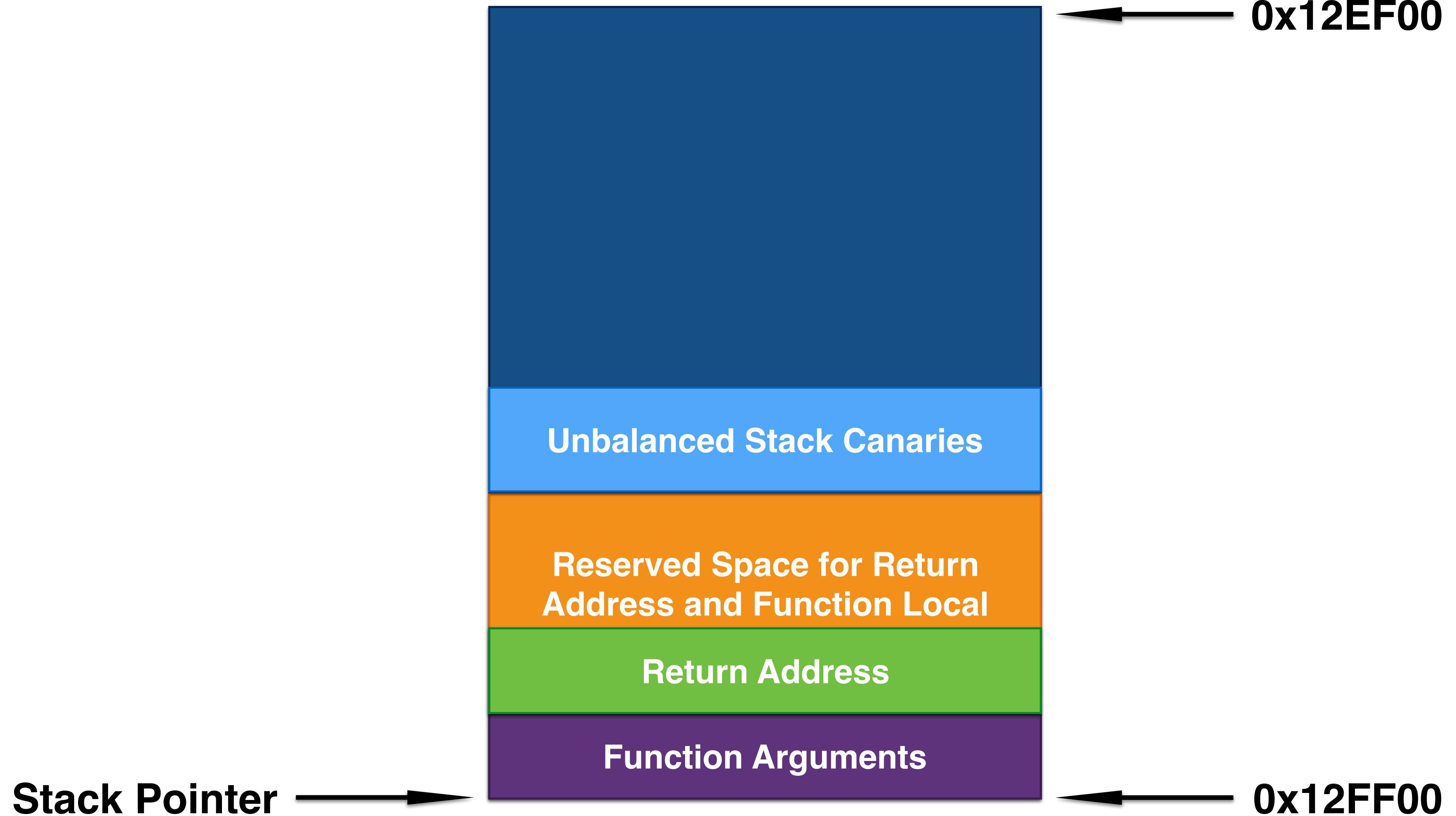
Unbalanced Stack



Fix

Cuckoo: Monitor

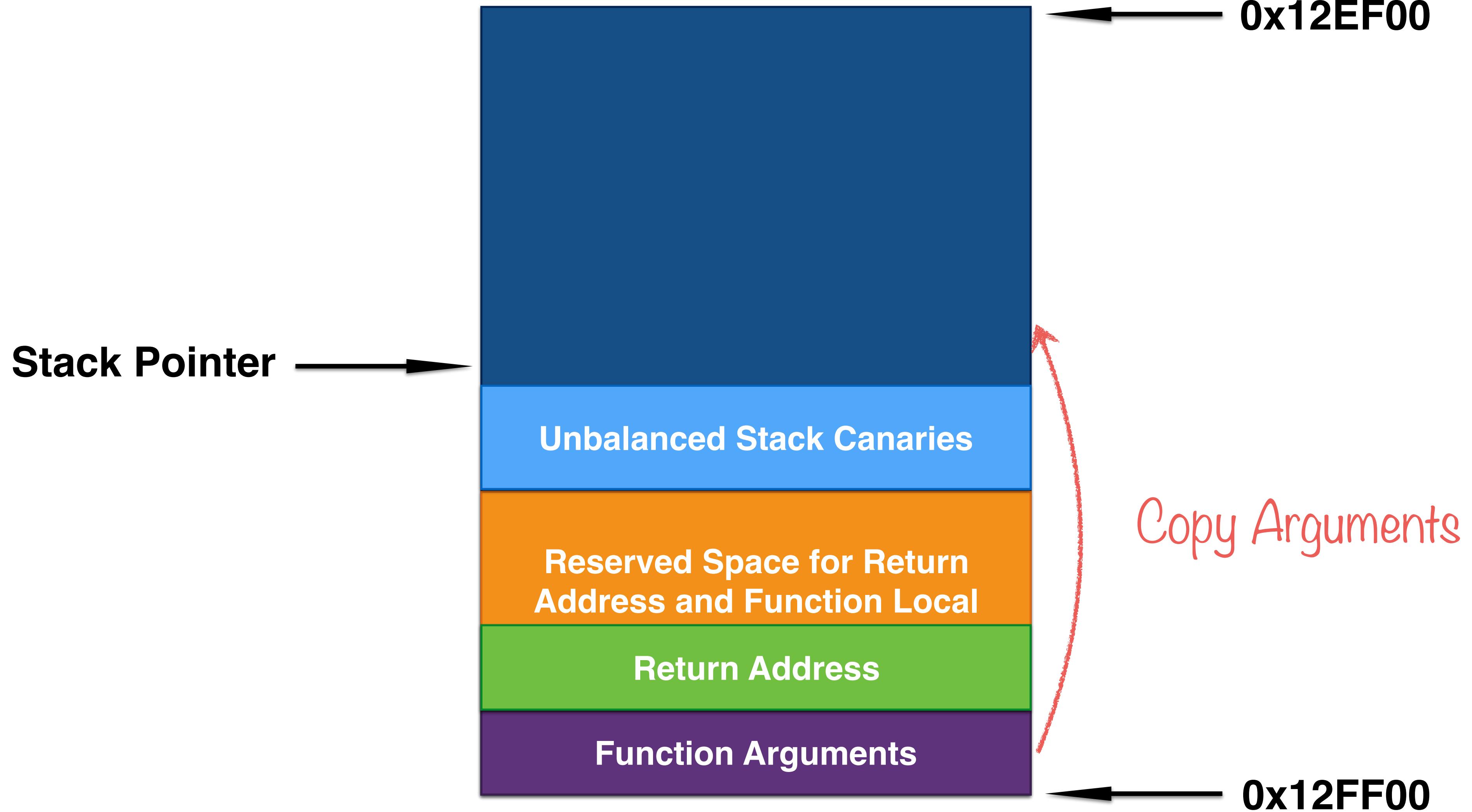
Unbalanced Stack



Fix

Cuckoo: Monitor

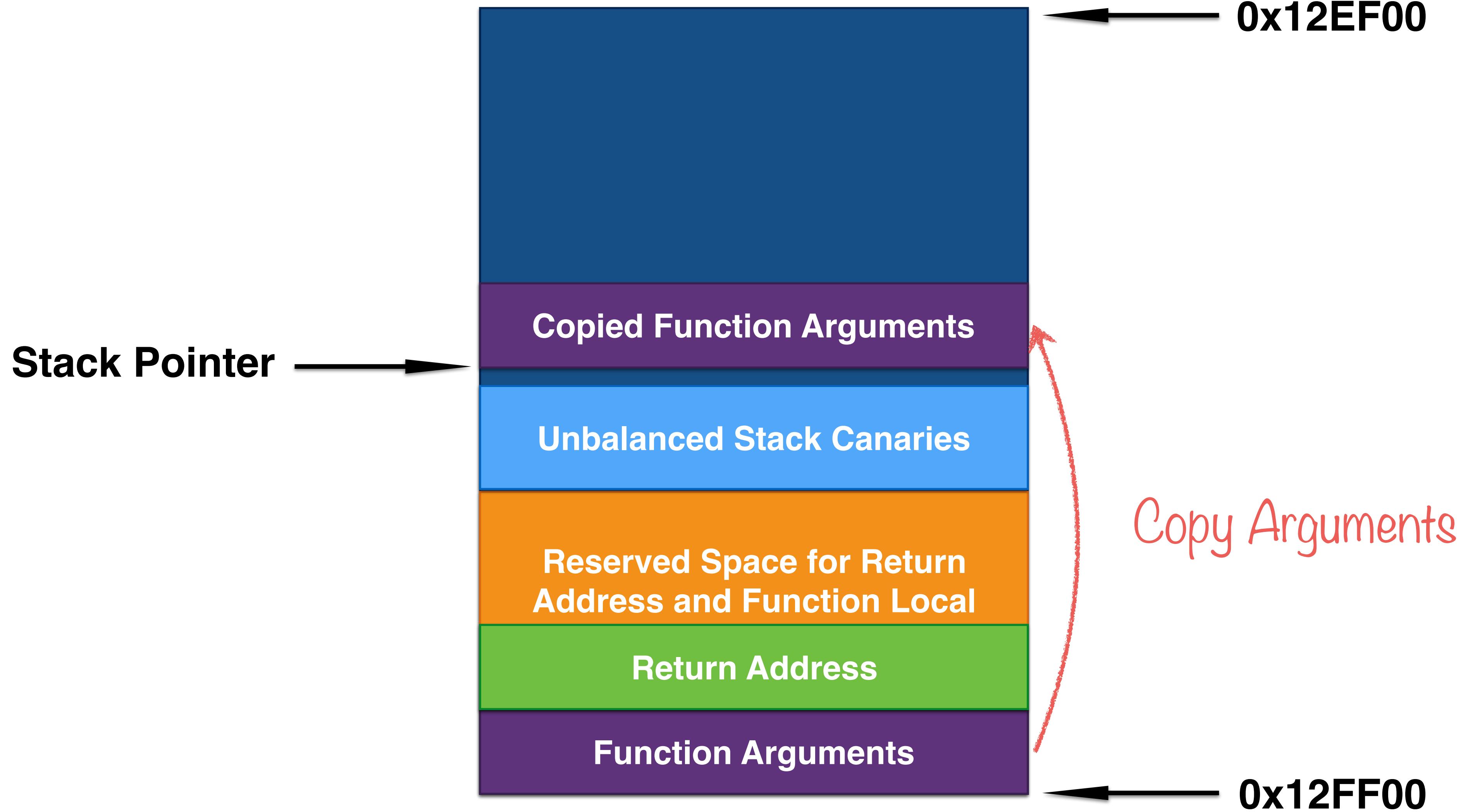
Unbalanced Stack



Fix

Cuckoo: Monitor

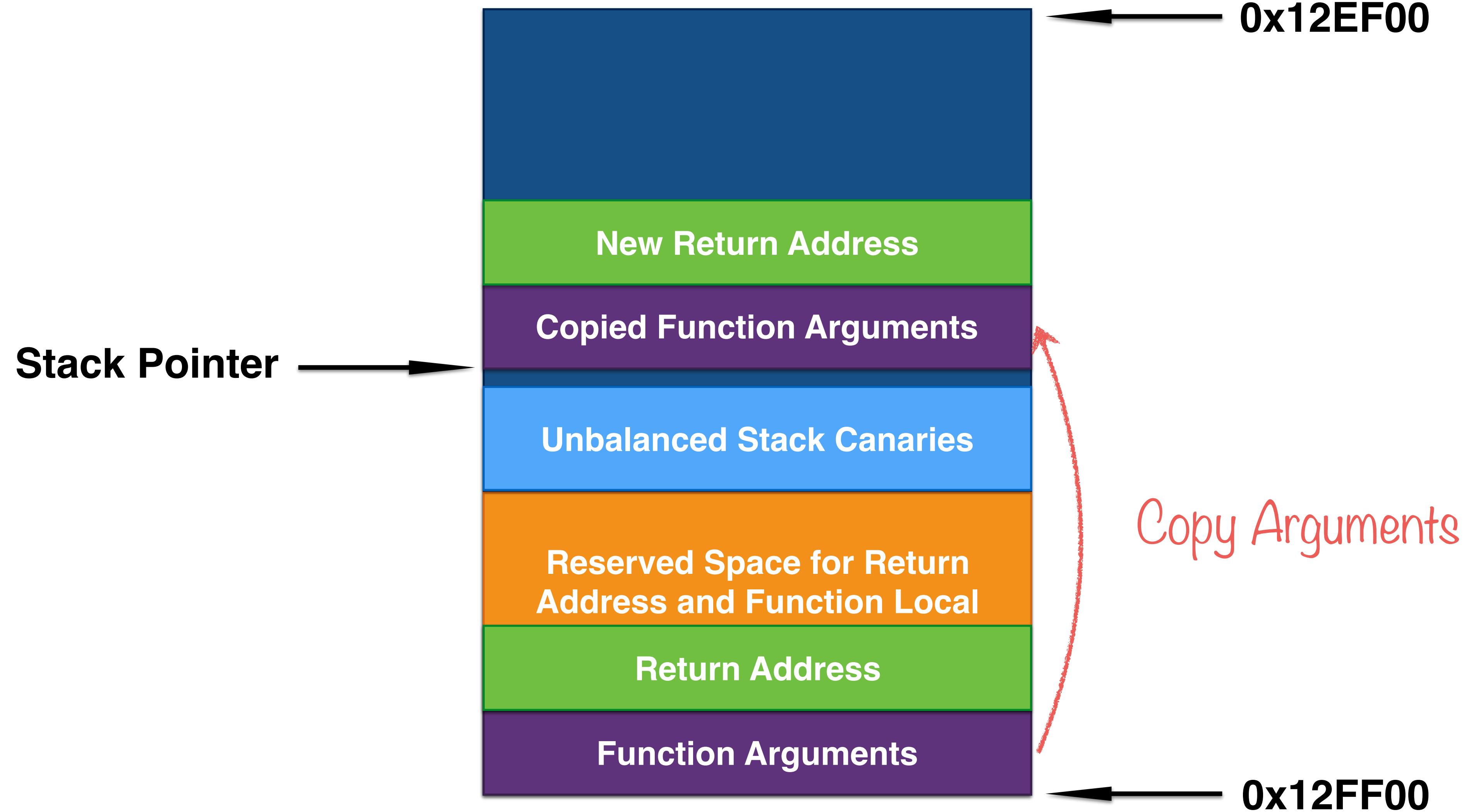
Unbalanced Stack



Fix

Cuckoo: Monitor

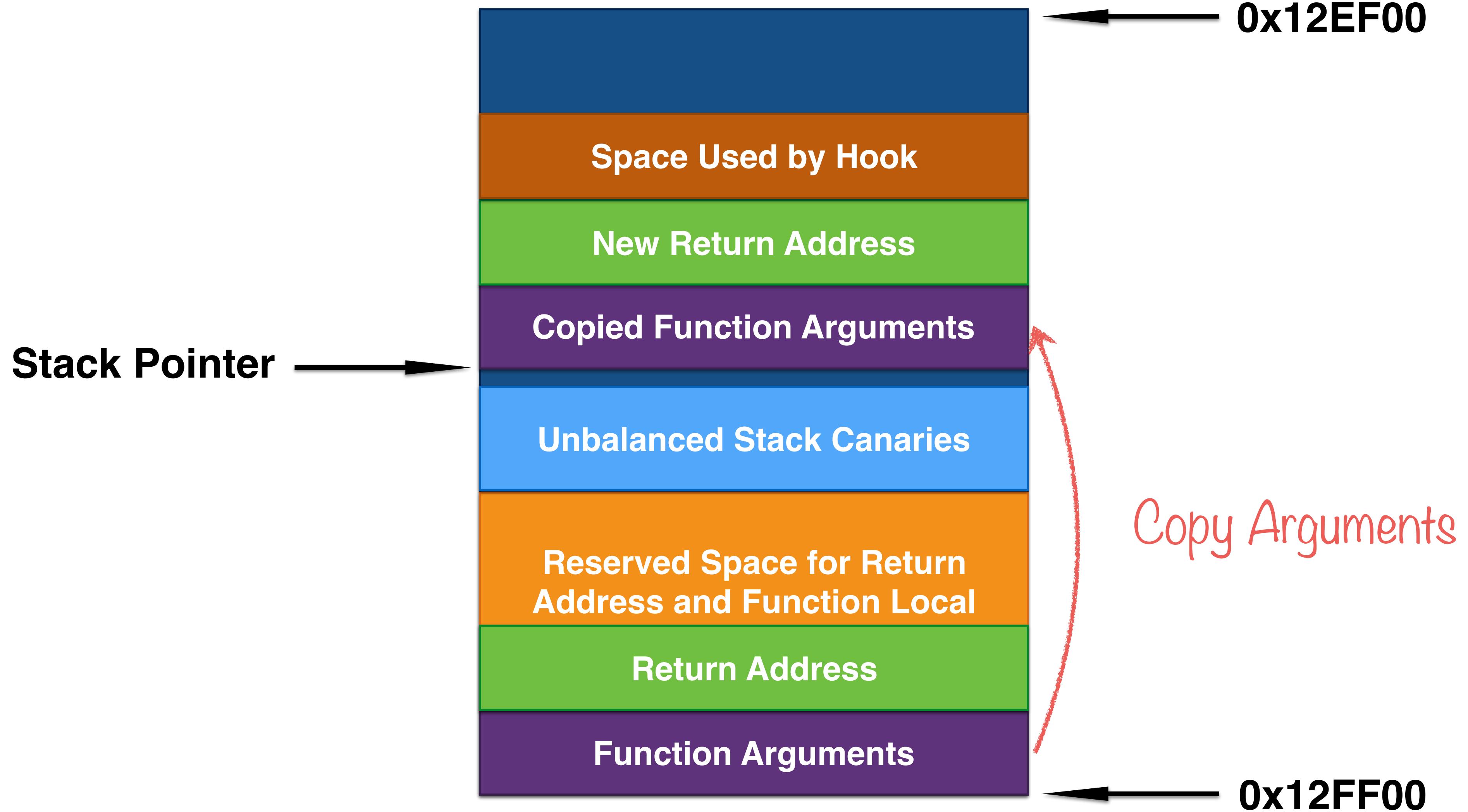
Unbalanced Stack



Fix

Cuckoo: Monitor

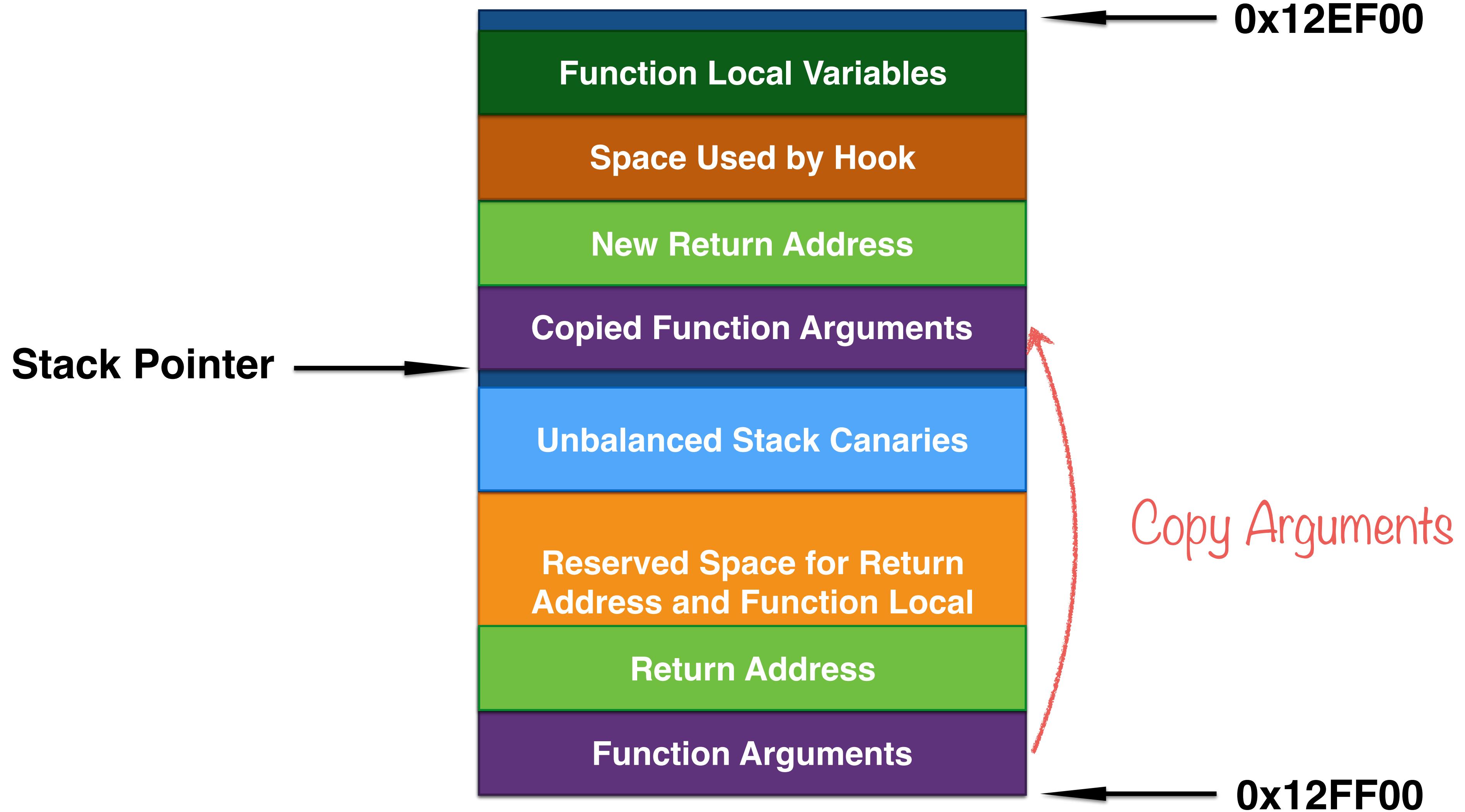
Unbalanced Stack



Fix

Cuckoo: Monitor

Unbalanced Stack



Info

Cuckoo: Monitor

Sleep Skip: Main Logic

```
int sleep_skip(LARGE_INTEGER *delay) {
    // code goes here

    if (current_time.QuadPart < g_time_start.QuadPart + g_sleep_max_skip * 10000) {
        g_time_skipped.QuadPart += -delay.QuadPart;
        delay.QuadPart -= 1000; // replace the time by tenth of millisecond
        return 1;
    }

    // code goes here
    return 0;
}
```

Info

Cuckoo: Monitor

Sleep Skip: Main Logic

```
int sleep_skip(LARGE_INTEGER *delay) {
    // code goes here

    if (current_time.QuadPart < g_time_start.QuadPart + g_sleep_max_skip * 10000) {
        g_time_skipped.QuadPart += -delay.QuadPart;
        delay.QuadPart -= 1000; // replace the time by tenth of millisecond
        return 1;
    }

    // code goes here
    return 0;
}
```

1

Cuckoo: Monitor

Sleep Skip: Main Logic

```
int sleep_skip(LARGE_INTEGER *delay) {
    // code goes here

    if (current_time.QuadPart < g_time_start.QuadPart + g_sleep_max_skip * 10000) {
        1
        g_time_skipped.QuadPart += -delay.QuadPart;
        2
        delay.QuadPart -= 1000; // replace the time by tenth of millisecond
        return 1;
    }

    // code goes here
    return 0;
}
```

1

2

Cuckoo: Monitor

Sleep Skip: Main Logic

```
int sleep_skip(LARGE_INTEGER *delay) {
    // code goes here

    if (current_time.QuadPart < g_time_start.QuadPart + g_sleep_max_skip * 10000) {
        1
        g_time_skipped.QuadPart += -delay.QuadPart;
        2
        delay.QuadPart = -1000; // replace the time by tenth of millisecond
        return 1;
    }
    3
    // code goes here
    return 0;
}
```

Info

Cuckoo: Monitor

Sleep Skip: Usage

sleep_skip()

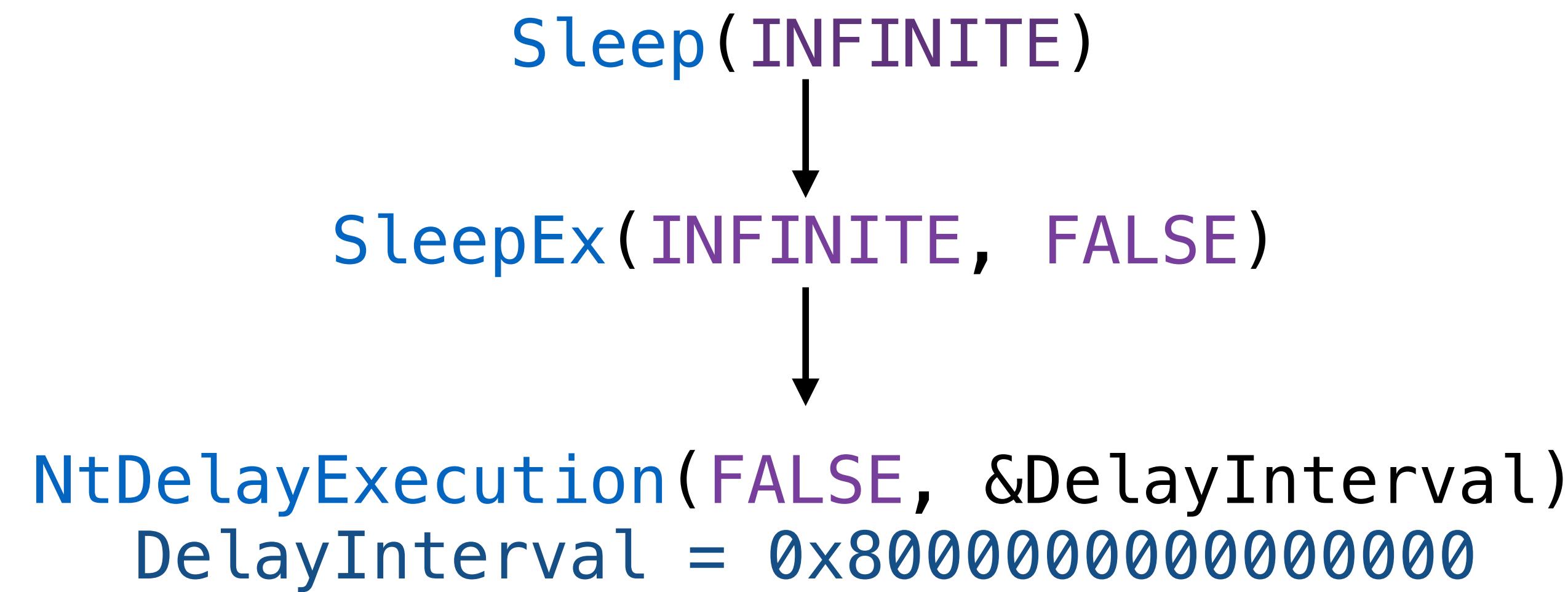
- NtDelayExecution

g_time_skipped

- NtQuerySystemTime
- GetTickCount
- GetLocalTime
- GetSystemTime
- GetSystemTimeAsFileTime

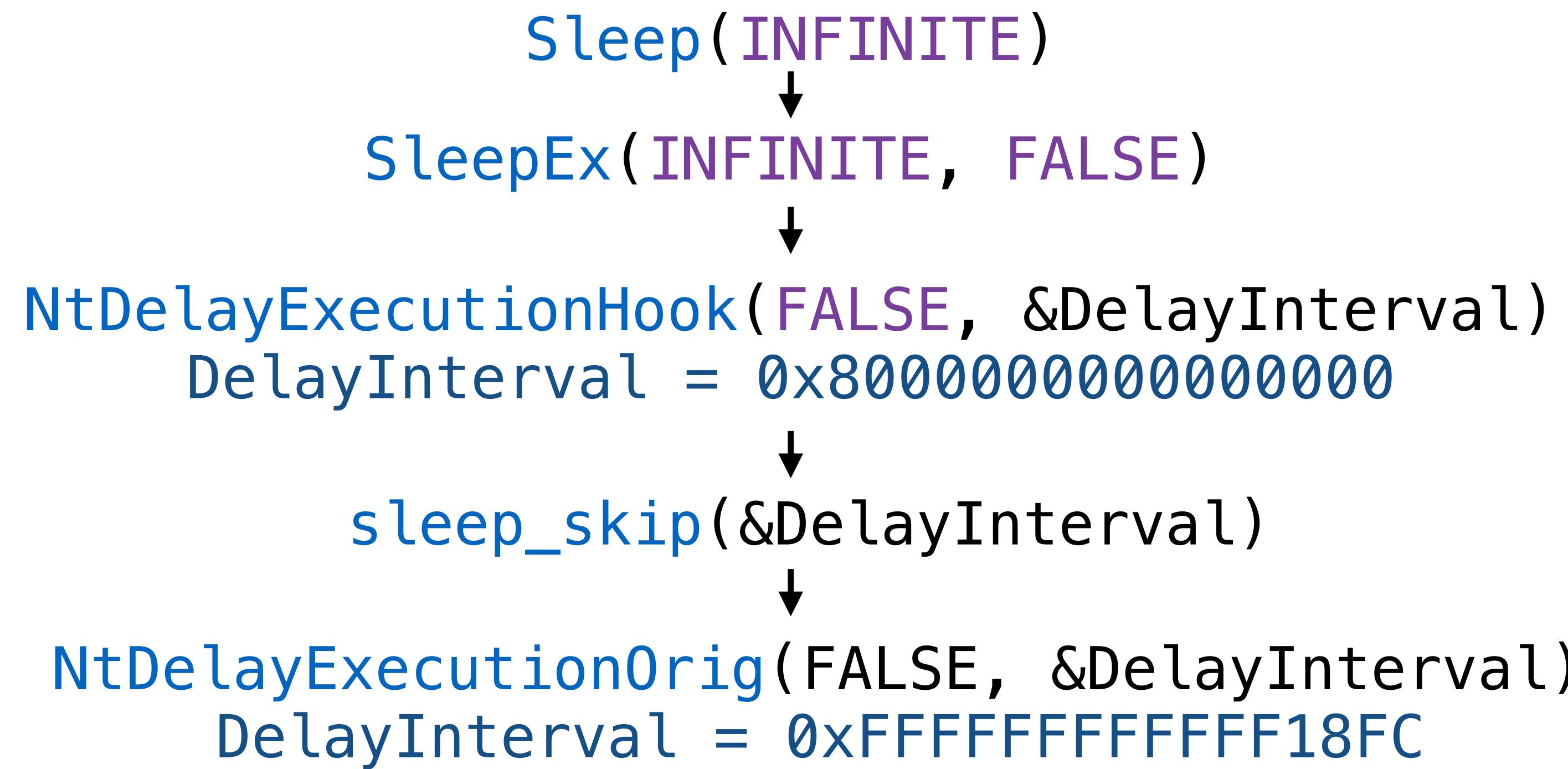
Info

Cuckoo: Monitor **INFINITE Delay**



Thread Sleeps Forever.

Cuckoo: Monitor INFINITE Delay



Thread Sleeps Only for 0.1ms => Cuckoo Detected.

Fix

Cuckoo: Monitor

INFINITE Delay

```
int sleep_skip(LARGE_INTEGER *delay) {  
    if (delay->QuadPart == 0x8000000000000000ll) return 0;  
  
    // code goes here  
  
    if (current_time.QuadPart < g_time_start.QuadPart + g_sleep_max_skip * 10000) {  
        g_time_skipped.QuadPart += -delay.QuadPart;  
        delay.QuadPart -= 1000; // replace the time by tenth of millisecond  
        return 1;  
    }  
  
    // code goes here  
    return 0;  
}
```

Info

Cuckoo: Monitor

Delays Accumulation

Thread #1

```
void zzz_get_st(LARGE_INTEGER *st)
{
    SleepEx(2000 * 1000, FALSE);
    NtQuerySystemTime(st);
}
```

NtDelayExecutionHook(st):
g_time_skipped.QuadPart += -2000*1000

NtQuerySystemTimeHook(st):
st->QuadPart += g_time_skipped.QuadPart

Skipped delays are added to current time.

Detection

Cuckoo: Monitor Delays Accumulation

Thread #1

Thread #2

Detection

Cuckoo: Monitor Delays Accumulation

Thread #1

```
LARGE_INTEGER st_start, st_end;  
HANDLE h0bj;  
NtQuerySystemTime(&st_start);  
WaitForSingleObject(h0bj, 100); // 100ms
```

Thread #2

Detection

Cuckoo: Monitor Delays Accumulation

Thread #1

```
LARGE_INTEGER st_start, st_end;  
HANDLE h0bj;  
NtQuerySystemTime(&st_start);  
WaitForSingleObject(h0bj, 100); // 100ms
```

Thread #2

```
SleepEx(1000*60*60*24*3, FALSE); // 3 days
```



```
g_time_skipped.QuadPart += -3days
```

Detection

Cuckoo: Monitor Delays Accumulation

Thread #1

```
LARGE_INTEGER st_start, st_end;  
HANDLE h0bj;  
NtQuerySystemTime(&st_start);  
WaitForSingleObject(h0bj, 100); // 100ms  
  
NtQuerySystemTime(&st_end);  
if (st_end - st_start > 2 days)  
    halt("Cuckoo sleep hooking detected")
```

Thread #2

```
SleepEx(1000*60*60*24*3, FALSE); // 3 days
```



```
g_time_skipped.QuadPart += -3days
```

Detection

Cuckoo: Monitor Delays Accumulation

Thread #1

```
LARGE_INTEGER st_start, st_end;  
HANDLE h0bj;  
NtQuerySystemTime(&st_start);  
WaitForSingleObject(h0bj, 100); // 100ms  
  
NtQuerySystemTime(&st_end);  
if (st_end - st_start > 2 days)  
    halt("Cuckoo sleep hooking detected")
```

Thread #2

```
SleepEx(1000*60*60*24*3, FALSE); // 3 days
```



```
g_time_skipped.QuadPart += -3days
```

After waiting for 100ms for some object, system time will change for about 3 days.

Cuckoo: Monitor

Delays Skipping: Behavioral Analysis Problem

Behavioral Analysis Problem

- Within the first `g_sleep_max_skip` seconds perform time consuming operations.
- Perform many large delay sleeps, thus exceeding critical timeout for one execution.
- Perform malicious activities.

Proposed Solution

- Skip all delays that are larger than `g_sleep_min_skip_delay` seconds.
- For smaller delays, use sliding window technique.

Fix

Cuckoo: Monitor

Delays Skipping: Sliding Window

Description

- Collect the number of hits for delays within the last **n** seconds.
- If the number of hits for a delay exceeds upper limit (**m** hits), then skip that delay till the end of execution.

Pros

- All large delays are skipped.
- Small delays in limited set are skipped.
- Works pretty stable and has very high emulation rate in test environment.

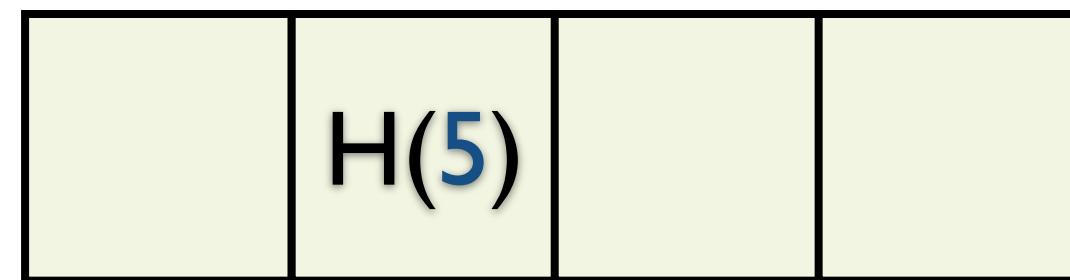
Cons

- Distribution of delays between **0** and **g_sleep_min_skip_delay** may be high, thus for specific delays the upper limit will not be exceeded.

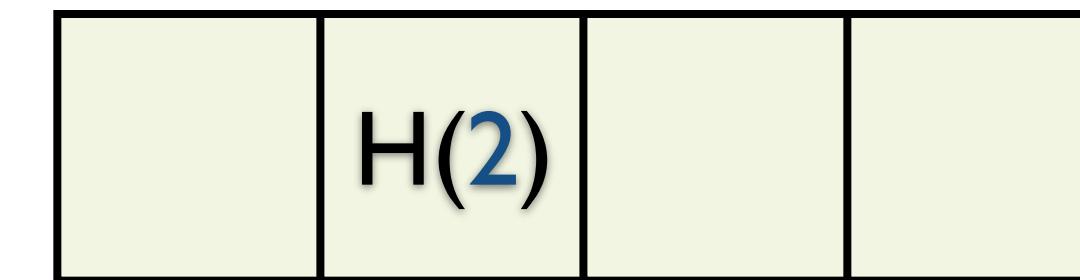
Demo

Cuckoo: Monitor

Delays Skipping: Sliding Window



...



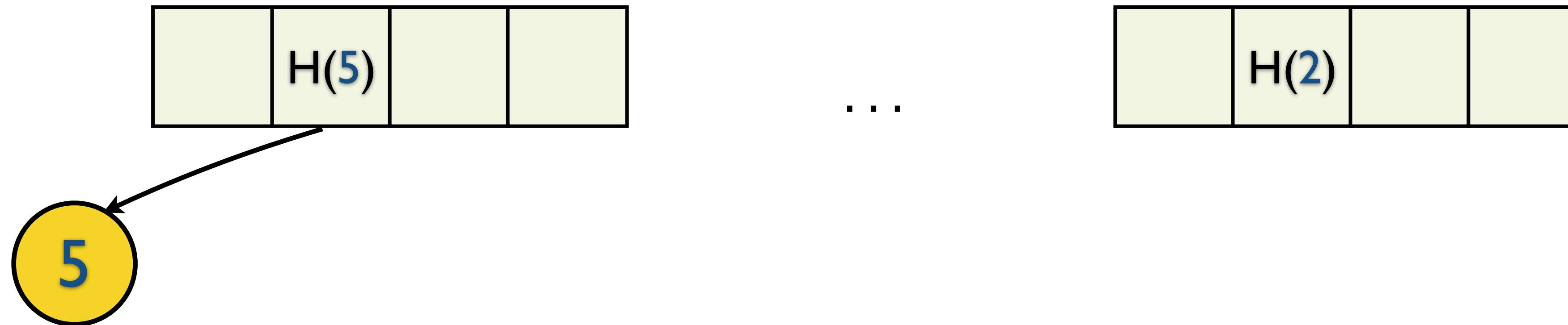
Sliding window: 10s

Threshold limit value: 3

Demo

Cuckoo: Monitor

Delays Skipping: Sliding Window



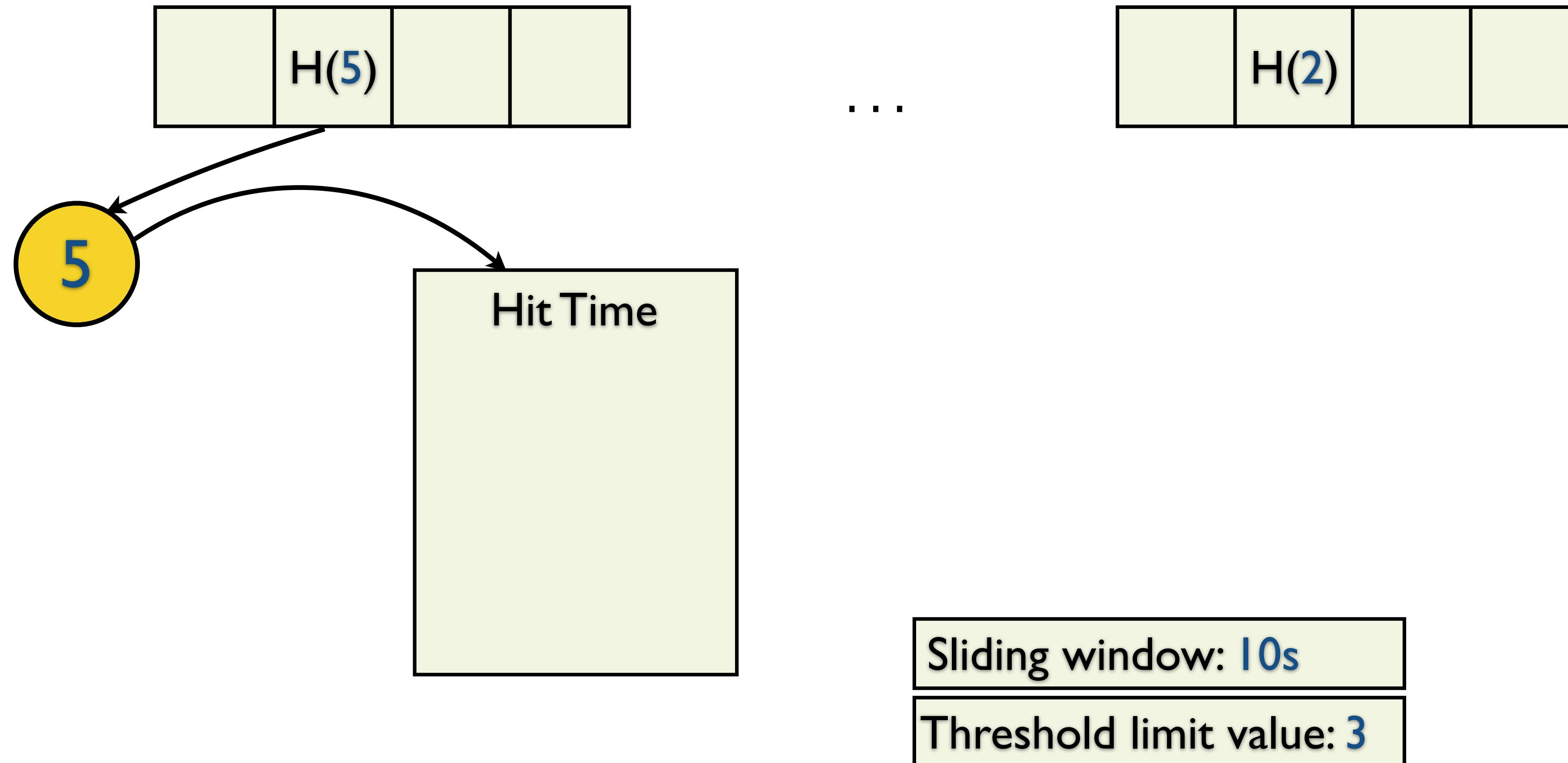
Sliding window: 10s

Threshold limit value: 3

Demo

Cuckoo: Monitor

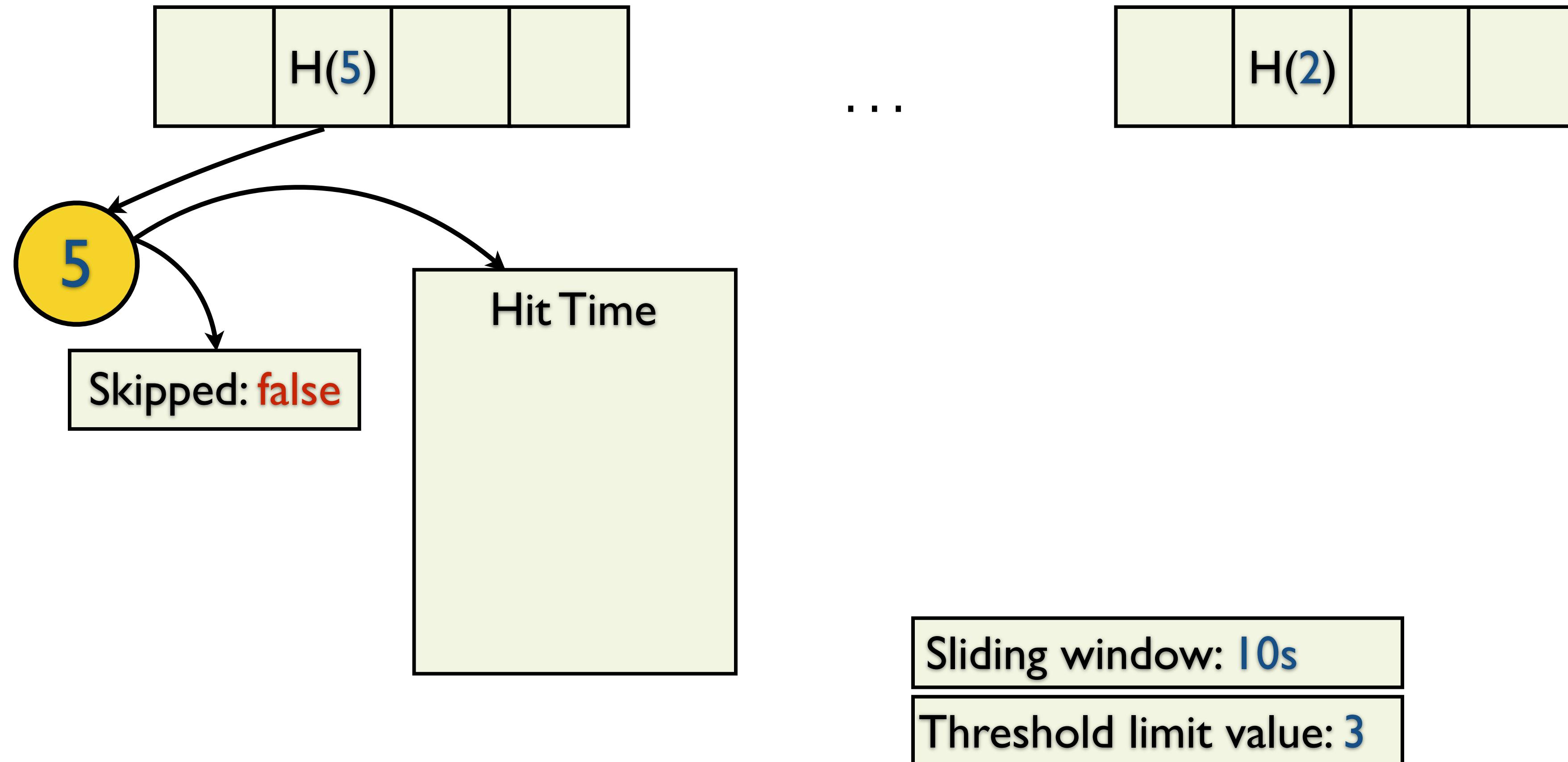
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

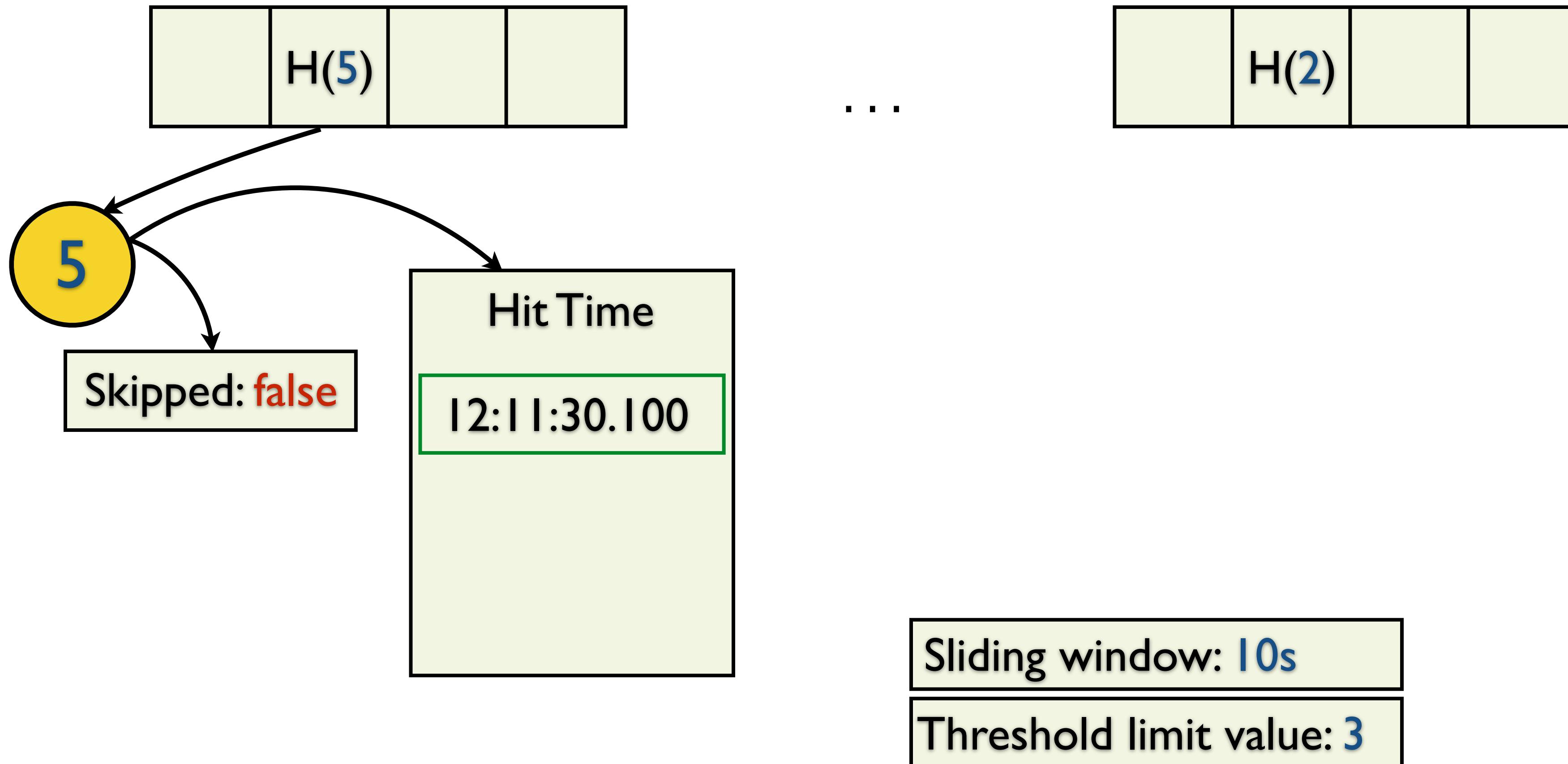
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

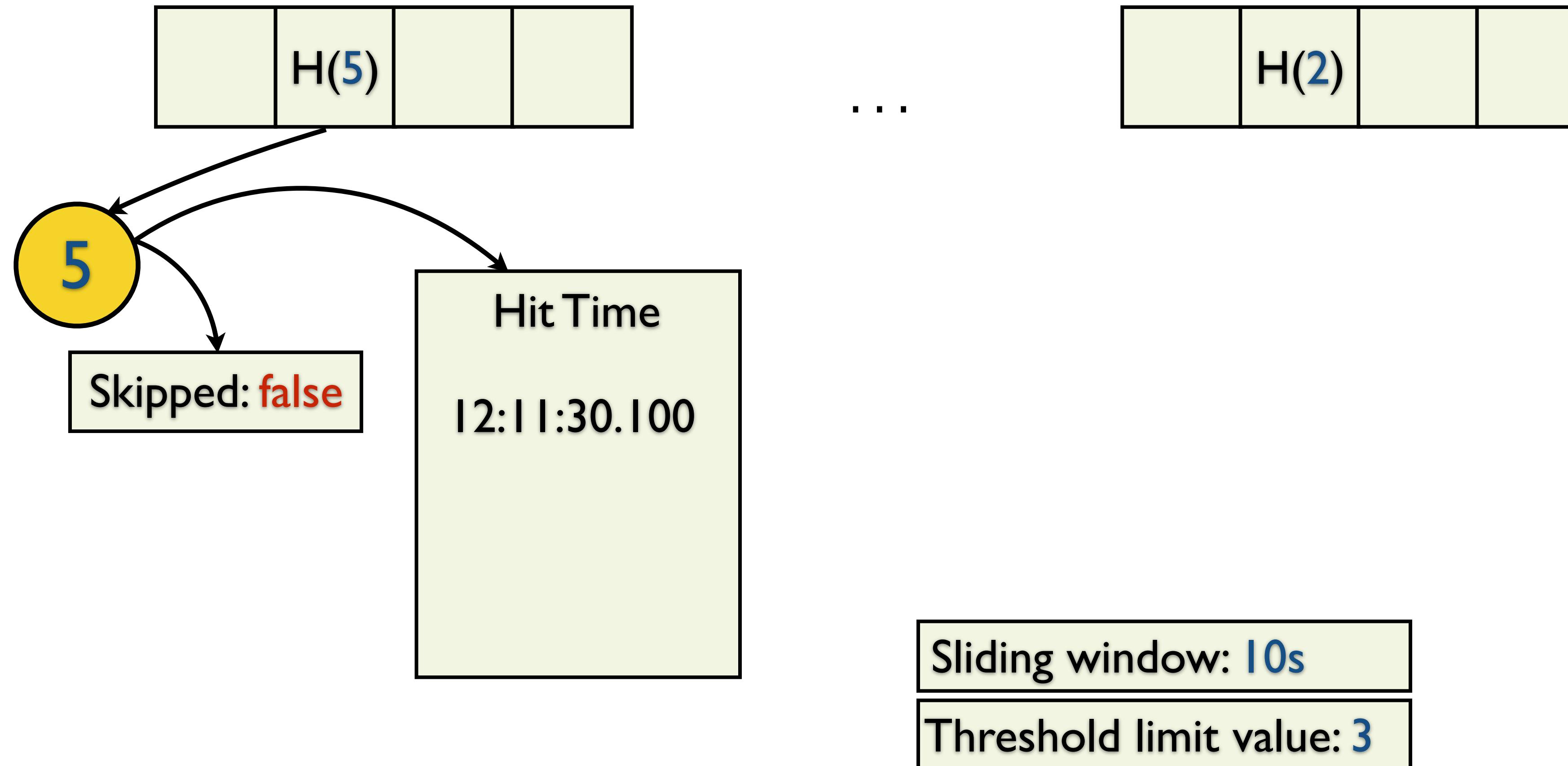
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

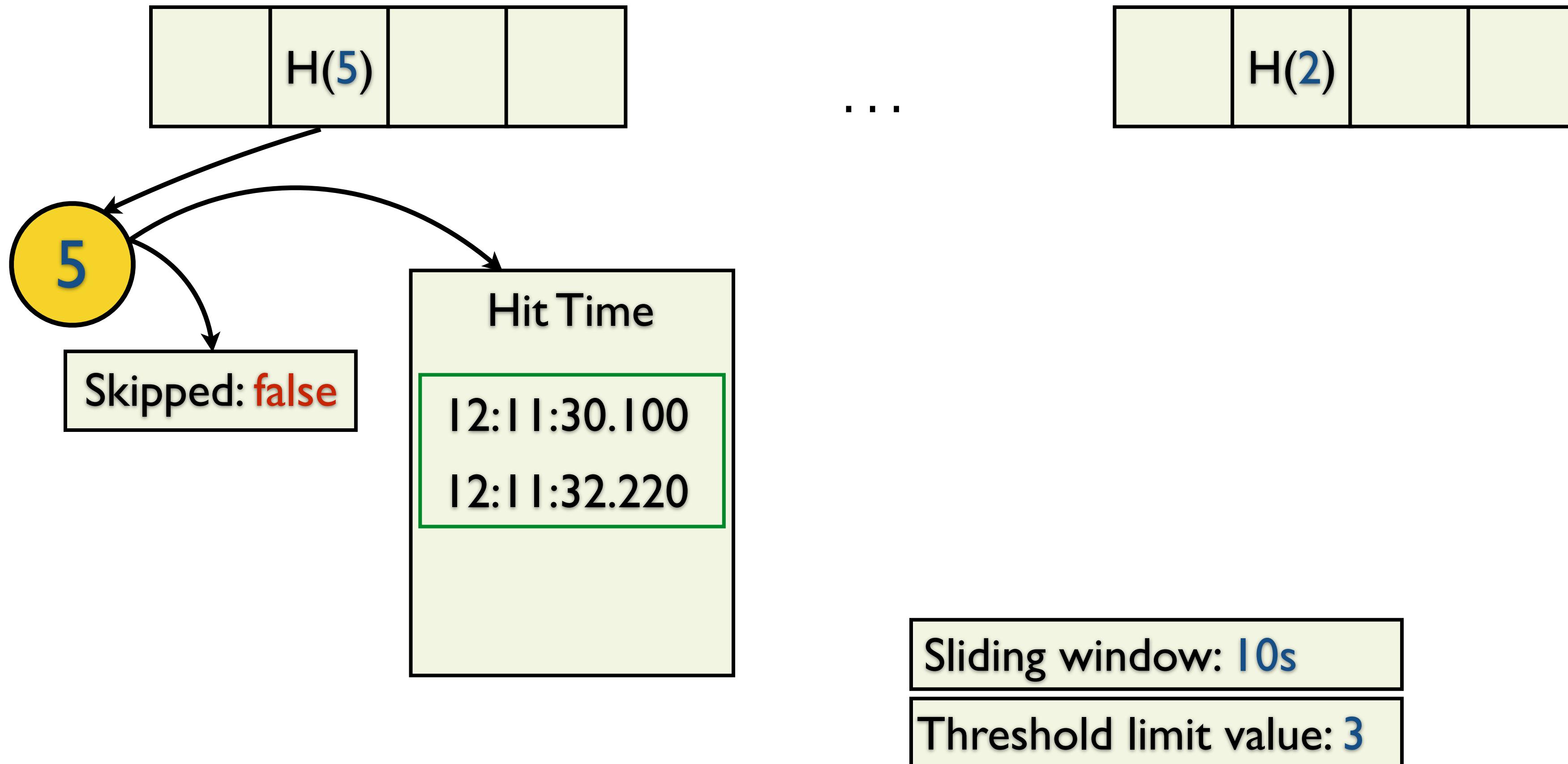
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

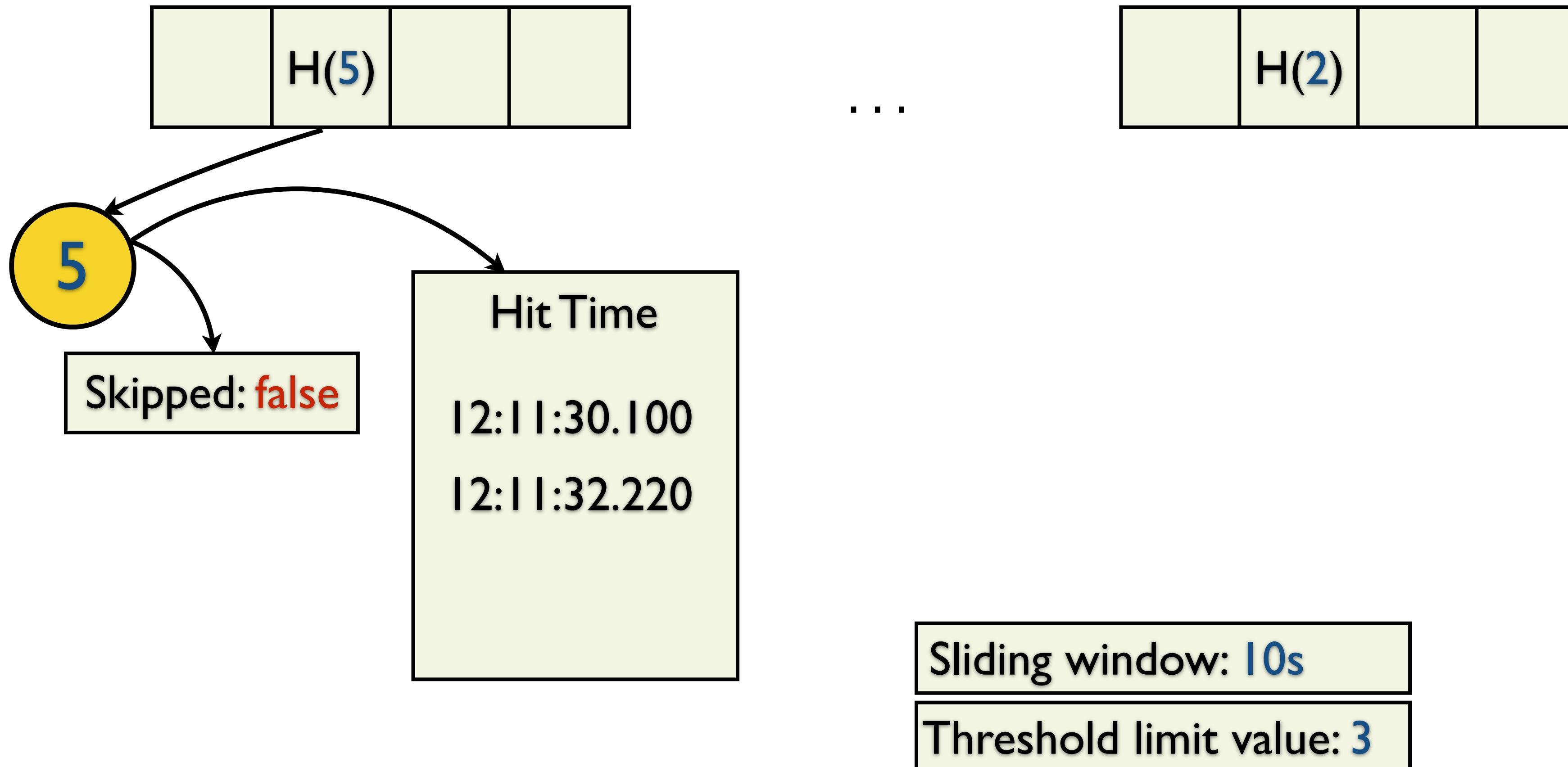
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

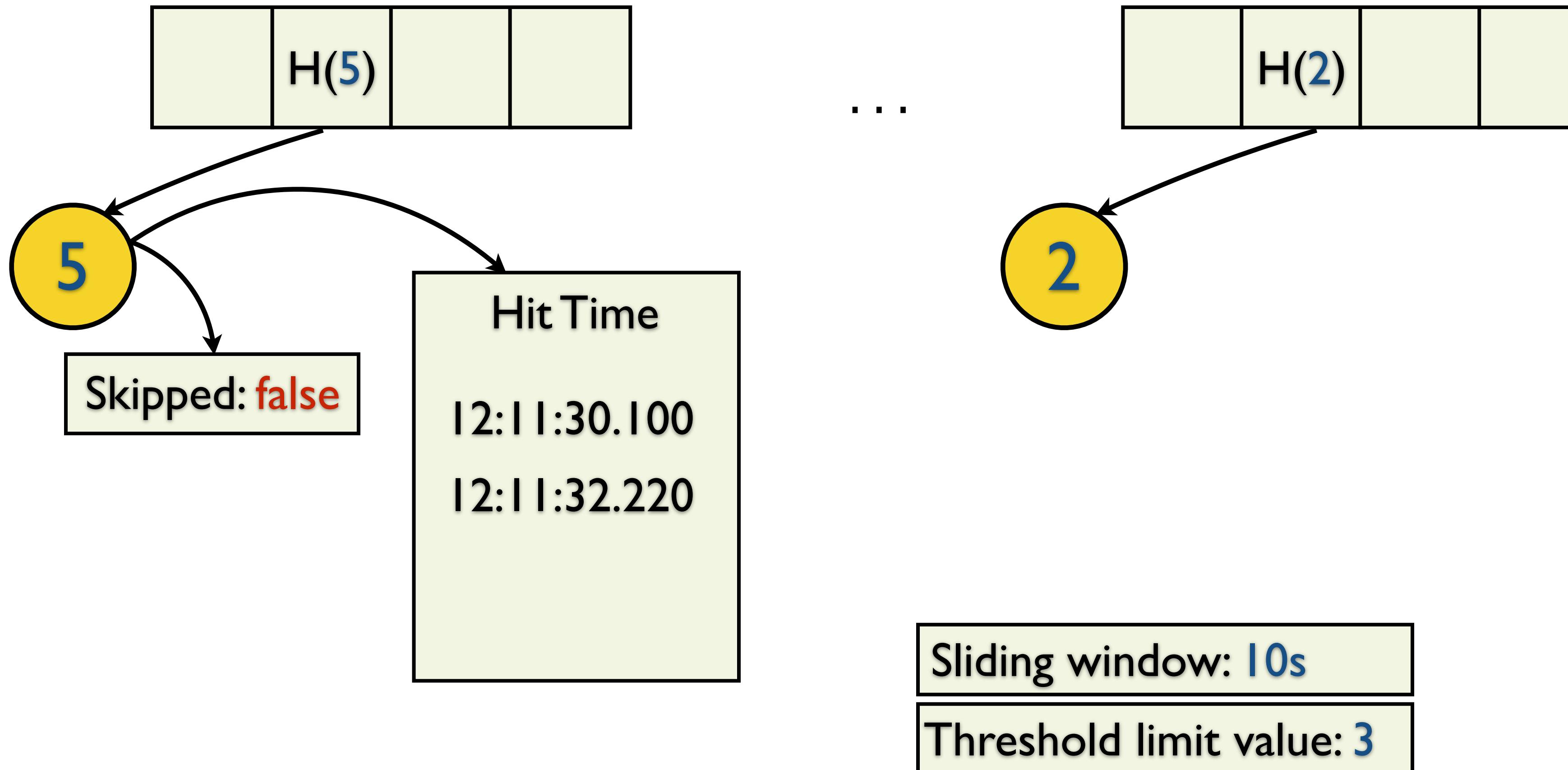
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

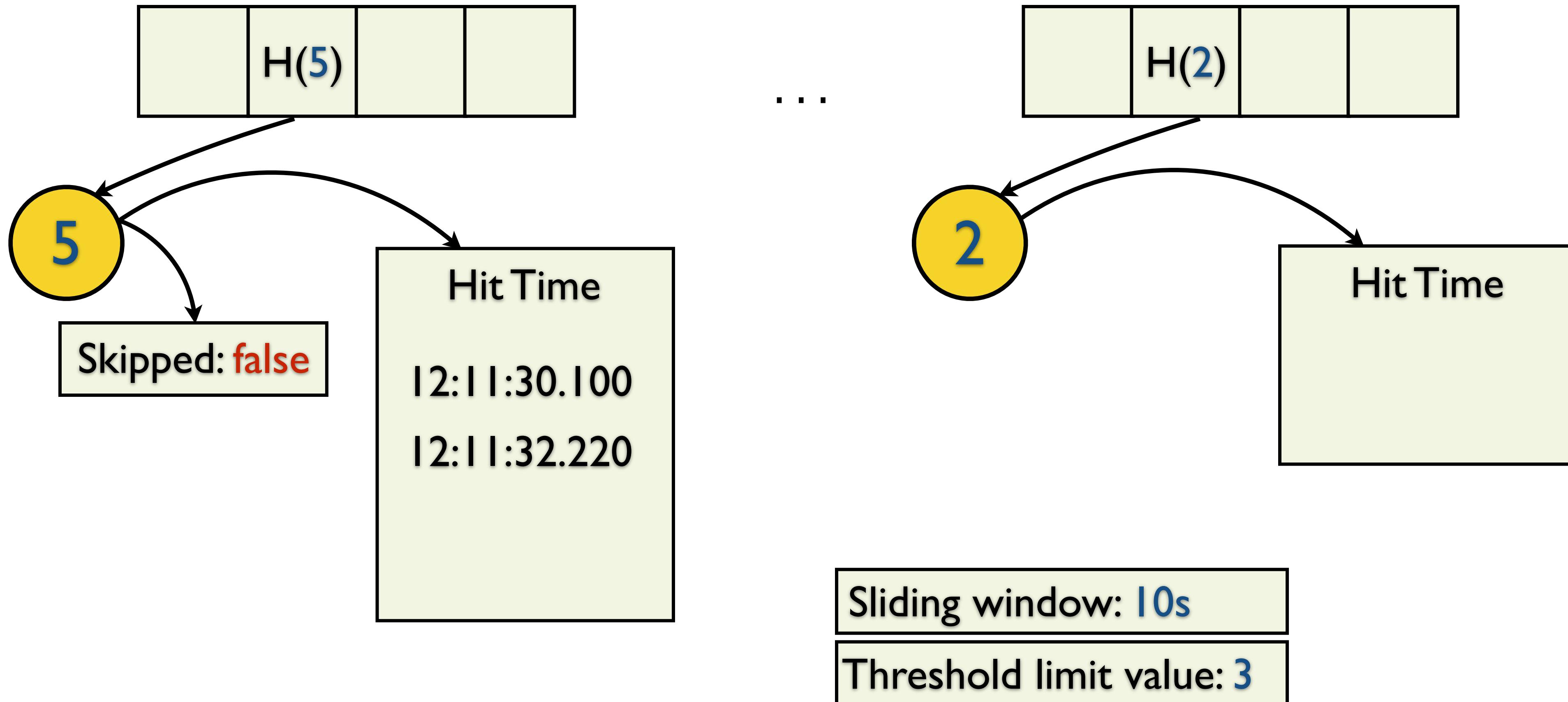
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

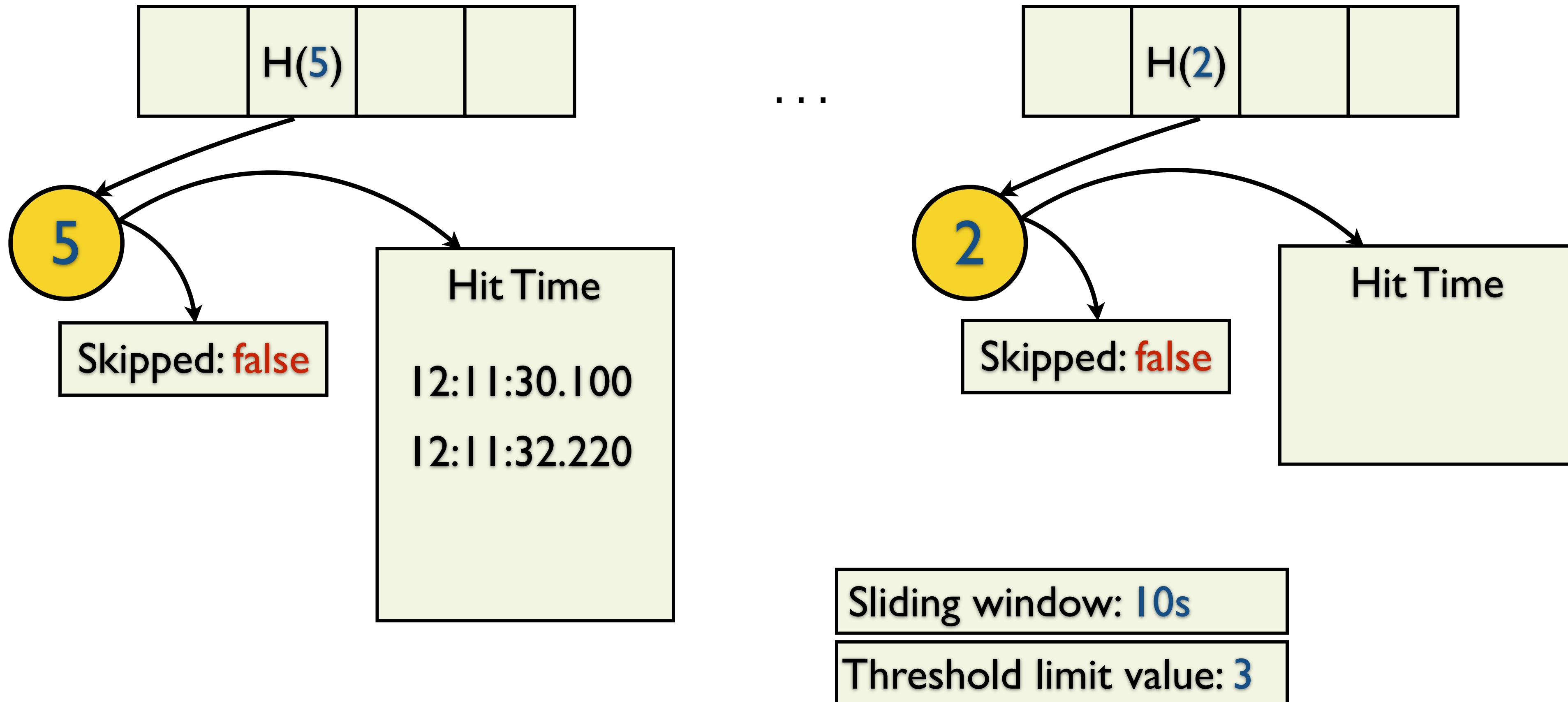
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

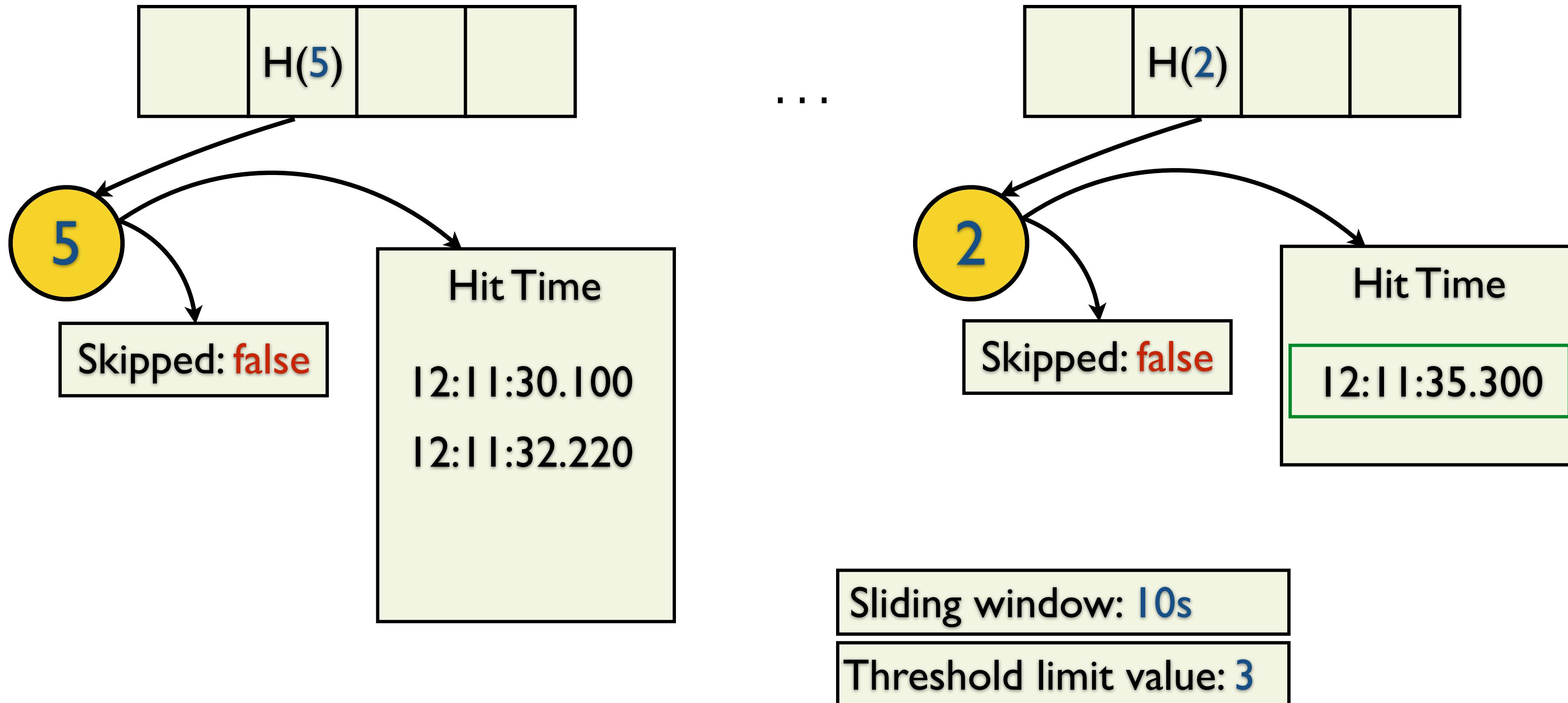
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

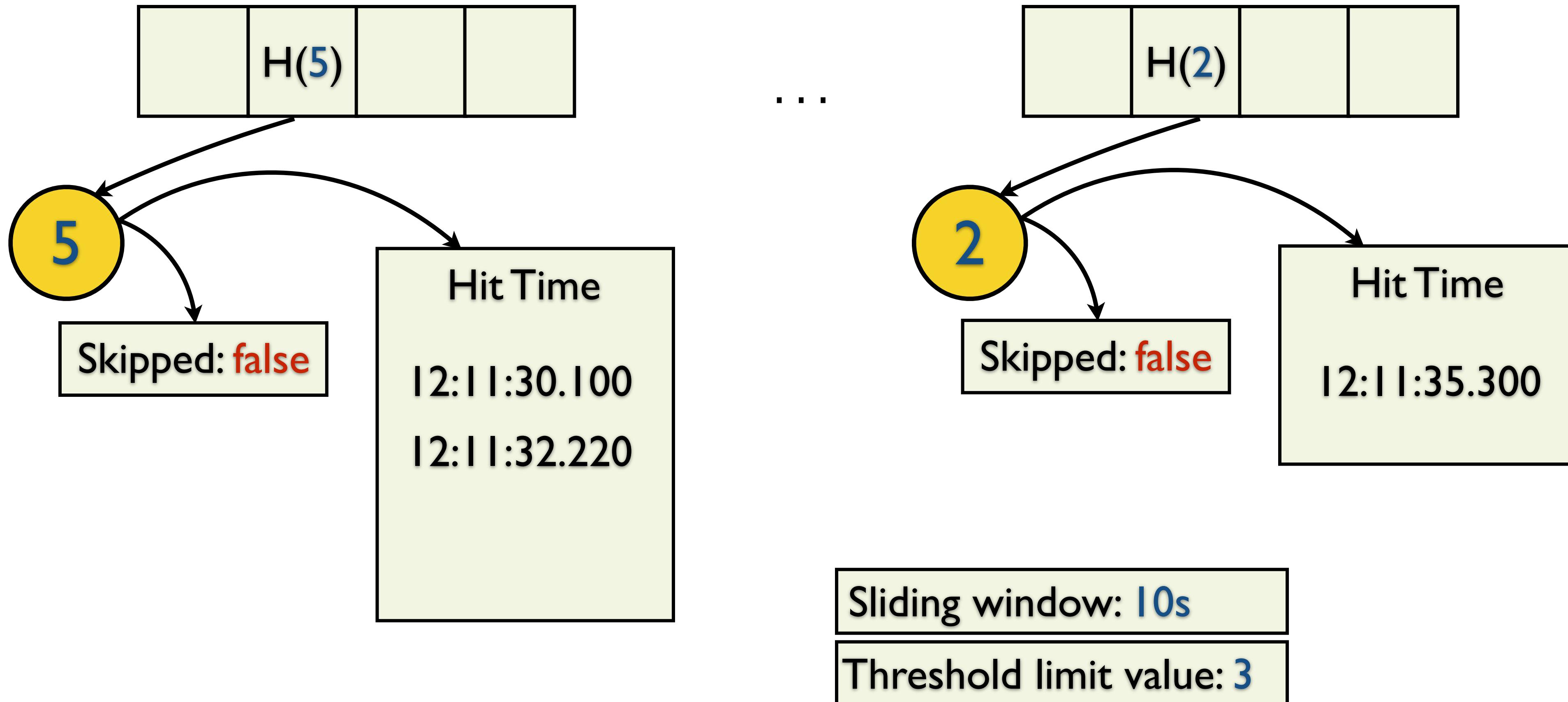
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

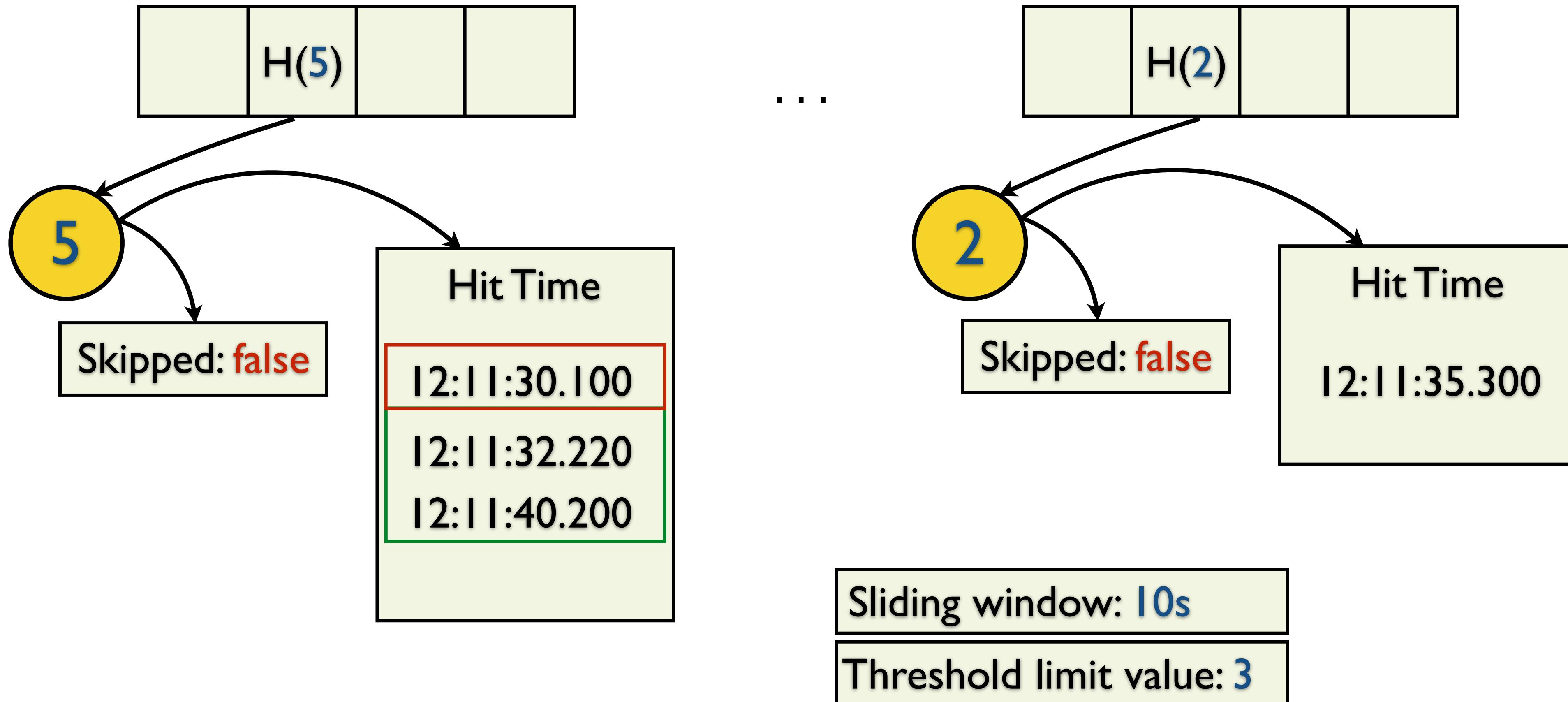
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

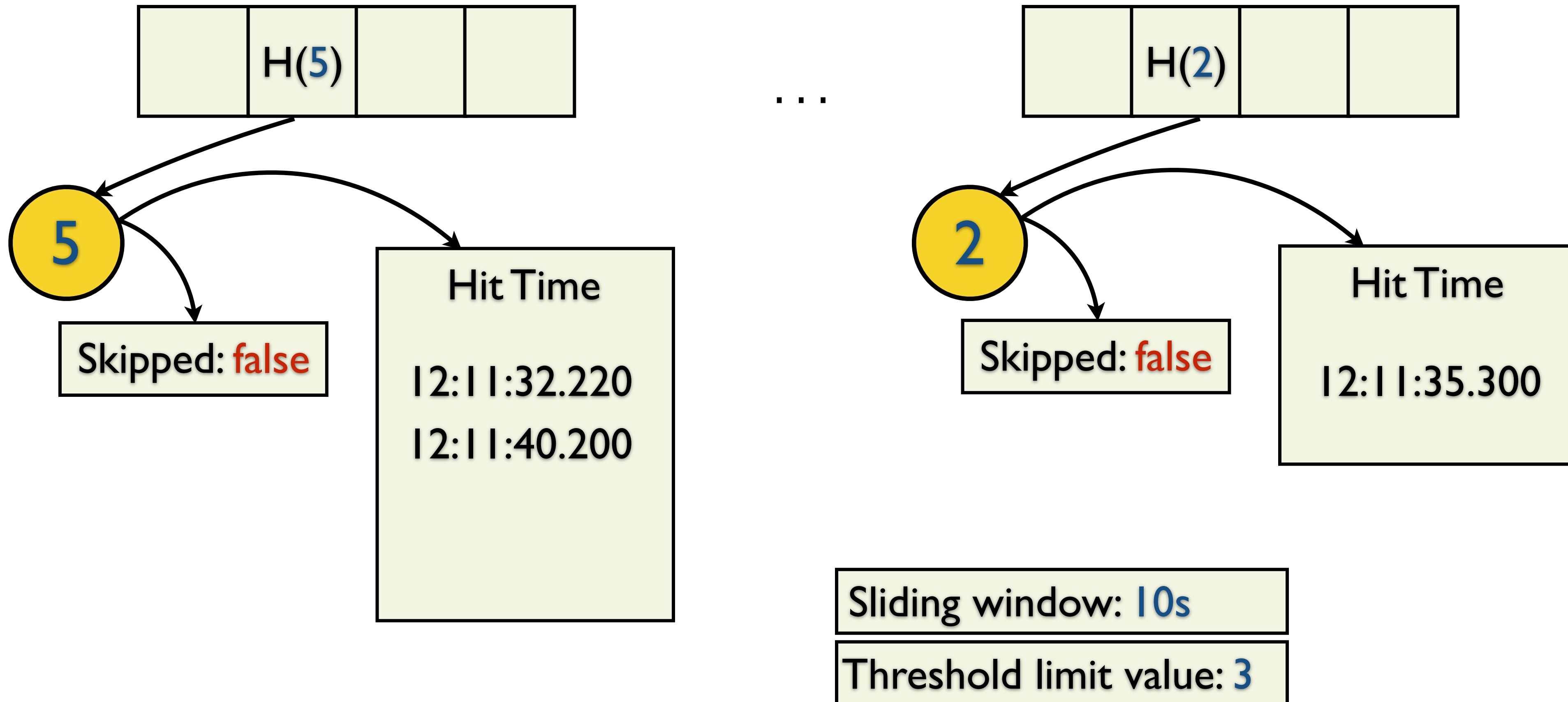
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

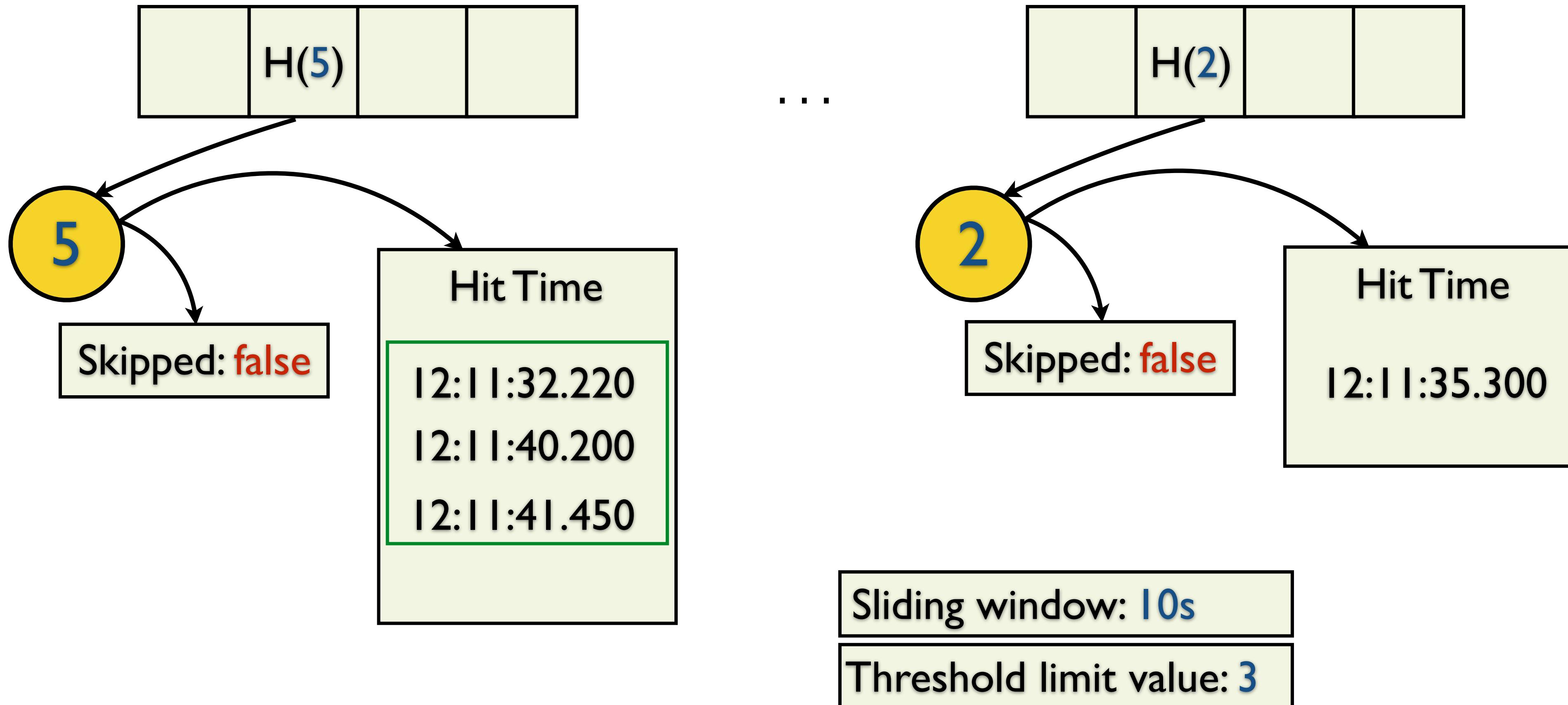
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

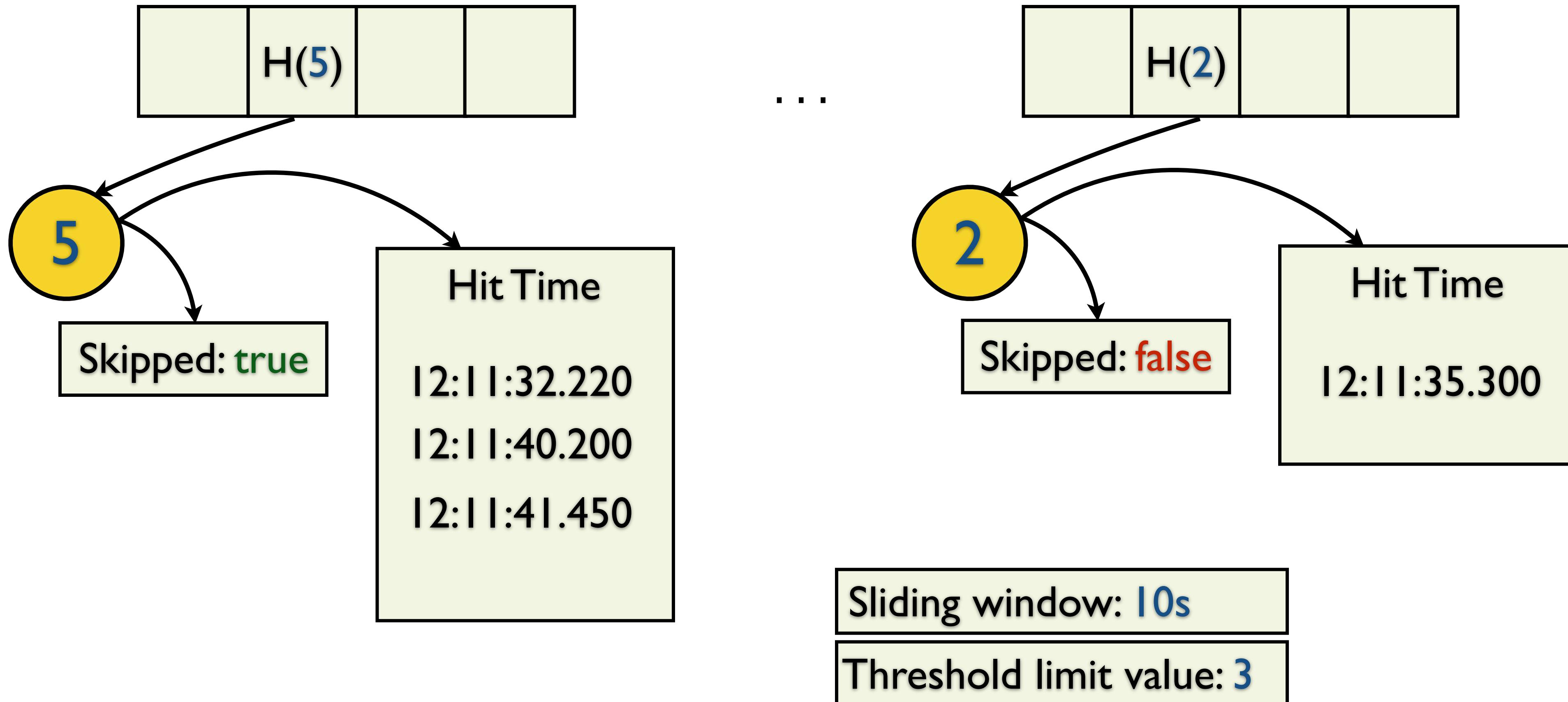
Delays Skipping: Sliding Window



Demo

Cuckoo: Monitor

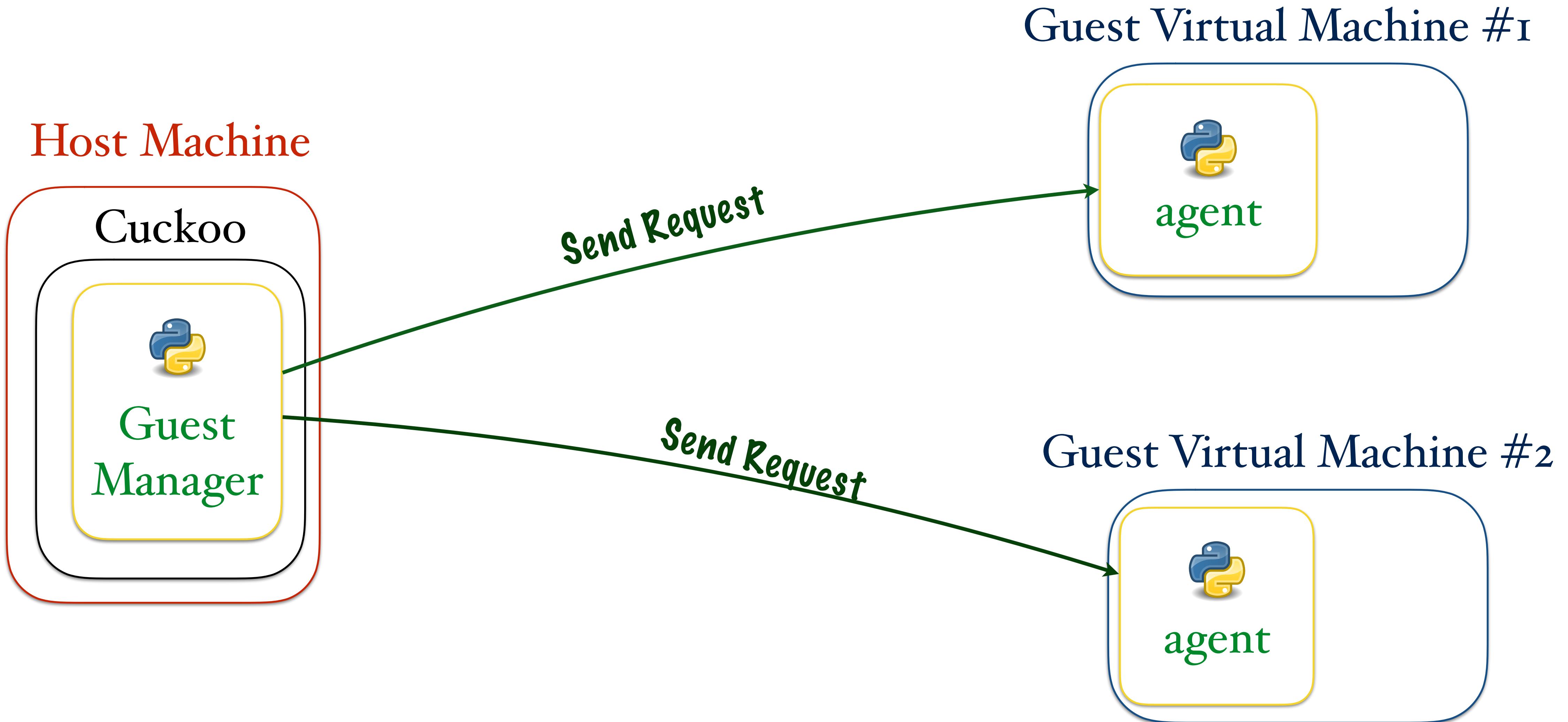
Delays Skipping: Sliding Window



Info

Cuckoo: Sandbox

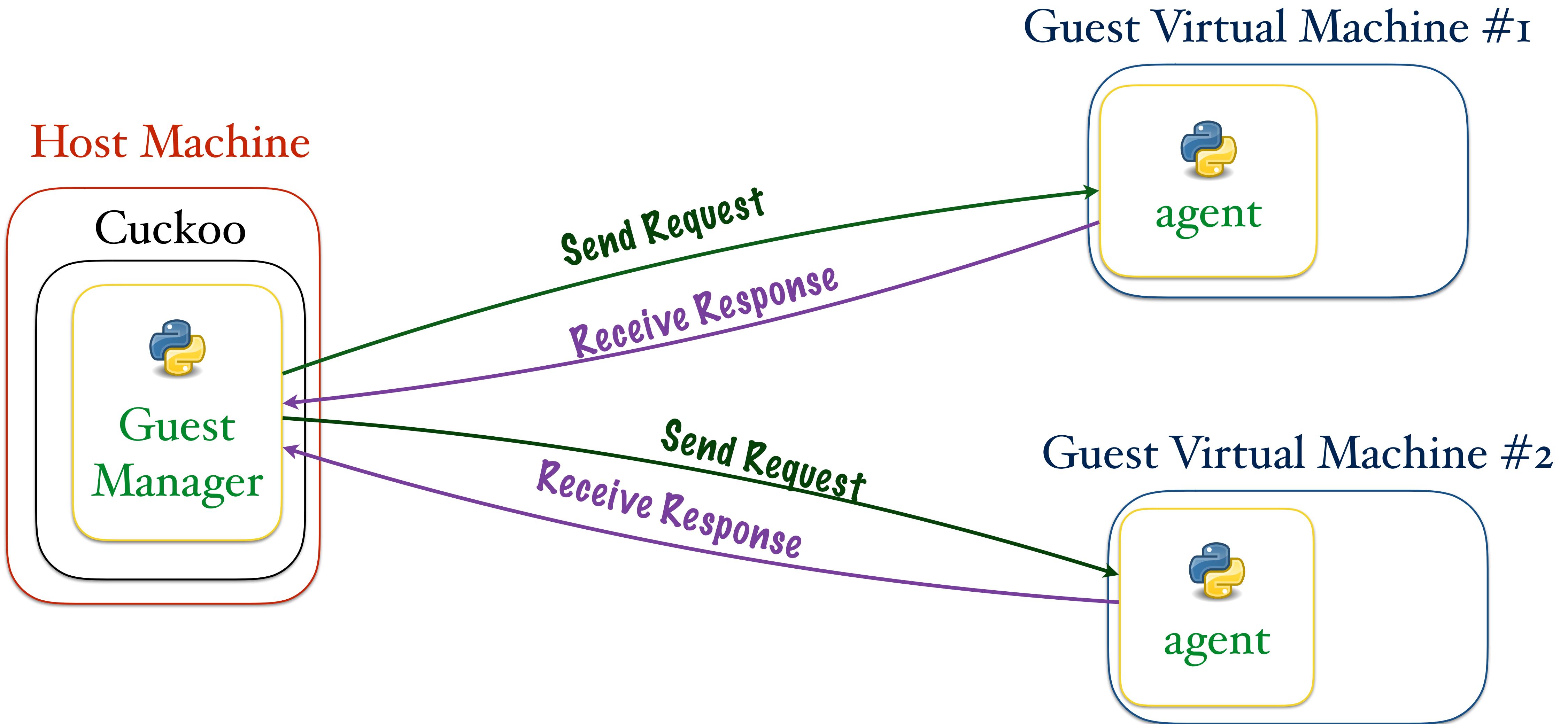
Host-Guest: Communication Architecture



Info

Cuckoo: Sandbox

Host-Guest: Communication Architecture



Cuckoo: Sandbox

Agent Module

Features

- Communication with the host machine
- Guest machine initialization
- Start module responsible for malware tracking

Some implementation details

- Listens on all interfaces on 8000 port (detection may be performed on any port)
- Uses *SimpleXMLRPCServer* class for communication interface

Cuckoo: Sandbox

Agent Module

Detection Technique

1. Enumerate all *LISTENING* sockets.
2. Send crafted packet and wait for response.
3. If specific response is received, then assume that we are running in Cuckoo Sandbox environment.

Crafted Packet

```
<methodCall>
<methodName>get_status</methodName>
<params></params>
</methodCall>
```

Response Regex

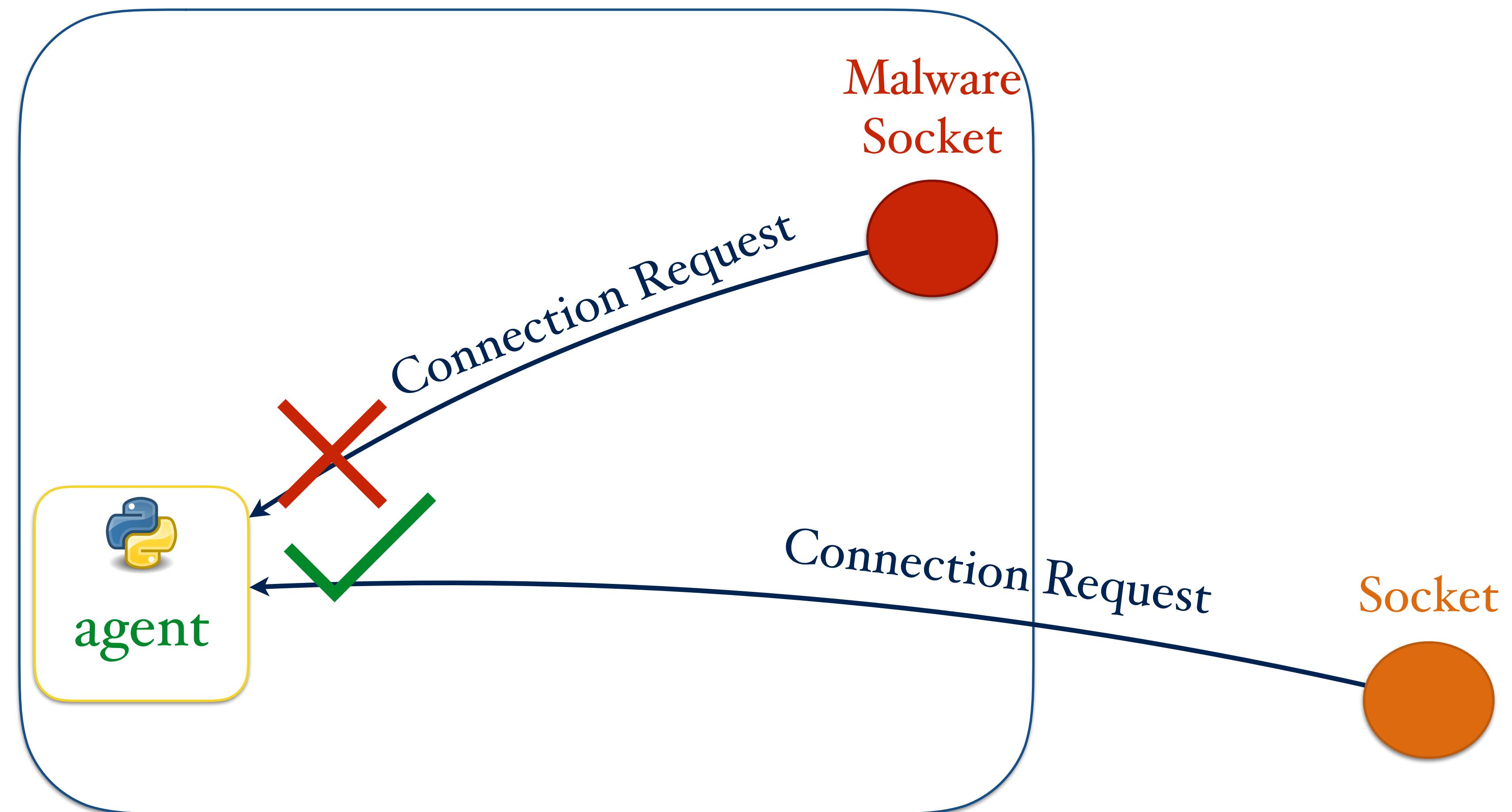
```
<methodResponse>
<params>
<param>
<value><int>[[digit]]</int></value>
</param>
</params>
</methodCall>
```

Fix

Cuckoo: Sandbox

Agent Module

Guest Virtual Machine



Fix

Cuckoo: Sandbox

Agent Module

```
class SimpleXMLRPCServerFilter(SimpleXMLRPCServer):  
    # !!! INITIALIZATION GOES HERE !!!  
  
    # overridden method  
    def verify_request(self, request, client_address):  
        for f_ip in self._forbid_ips:  
            if f_ip == client_address[0]:  
                return False  
  
        return True
```

Cuckoo: Sandbox Analyzer Module

Features

- Initialization of analysis procedure
- Execution of analysis procedure, thus handling pipe commands and managing injections

Implementation details

- Started by Agent module
- Location is calculated in the “random” way

Cuckoo: Sandbox

Analyzer Module: Implementation

```
random.seed(time.time())
container = "".join(random.choice(string.ascii_lowercase) for x in
range(random.randint(5, 10)))
if self.system == "windows":
    system_drive = os.environ["SYSTEMDRIVE"] + os.sep
    self.analyzer_folder = os.path.join(system_drive, container)
self.analyzer_path = os.path.join(self.analyzer_folder, "analyzer.py")
config_path = os.path.join(self.analyzer_folder, "analysis.conf")
```



Cuckoo: Sandbox

Analyzer Module

Detection Technique

1. Enumerate all folders on the *SYSTEMDRIVE*.
2. Check if any folder contains `analyzer.py` and `analysis.conf` files.
3. If files are present, then assume that we are running in Cuckoo Sandbox environment.

Proposed Solution

- Hide the whole directory using filesystem filter driver.

Cuckoo: Sandbox

PID Tracking: Adding Process Id

```
# We inject the process only if it's not being monitored already
if self.analyzer.process_list.has_pid(process_id):
    some_operations_here()
    return

if not self.analyzer.files.is_protected_filename(filename):
    # Add the new process ID to the list of monitored processes.
    self.analyzer.process_list.add_pid(process_id)

# If we have both pid and tid, then we can use APC to inject.
if process_id and thread_id: proc.inject(dll, apc=True, mode="%s" % mode)
else: proc.inject(dll, apc=False, mode="%s" % mode)
```

Cuckoo: Sandbox

PID Tracking: Removing Process Id

```
# Check in the options if the user toggled the timeout enforce. If so,  
# we need to override pid_check and disable process monitor.  
if self.config.enforce_timeout:  
    log.info("Enabled timeout enforce, running for the full timeout.")  
    pid_check = False  
  
# If the process monitor is enabled we start checking whether  
# the monitored processes are still alive.  
if pid_check:  
    for pid in self.process_list.pids:  
        if not Process(pid=pid).is_alive():  
            log.info("Process with pid %s has terminated", pid)  
            self.process_list.remove_pid(pid)
```

Evasion

Cuckoo: Sandbox

PID Tracking: Enforced Timeout

```
# list of already used pids
pids = []
pid = None

while True:
    pid = create_process()
    if pid in pids:
        break
    kill_process(pid)
    pids.append(pid)

if pid is not None:
    print "Cuckoo Sandbox has been evaded."
```

Cuckoo: Sandbox

PID Tracking: Enforced Timeout

```
# list of already used pids
pids = []
pid = None

while True:
    pid = create_process()
    if pid in pids:
        break
    kill_process(pid)
    pids.append(pid)

if pid is not None:
    print "Cuckoo Sandbox has been evaded."
```

Analyzer's process_list already contains pid,
so Monitor will not be injected there =>
Cuckoo Evaded.

Fix

Cuckoo: Sandbox

PID Tracking: Enforced Timeout

```
def _handle_kill(self, data):
    """A process is being killed."""
    if not data.isdigit():
        log.warning("Received KILL command with an incorrect argument.")
        return

    if self.analyzer.config.options.get("procmemdump"):
        Process(pid=int(data)).dump_memory()

    self.analyzer.process_list.acquire()
    self.analyzer.process_list.remove_pid(int(data))
    self.analyzer.process_list.release()
```

Info

Cuckoo: Sandbox

Suspended Thread

Process #1

PIPE

Analyzer

Info

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

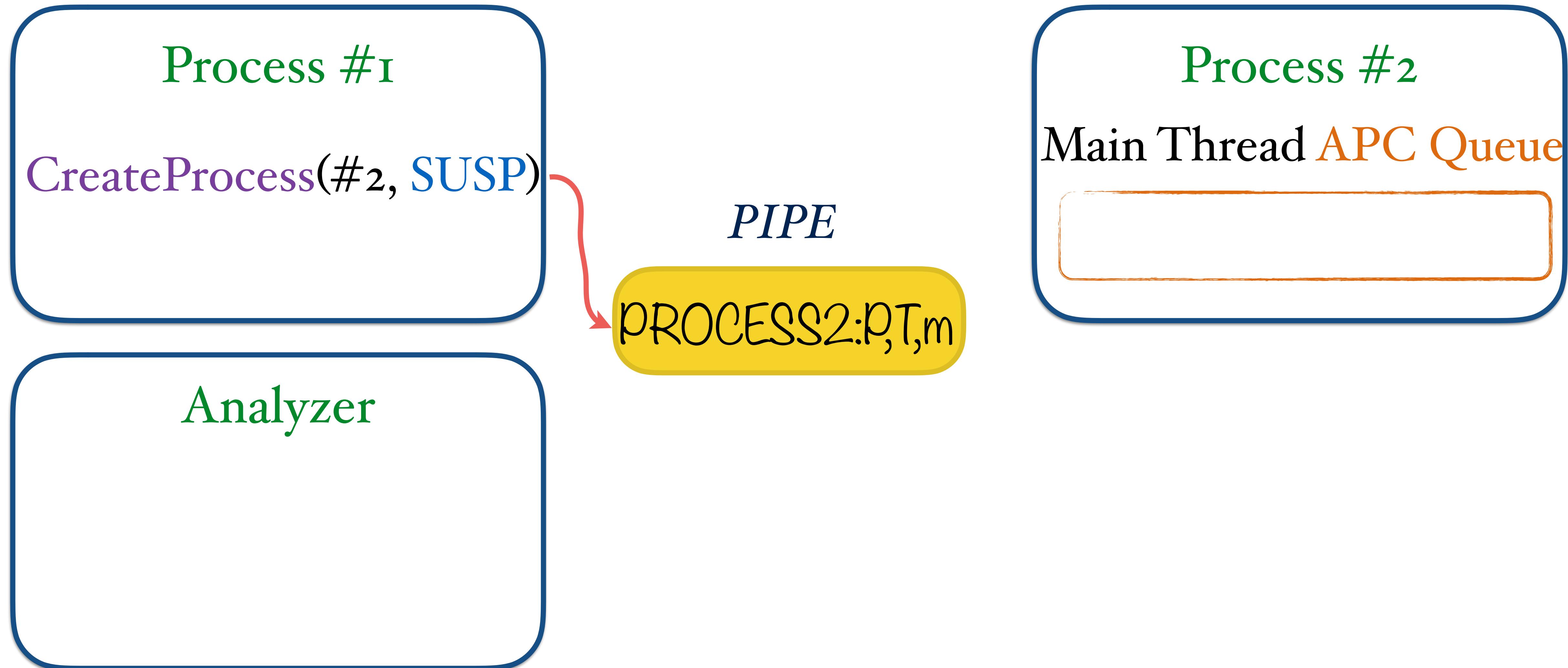
PIPE

Analyzer

Info

Cuckoo: Sandbox

Suspended Thread



Info

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

Analyzer

PIPE

Process #2

Main Thread APC Queue



Info

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

Analyzer

Start inject app
with specific params

PIPE

Process #2

Main Thread APC Queue

Inject



Info

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

Analyzer

Start **inject** app
with specific params

PIPE

Process #2

Main Thread APC Queue

LdrLoadDll

Inject

Queue LdrLoadDll

Cuckoo: Sandbox

Suspended Thread

Windows Internals

- Suspended thread should be resumed to execute queued APC.

Implementation details

- When Monitor is successfully initialized, *LOADED* command is sent via pipe.
- Analyzer does not wait for the notification from the injected Monitor.
- PID of the process is already added to process_list.

So What If The Suspended Thread will never be resumed?

Evasion

Cuckoo: Sandbox

Suspended Thread

Process #1

Evasion

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

Evasion

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)

Process #2

Main Thread (SUSP)

APC Queue

LdrLoadDll

Evasion

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)
CRT(Malicious, RUN)



Process #2

Main Thread (SUSP)

APC Queue

LdrLoadDll

Malicious Thread (RUN)

Evasion

Cuckoo: Sandbox

Suspended Thread

Process #1

CreateProcess(#2, SUSP)
CRT(Malicious, RUN)

Belief that Monitor was injected,
so lack of repeated injections =>
Process #2 is not monitored.

Process #2

Main Thread (SUSP)
APC Queue

LdrLoadDll

Malicious Thread (RUN)

Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

PIPE

Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

`_handle_cmd(data):`

PIPE

Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

_handle_cmd(data):

PIPE

\$cmd:\$data

Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

`_handle_cmd(data):`

PIPE



Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

`_handle_cmd(data):`
`cmd_action(data)`

PIPE

Info

Cuckoo: Sandbox

Analyzer Module: Commands Handling

Analyzer

PipeHandler

```
_handle_cmd(data):  
cmd_action(data)
```

PIPE

So What Happens If Analyzer Is Dead?

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: **wininit.exe**

who: Lsass
app: **lsass.exe**

who: Agent
app: **python.exe**

who: Analyzer
app: **python.exe**

...

who: Explorer
app: **explorer.exe**

who: FireFox
app: **firefox.exe**

who: PyServer
app: **python.exe**

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...
who: Explorer
app: explorer.exe

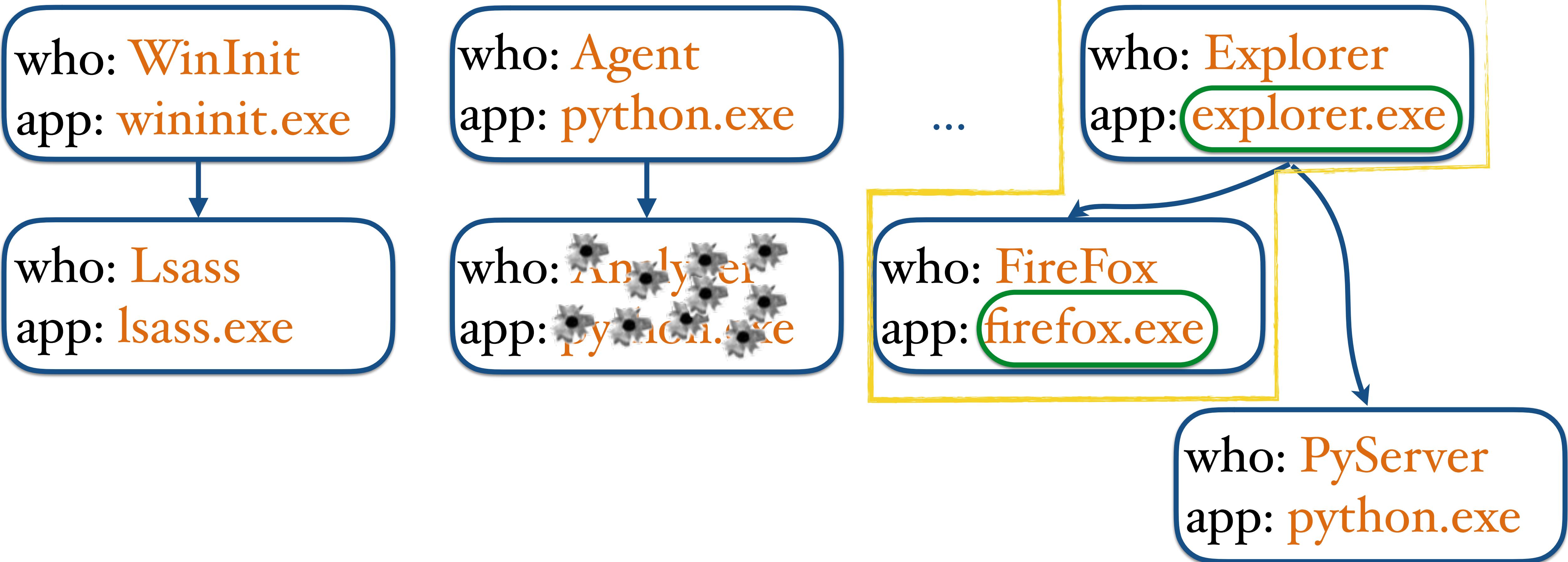
who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer



Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

Cuckoo: Sandbox

Dead Analyzer

who: WinInit
app: wininit.exe

who: Lsass
app: lsass.exe

who: Agent
app: python.exe

who: Analyzer
app: python.exe

...

who: Explorer
app: explorer.exe

who: FireFox
app: firefox.exe

who: PyServer
app: python.exe

Evasion

who: WinInit

app: wininit.exe

who: Lsass

app: lsass.exe

Cuckoo: Sandbox Dead Analyzer

Dead Analyzer,
so Commands Are Not Tracked =>
System Is Not Monitored.

o: Explorer
o: explorer.exe

xe

who: PyServer
app: python.exe

Info

Cuckoo: Monitor

Task Scheduler

- Used for scheduling tasks for the specific time or interval
- Available on all versions of Windows starting from Windows 95
- Task creation is not handled by Cuckoo Monitor at all



Evasion

Cuckoo: Monitor Task Scheduler

Process #1

Tasks Database

Task #1
Task #2
...
Task #n

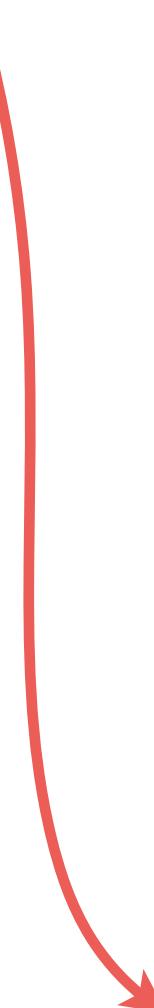
Evasion

Cuckoo: Monitor Task Scheduler

Process #1

```
Task task;  
task = create_task("SE_Task");
```

Tasks Database



```
Task #1  
Task #2  
...  
Task #n  
Task SE_Task
```

Evasion

Cuckoo: Monitor

Task Scheduler

Process #1

```
Task task;  
task = create_task("SE_Task");  
task.start_task(NOW);
```

Process #2 ("SE_Task")

```
perform_malicious_activity();
```

Tasks Database

Task #1
Task #2
...
Task #n
Task SE_Task

Process created as Task object is not monitored.

Info

Cuckoo: Monitor

Whitelisted Processes: Initialization

```
BOOL APIENTRY DllMain(HANDLE hModule, DWORD dwReason, LPVOID lpReserved) {
    (void) hModule; (void) lpReserved;

    if(dwReason == DLL_PROCESS_ATTACH && is_ignored_process() == 0) {
        monitor_init(hModule);
        monitor_hook(NULL, NULL);
        pipe("LOADED:%d,%d", get_current_process_id(), g_monitor_track);
    }

    return TRUE;
}
```

Cuckoo: Monitor

Whitelisted Processes: Initialization (Cont.)

```
static const wchar_t *g_ignored_processpaths[] = {
L"C:\\WINDOWS\\system32\\dwwin.exe",
L"C:\\WINDOWS\\system32\\dumprep.exe",
L"C:\\WINDOWS\\system32\\drwtsn32.exe",
NULL,
};

int is_ignored_process() {
    wchar_t process_path[MAX_PATH];
    GetModuleFileNameW(NULL, process_path, MAX_PATH);
    GetLongPathNameW(process_path, process_path, MAX_PATH);
    for (uint32_t idx = 0; g_ignored_processpaths[idx] != NULL; idx++)
        if(!wcsicmp(g_ignored_processpaths[idx], process_path))
            return 1;
    return 0;
}
```



Process #1

Cuckoo: Monitor Whitelisted Processes



Cuckoo: Monitor Whitelisted Processes

Process #1

```
HANDLE hP, hT;  
wchar_t *pn;  
pn = ChooseProcNameFromWhiteList();  
hP, hT = CreateProcess(pn, SUSPENDED);
```

Process #2 (Whitelisted Name)



Cuckoo: Monitor

Whitelisted Processes

Process #1

```
HANDLE hP, hT;  
wchar_t *pn;  
pn = ChooseProcNameFromWhiteList();  
hP, hT = CreateProcess(pn, SUSPENDED);
```

```
InjectMaliciousCode(hP);  
ResumeThread(hT);
```

Process #2 (Whitelisted Name)

perform_malicious_activity();



Cuckoo: Monitor Whitelisted Processes

Process #1

```
HANDLE hP, hT;  
wchar_t *pn;  
pn = ChooseProcNameFromWhiteList();  
hP, hT = CreateProcess(pn, SUSPENDED);  
  
InjectMaliciousCode(hP);  
ResumeThread(hT);
```

A diagram illustrating a process flow. On the left, under "Process #1", there is C++ code. An arrow originates from the assignment of the variable "pn" in this code and points to "Process #2 (Whitelisted Name)" on the right. Another arrow originates from the call to "perform_malicious_activity()" in "Process #2" and points back to the "ResumeThread(hT);" line in "Process #1".

Whitelisted Process is not monitored.

Proposed Solution

- Remove all processes' paths from the whitelist.

Info

Cuckoo: Monitor

Exceptions Number: Handler's Code

```
#define EXCEPTION_MAXCOUNT 1024 /* before Aug11 commit */

void log_exception(CONTEXT *ctx, EXCEPTION_RECORD *rec,
                    uintptr_t *return_addresses, uint32_t count, uint32_t flags) {

    static int exception_count;
    if(exception_count++ == EXCEPTION_MAXCOUNT) {
        our_snprintf(buf, sizeof(buf), "Encountered %d exceptions, quitting.",
                     exception_count);
        log_anomaly("exception", NULL, buf);
        ExitProcess(1);
    }
}
```

Info

Cuckoo: Monitor

Exceptions Number

```
void RtlDispatchExceptionHook(EXCEPTION_RECORD *er, CONTEXT *c) {
    // code goes here

    if (exception_code == STATUS_ACCESS_VIOLATION &&
        is_exception_address_whitelisted(pc)) {
        // Ignore several exception codes such as the one caused by calling
        // OutputDebugString()
    }
    else if (is_exception_address_whitelisted(pc) == 0) {
        uintptr_t addrs[RETADDRCNT]; uint32_t count = 0;
        log_exception(c, er, addrs, count, 0);
    }
}

// code goes here
}
```

Detection

Cuckoo: Monitor Exceptions Number

Process #1

Detection

Cuckoo: Monitor Exceptions Number

Process #1

```
HANDLE hP;  
DWORD ec;  
hP = CreateProcess("Process #2")
```

Process #2

Detection

Cuckoo: Monitor Exceptions Number

Process #1

```
HANDLE hP;  
DWORD ec;  
hP = CreateProcess("Process #2")
```

Process #2

```
_try {  
    int i;  
    for (i=0;i<=EXCEPTION_MAXCOUNT; ++i) {  
        RaiseException(EXC+i,0,0,NULL);  
    }  
} __except(EXCEPTION_EXECUTE_HANDLER) {}  
ExitProcess(0);
```

Detection

Process #1

```
HANDLE hP;  
DWORD ec;  
hP = CreateProcess("Process #2")
```

```
GetExitCodeProcess(hP, &ec);  
if (ec == 1)  
    halt("Cuckoo detected by EXC");
```

Exceptions Number

Process #2

```
_try {  
    int i;  
    for (i=0;i<=EXCEPTION_MAXCOUNT; ++i) {  
        RaiseException(EXC+i, 0, 0, NULL);  
    }  
} __except(EXCEPTION_EXECUTE_HANDLER) {}  
ExitProcess(0);
```

Detection

Process #1

```
HANDLE hP;  
DWORD ec;  
hP = CreateProcess("Process #2")
```

```
GetExitCodeProcess(hP, &ec);  
if (ec == 1)  
    halt("Cuckoo detected by EXC");
```

Cuckoo: Monitor Exceptions Number

Process #2

```
_try {  
    int i;  
    for (i=0;i<=EXCEPTION_MAXCOUNT; ++i) {  
        RaiseException(EXC+i, 0, 0, NULL);  
    }  
} __except(EXCEPTION_EXECUTE_HANDLER) {}  
ExitProcess(0);
```

Process #2 has exited with specific code after MAXCOUNT exceptions.

Cuckoo: Monitor

Configuration Artifacts

Responsibilities

- The configuration file is responsible for the configuration of the injected Cuckoo Monitor module into tracked process.

Implementation details

- The location is well-known

```
static wchar_t filepath[MAX_PATH_W];
wsprintfW(filepath, L"C:\\cuckoo_%d.ini", pid);

if(MoveFileW(config_file, filepath) == FALSE) {
    error("[-] Error dropping configuration file: %ld\\n", GetLastError());
}
```

Cuckoo: Monitor Configuration Artifacts

Detection Technique

1. Create some process in **CREATE_SUSPENDED** state.
2. As we have the Process ID of the created process, check the **C:** drive for the presence of the **“cuckoo_%d.ini”% pid** file for a few times with delay.
3. If such a file is present, then assume that we are running inside Cuckoo Sandbox environment.

Cuckoo: Monitor

PID Tracking: Default Timeout

```
# If the process monitor is enabled we start checking whether  
# the monitored processes are still alive.
```

```
if pid_check:  
    for pid in self.process_list.pids:  
        if not Process(pid=pid).is_alive():  
            log.info("Process with pid %s has terminated", pid)  
            self.process_list.remove_pid(pid)
```

```
static wchar_t filepath[MAX_PATH_W];  
wsprintfW(filepath, L"C:\\cuckoo_%d.ini", pid);  
  
if (MoveFileW(config_file, filepath) == FALSE) {  
    error("[-] Error dropping configuration file: %ld\\n", GetLastError());  
}
```

Cuckoo: Monitor

PID Tracking: Default Timeout

```
# If the pi  
# the monit  
if pid_chec  
for pid:  
    if not  
        log.  
        self.  
  
BOOL WINAPI MoveFile(  
    _In_  LPCTSTR lpExistingFileName,  
    _In_  LPCTSTR lpNewFileName  
) ;
```

lpNewFileName [in]

The new name for the file or directory.
The new name must not already exist.



Cuckoo: Monitor

PID Tracking: Default Timeout

```
# list of already used pids
pids = []
pid = None

while True:
    pid = create_process(SUSPENDED)
    if pid in pids:
        break
    kill_process(pid)
    pids.append(pid)

if pid is not None:
    print "Cuckoo Sandbox has been evaded."
```

Cuckoo: Monitor

PID Tracking: Default Timeout

```
# list of already used pids
pids = []
pid = None

while True:
    pid = create_process(SUSPENDED)
    if pid in pids:
        break
    kill_process(pid)
    pids.append(pid)

if pid is not None:
    print "Cuckoo Sandbox has been evaded."
```

“C:\cuckoo_%d.ini” % rpid is already present,
so MoveFile function will fail,
thus Monitor will not be injected there =>
Cuckoo Evaded.

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

Process/Monitor

Analyzer

Inject

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

Analyzer

- random name for *config* (ex: *shmemcon*)

Process/Monitor

shmemcon

Inject

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

Process/Monitor

Analyzer

- random name for *config* (ex: *shmemcon*)

shmemcon

config

Inject

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

Analyzer

- random name for *config* (ex: *shmemcon*)

Process/Monitor

shmemcon

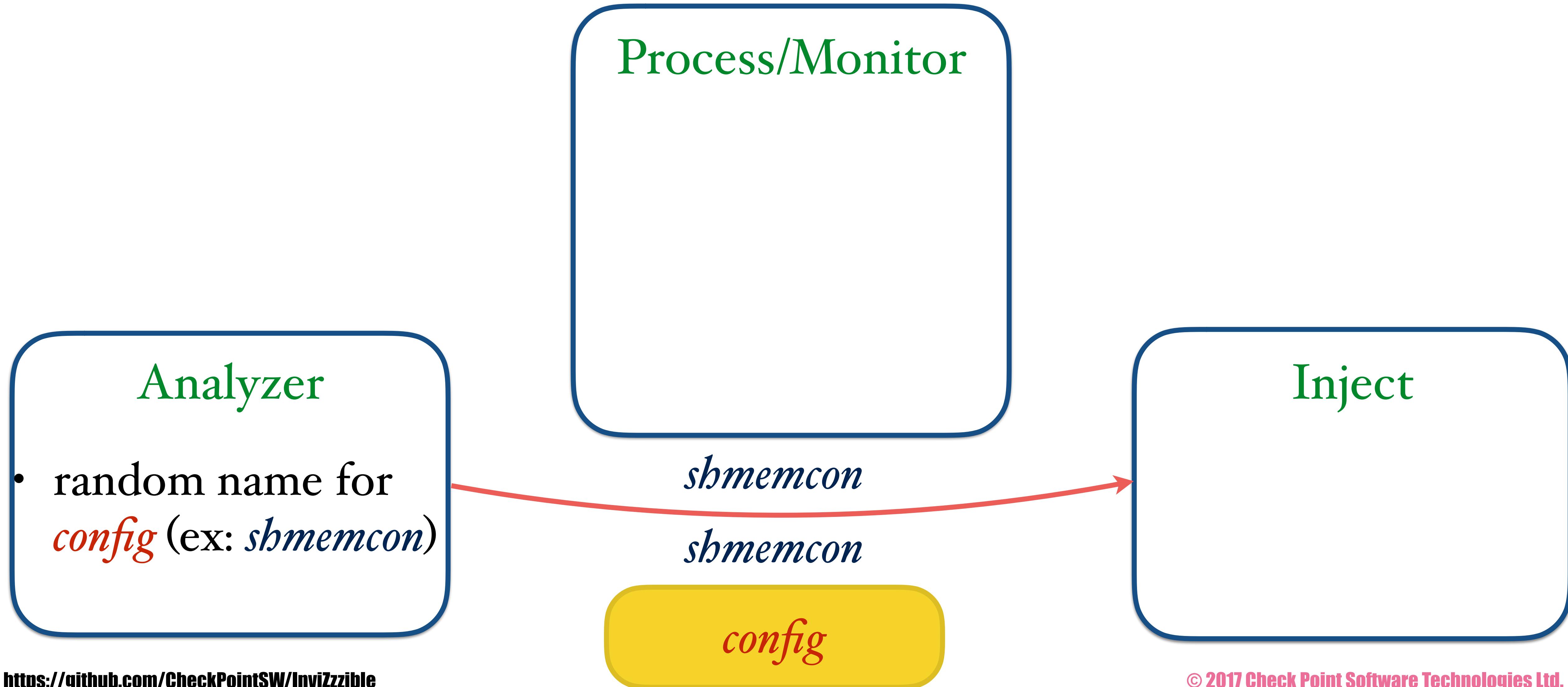
config

Inject

Fix

Cuckoo: Monitor

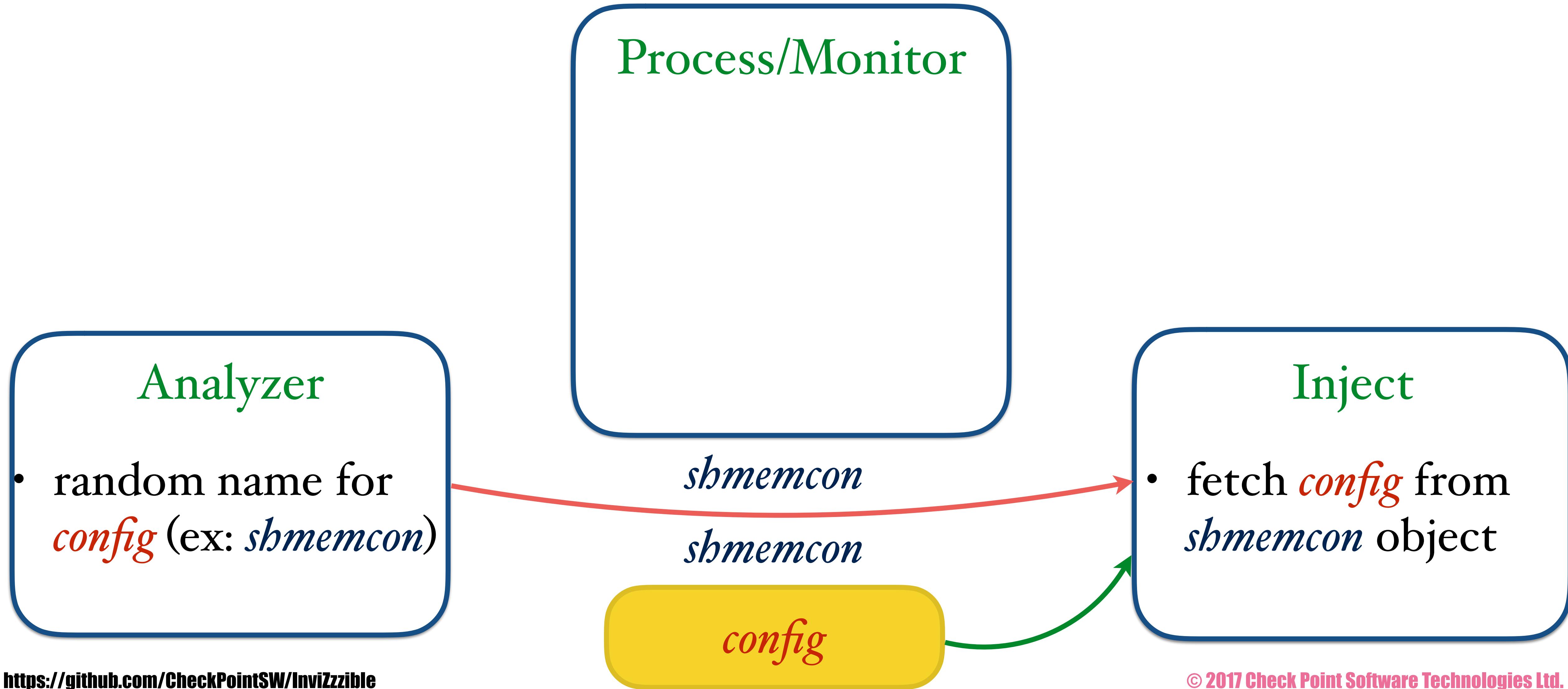
Configuration Artifacts & PID Tracking



Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking



Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

Process/Monitor

Analyzer

- random name for *config* (ex: *shmemcon*)

shmemcon

shmemcon

config

Inject

- fetch *config* from *shmemcon* object

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

config address →

Process/Monitor

Analyzer

- random name for *config* (ex: *shmemcon*)

shmemcon

shmemcon

config

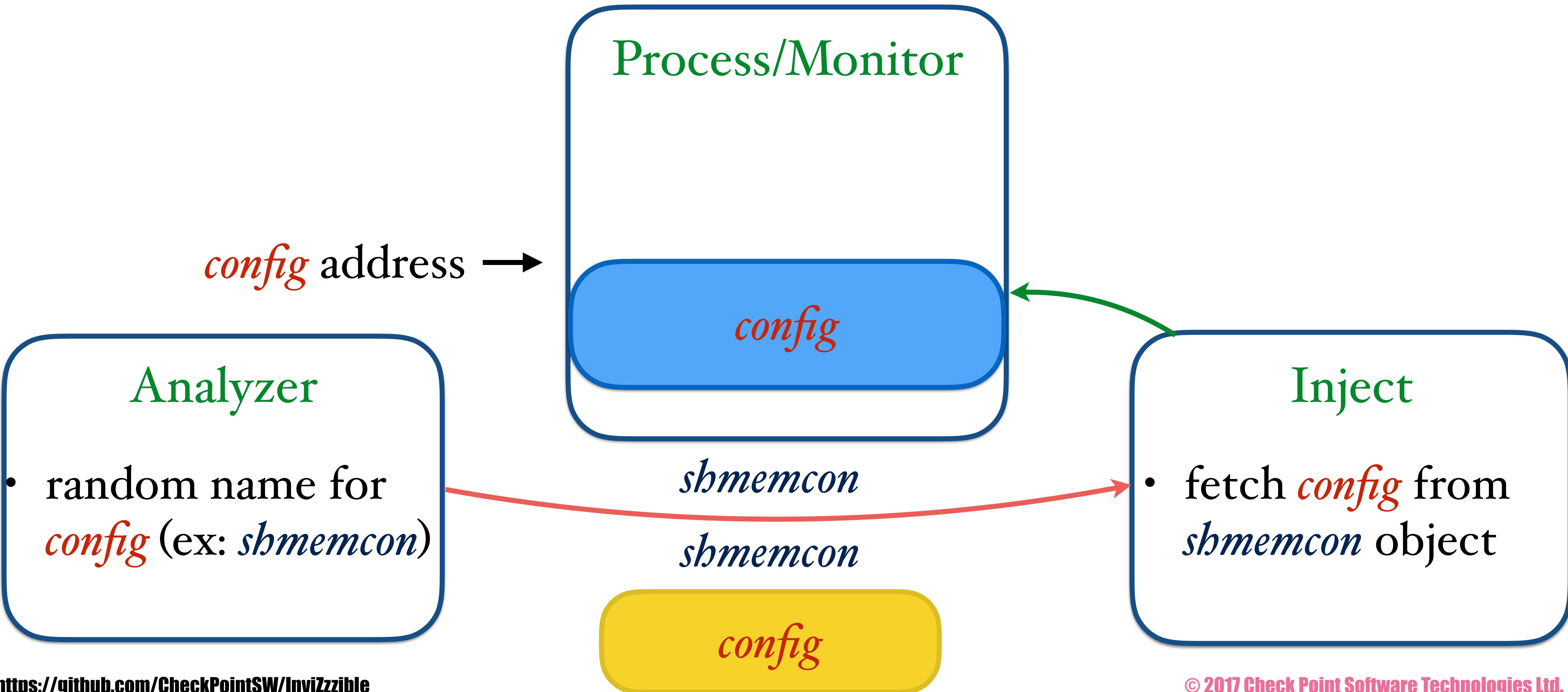
Inject

- fetch *config* from *shmemcon* object

Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking



Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking

config address →

Process/Monitor

config

Analyzer

- random name for *config* (ex: *shmemcon*)

shmemcon

shmemcon

config

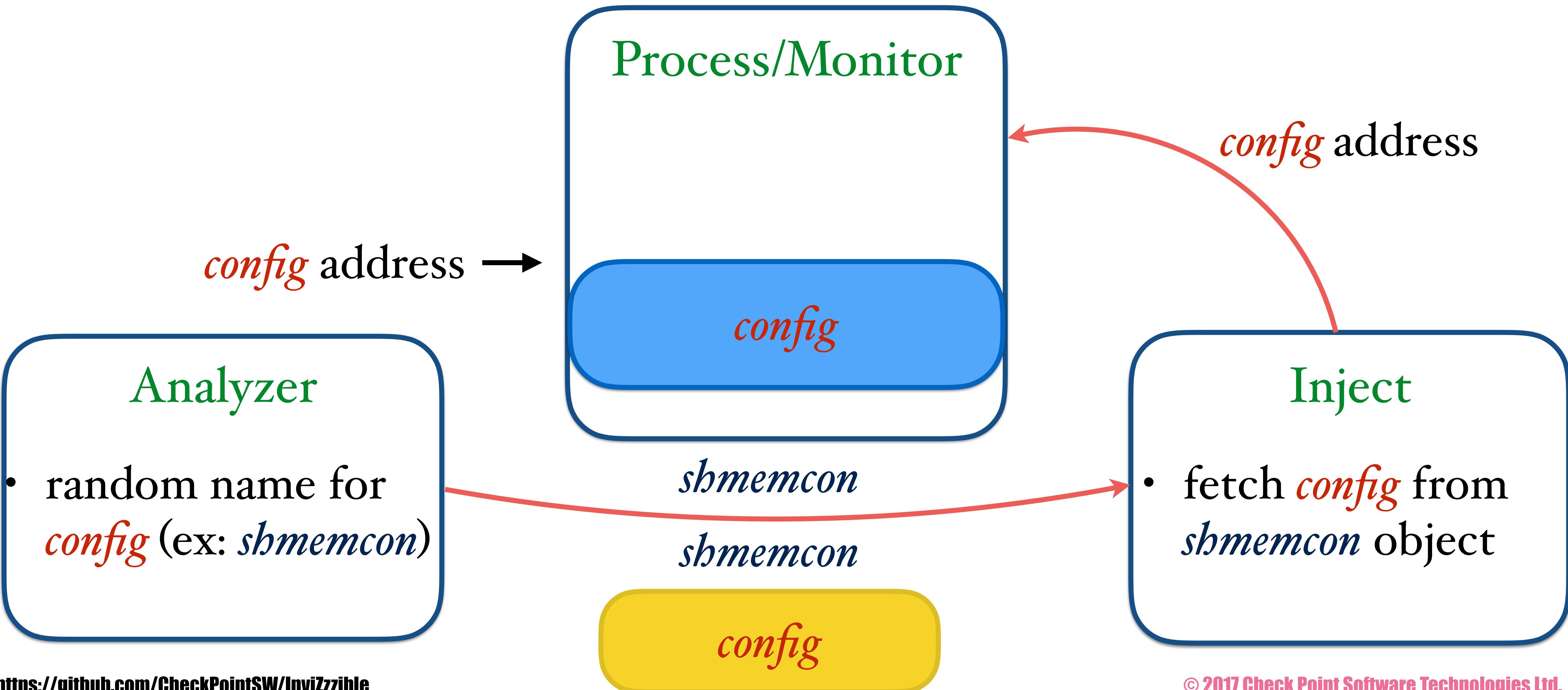
Inject

- fetch *config* from *shmemcon* object

Fix

Cuckoo: Monitor

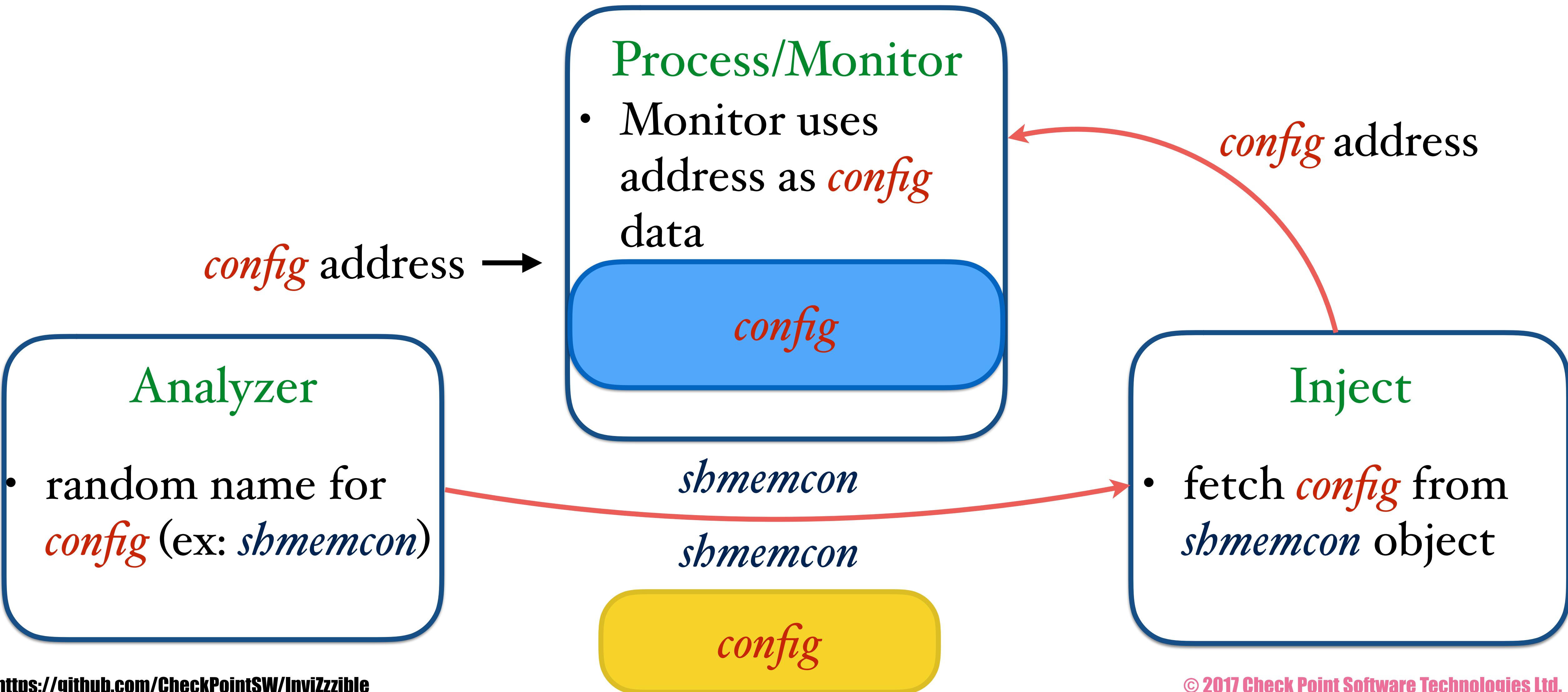
Configuration Artifacts & PID Tracking



Fix

Cuckoo: Monitor

Configuration Artifacts & PID Tracking



Virtual Environment

Virtual Environment

Date/Time Tampering

Description

- In a real environment system, time and web time should be similar with regard to time zone.
- In a virtual environment, sleep functions may be hooked to minimize execution time.
- In a virtual environment, response from web services may be static to avoid access to the external network.
- The described limitations may lead to the possibility of virtual environment detection.

Detection

Virtual Environment

Date/Time Tampering

Process #1

Google

Static Response

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.by/?gfe_rd=cr&ei=Zn09V4uIDemH8Qfv3ZP4Dw
Content-Length: 258
Date: Thu, 19 May 2016 08:46:30 GMT
```

Detection

Virtual Environment

Date/Time Tampering

Process #1

```
GetLocalTime(&lst);  
GetWebTime(&rst);
```

Google

Static Response

```
HTTP/1.1 302 Found  
Cache-Control: private  
Content-Type: text/html; charset=UTF-8  
Location: http://www.google.by/?gfe_rd=cr&ei=Zn09V4uIDemH8Qfv3ZP4Dw  
Content-Length: 258  
Date: Thu, 19 May 2016 08:46:30 GMT
```

Detection

Virtual Environment

Date/Time Tampering

Process #1

```
GetLocalTime(&lst);  
GetWebTime(&rst);  
Sleep(1000*60); //60s
```

Google

Static Response

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.by/?gfe_rd=cr&ei=Zn09V4uIDemH8Qfv3ZP4Dw
Content-Length: 258
Date: Thu, 19 May 2016 08:46:30 GMT

Detection

Virtual Environment

Date/Time Tampering

Process #1

```
GetLocalTime(&lst);  
GetWebTime(&rst);  
Sleep(1000*60); //60s  
GetLocalTime(&lend);  
GetWebTime(&rend);
```

Google

Static Response

```
HTTP/1.1 302 Found  
Cache-Control: private  
Content-Type: text/html; charset=UTF-8  
Location: http://www.google.by/?gfe_rd=cr&ei=Zn09V4uIDemH8Qfv3ZP4Dw  
Content-Length: 258  
Date: Thu, 19 May 2016 08:46:30 GMT
```

Detection

Virtual Environment

Date/Time Tampering

Process #1

```
GetLocalTime(&lst);
GetWebTime(&rst);
Sleep(1000*60); //60s
GetLocalTime(&lend);
GetWebTime(&rend);

tdiff = abs(rend-rst);
ldiff = abs(lend-lst);
if (abs(tdiff-ldiff) > 5*1000)
    halt("VE Detected")
if (tdiff < 60*1000)
    halt("Web Service emulation")
if (ldiff < 60*1000)
    halt("Sleep emulation")
```

Google

Static Response

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.by/?gfe_rd=cr&ei=Zn09V4uIDemH8Qfv3ZP4Dw
Content-Length: 258
Date: Thu, 19 May 2016 08:46:30 GMT

Virtual Environment

Number Of Processors/Cores

Description

- Check if the number of processors/cores is larger than 1.
- The technique is old and well-known.
- The number of cores/processors may be retrieved using:
 - Function [GetSystemInfo](#)
 - Directly [PEB](#)
 - Function [GetNativeSystemInfo](#)

Problems

- The functions described above may be hooked.
- The memory under [PEB](#) may be patched.

Virtual Environment

Number Of Processors/Cores

Thread #1

```
DWORD cn;
unsigned char apic;
set<unsigned char> apic_ids;
GetNumberOfCores(&cn);

for (DWORD i=0;i<cn;++i) {
    SetThreadAffinityMask(Thread_1, i);
    __asm {
        mov eax, 1;
        cpuid;
    }
    apic = ebx[31:24];
    add_to_set(apic_ids, apic);
}

if (apic_ids.size() <= 1)
    halt("VE detected by number of cores")
```

Virtual Environment

Firmware Tables

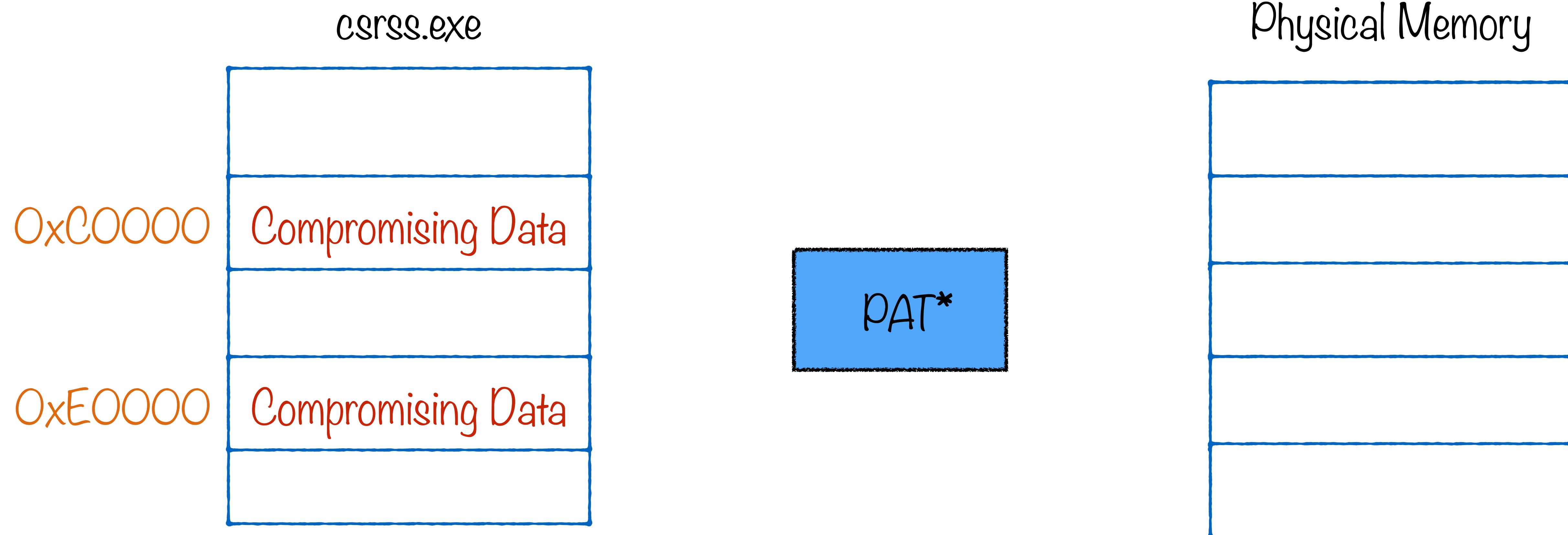
Description

- Technique based on raw and SMBIOS firmware tables content was implemented in VMDE tool.
- A list of detected virtual machines may be found on VMDE source code web page.
- On systems older than Vista systems, content was retrieved from csrss.exe process using [NtReadVirtualMemory](#) system call.
- On Vista and higher systems, content was retrieved using [NtQuerySystemInformation](#) system call.
- There is currently no proposed fix for that detection.

Fix

Virtual Environment

Firmware Tables: XP

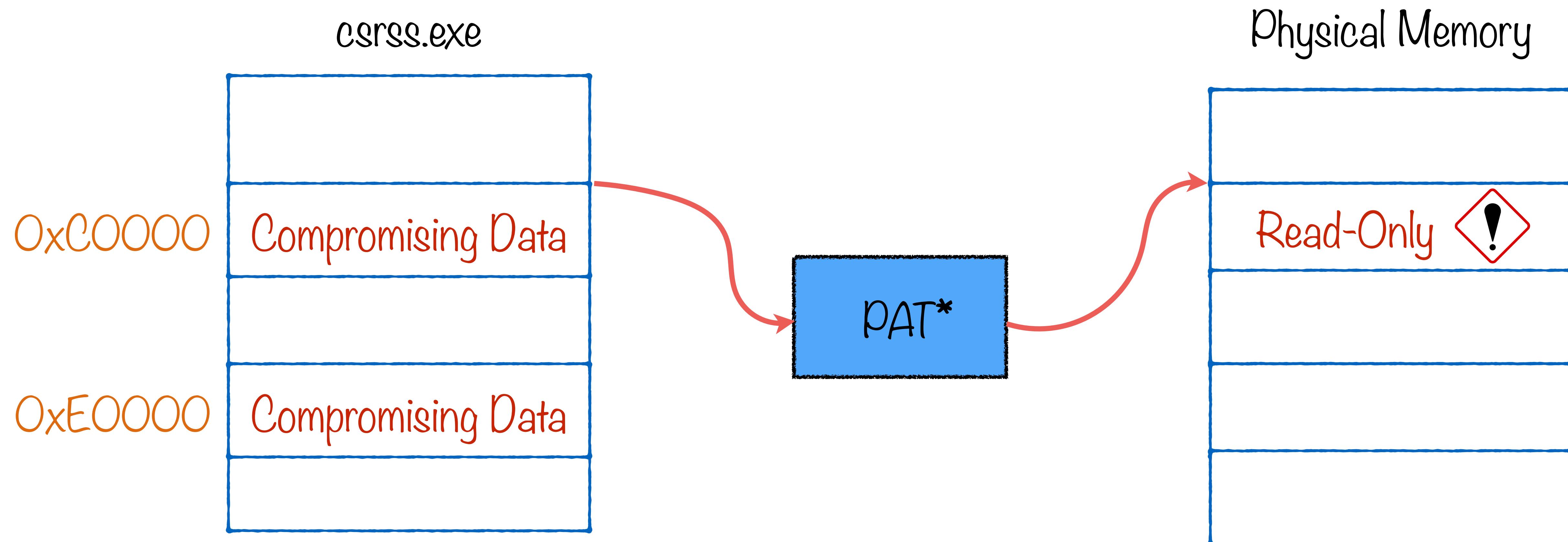


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

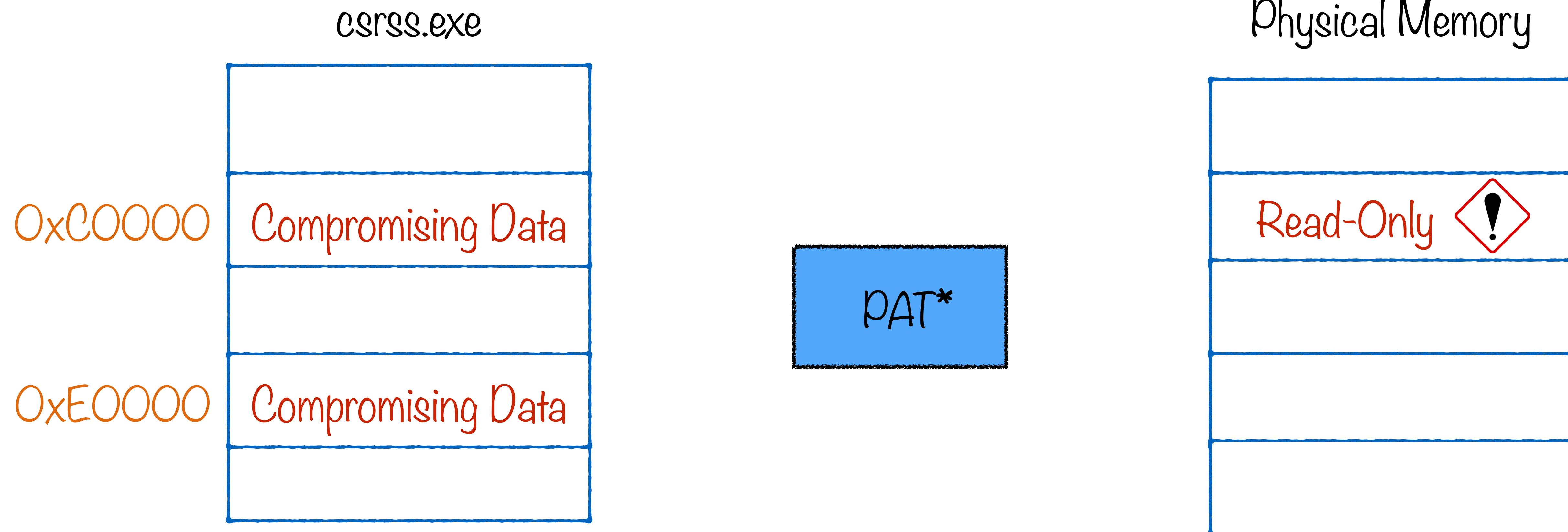


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

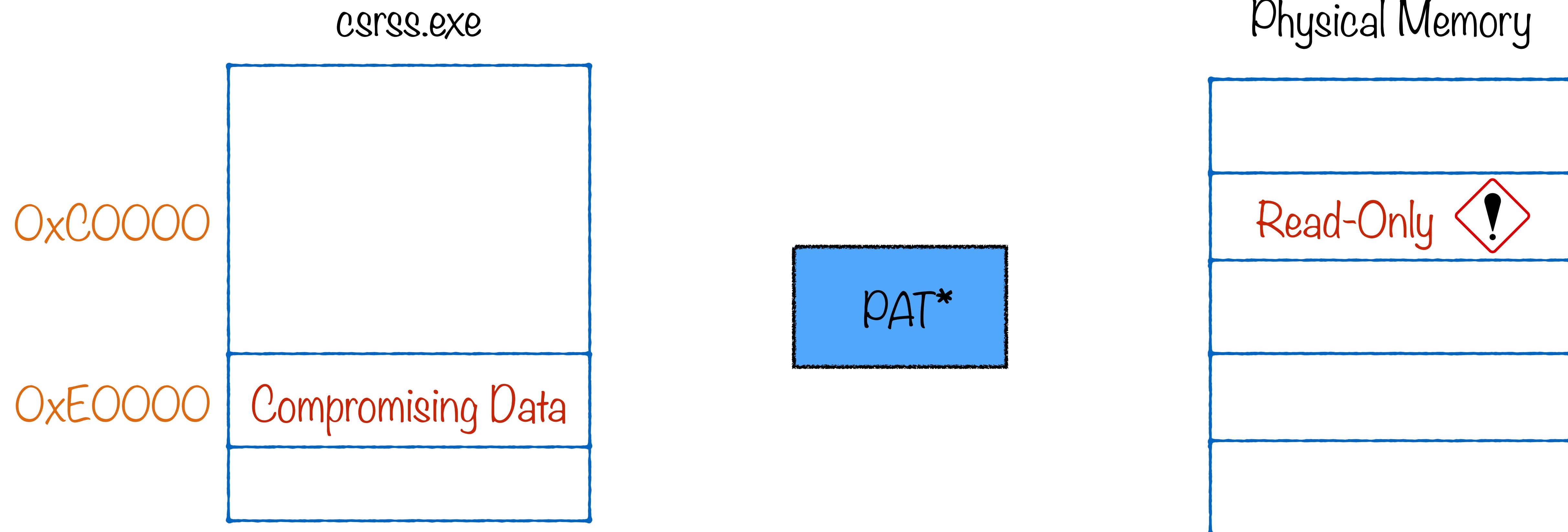


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

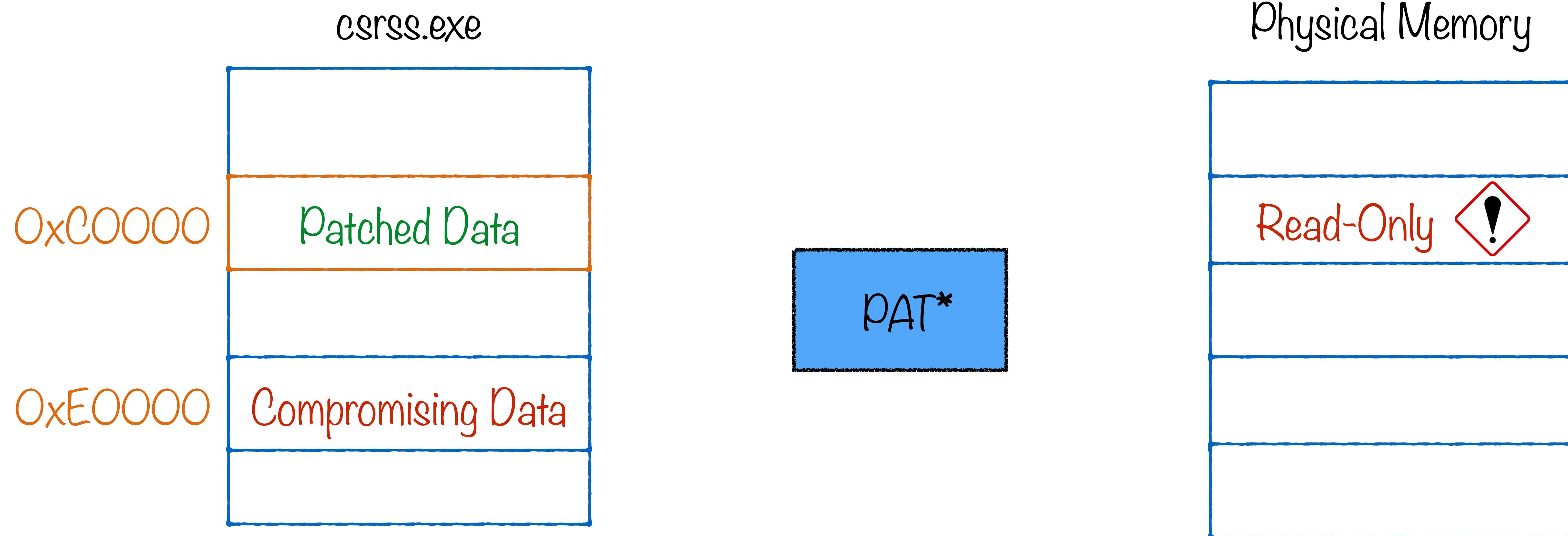


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

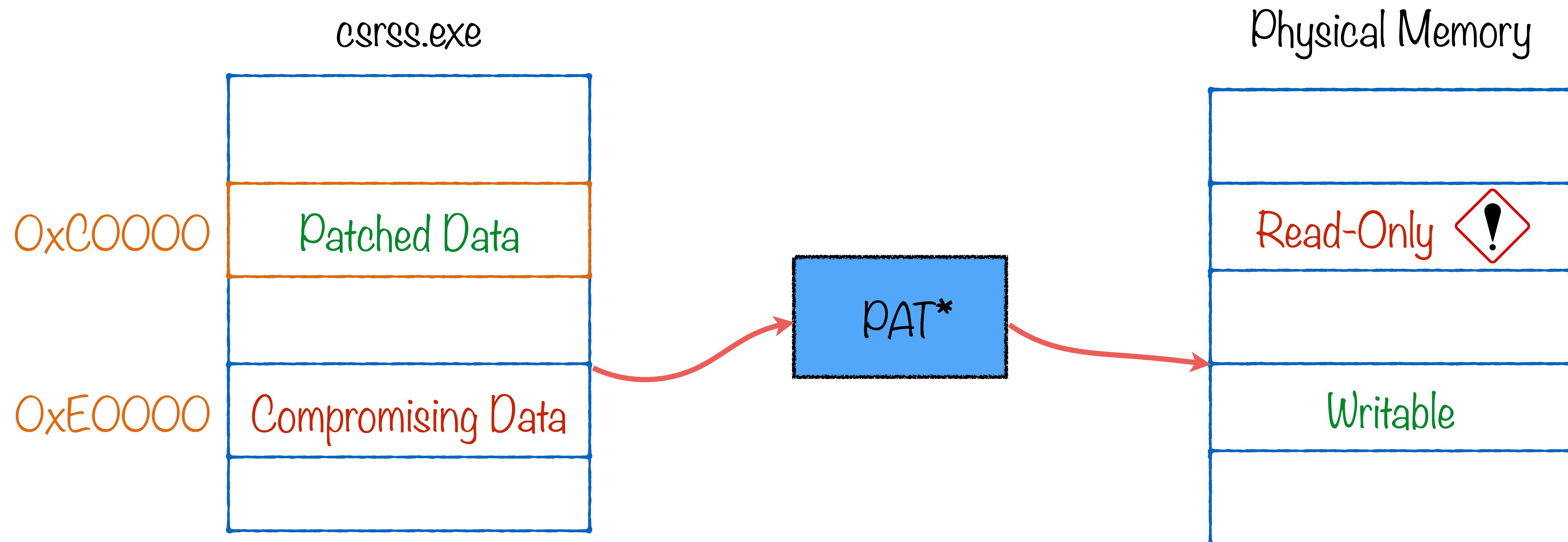


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

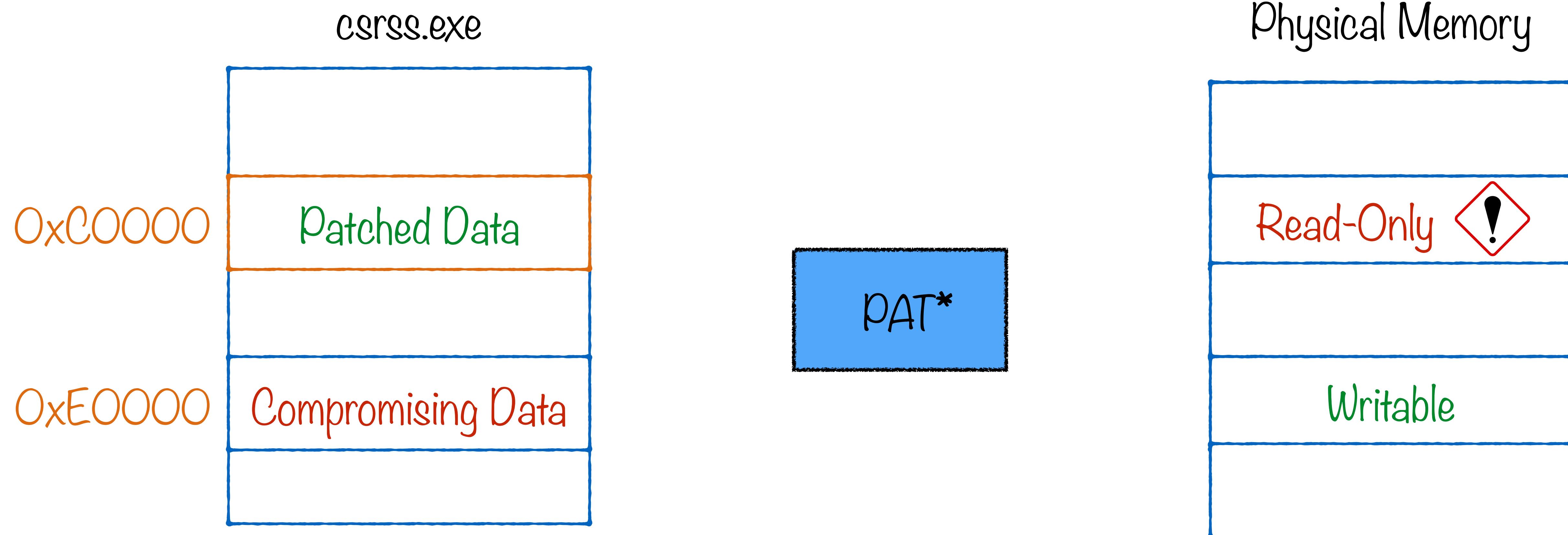


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP

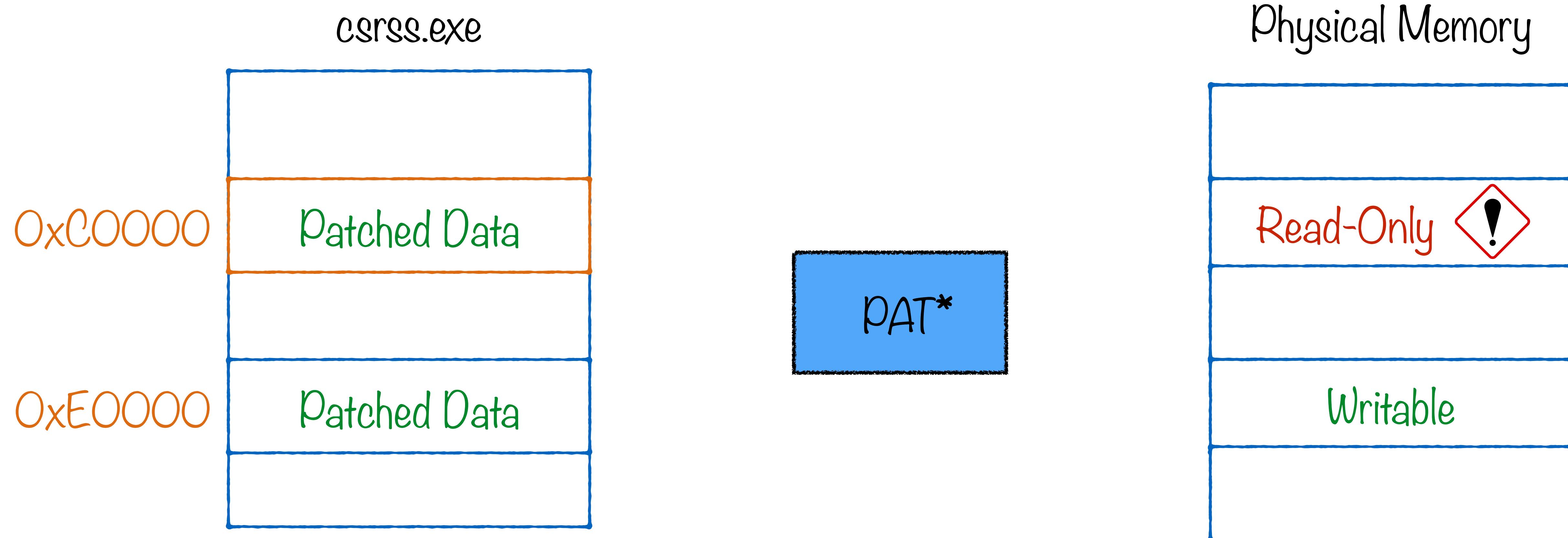


*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: XP



*PAT - Physical Address Translation

Fix

Virtual Environment

Firmware Tables: Vista+

Proposed Solution

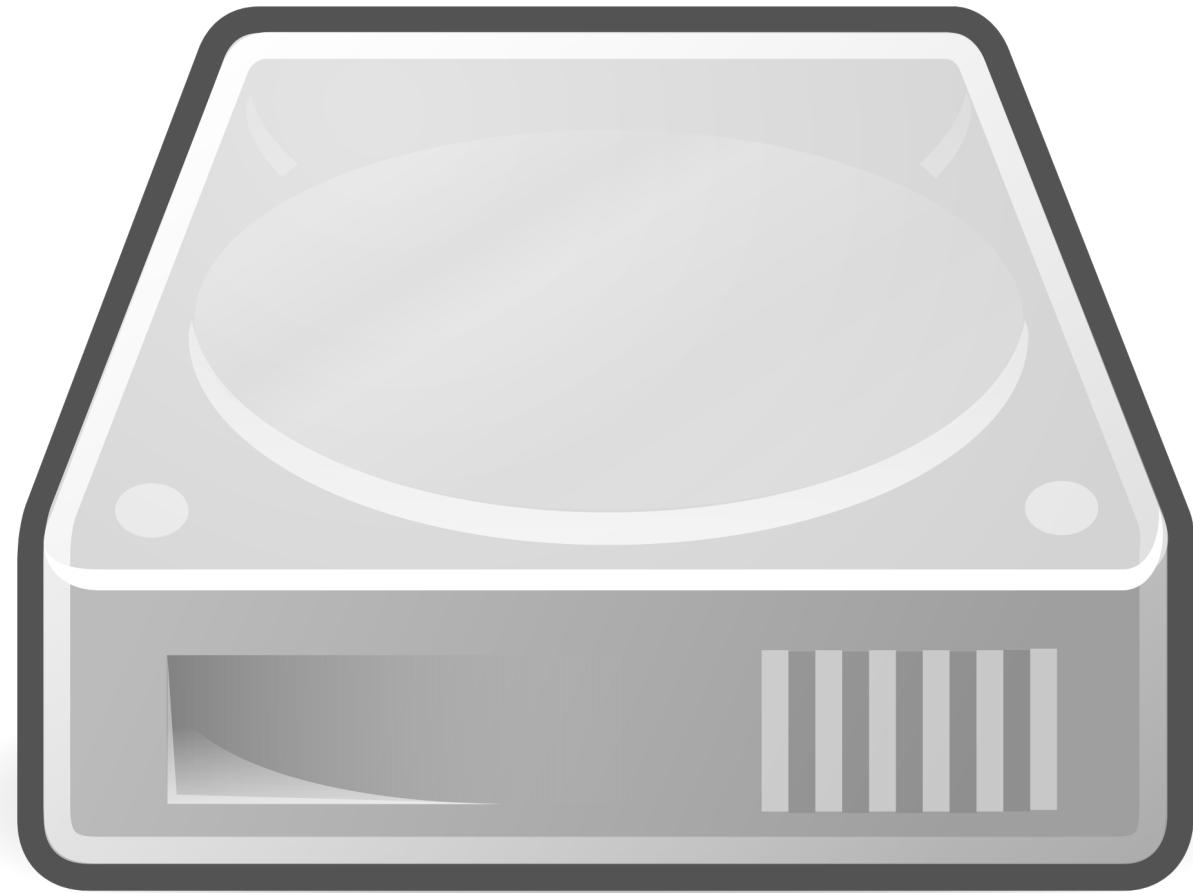
1. Check if `SystemInformationClass` is equal to `SystemFirmwareTableInformation`.
2. Parse the SFTI structure for `ProviderSignature` and `TableId`.
3. Check if the `ProviderSignature` is ‘`FIRM`’ or ‘`RSMB`’.
4. Call the original `NtQuerySystemInformation` routine.
5. Modify the buffer that is returned to user space.

Detection

Virtual Environment

Drive Info

\.\PhysicalDriveX

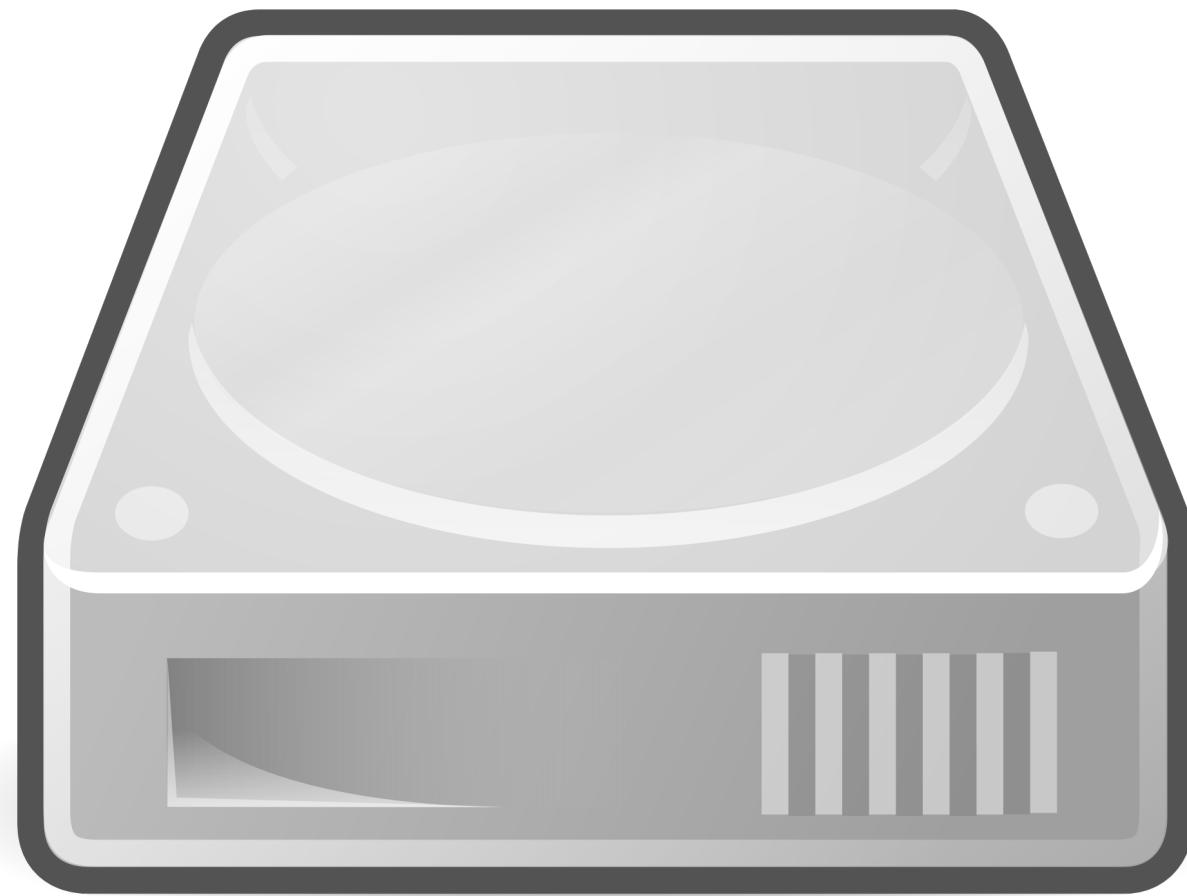


Detection

Virtual Environment

Drive Info

\.\PhysicalDriveX



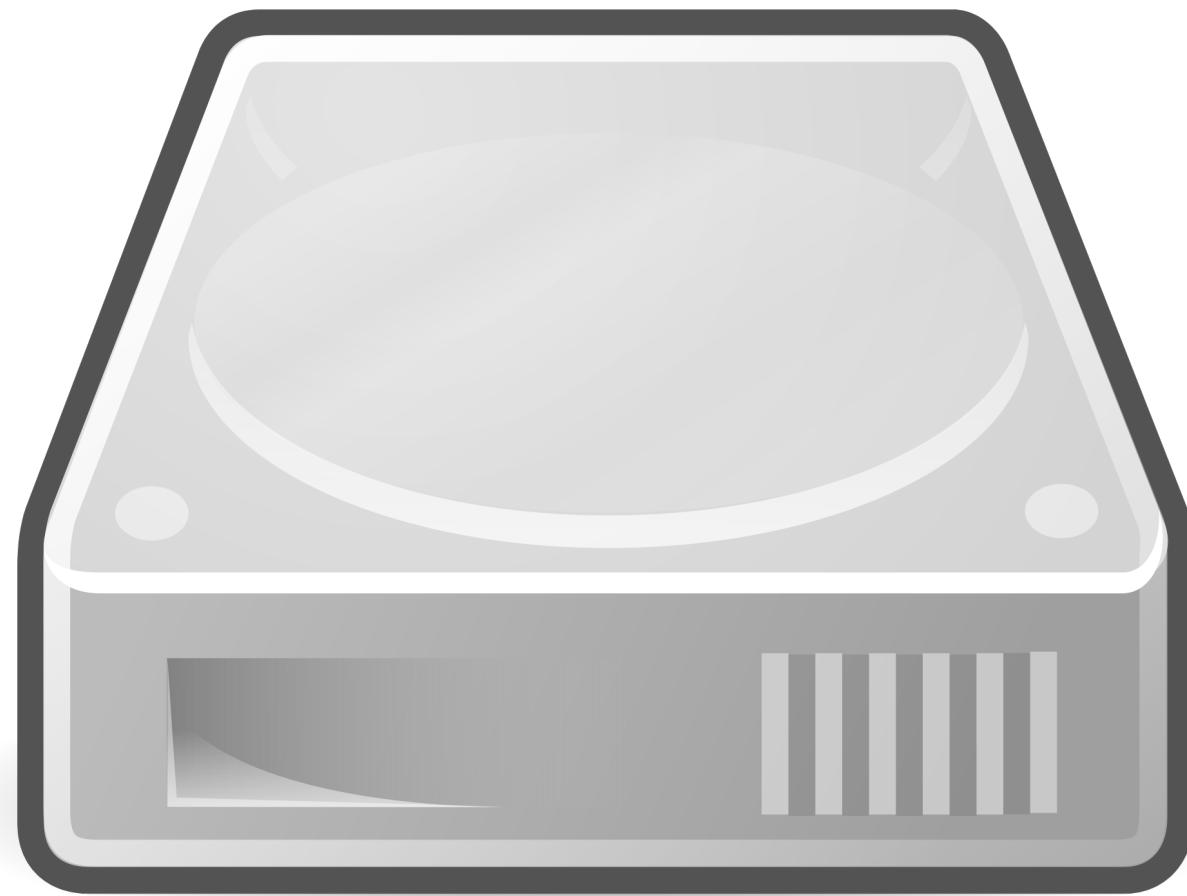
IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)

Detection

Virtual Environment

Drive Info

\.\PhysicalDriveX



IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)



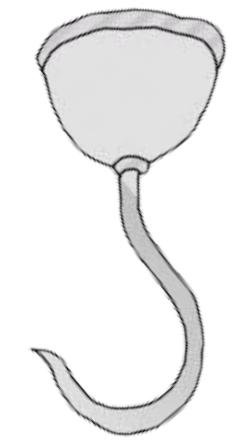
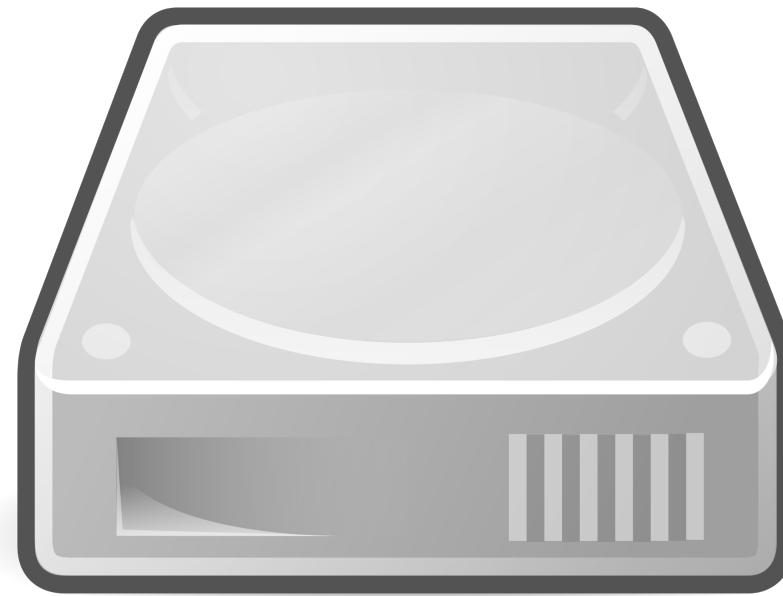
DriveModelNumber
DriveSerialNumber
DriveControllerRevisionNumber

Fix

Virtual Environment

Drive Info

\.\PhysicalDriveX

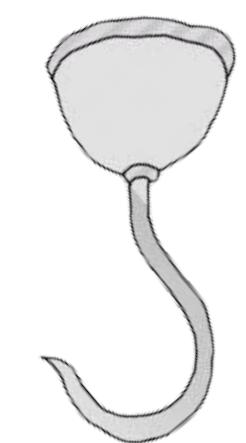
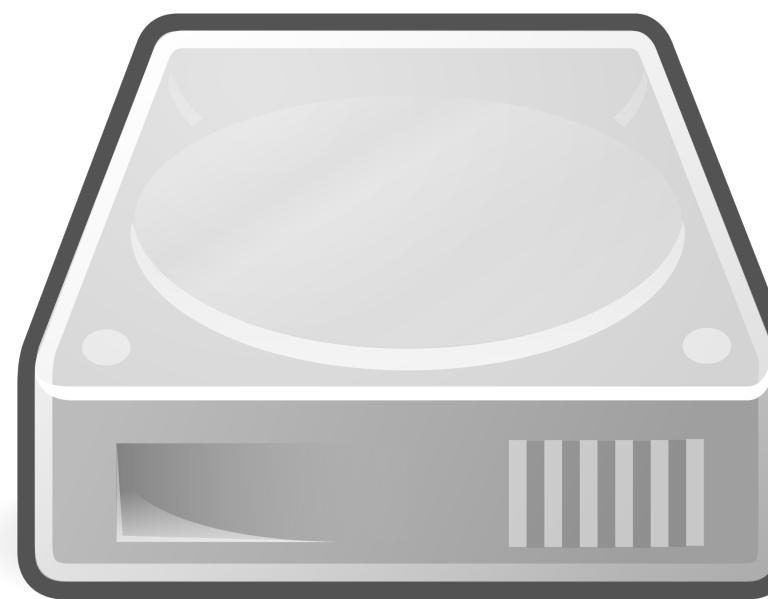


Fix

Virtual Environment

Drive Info

\.\PhysicalDriveX



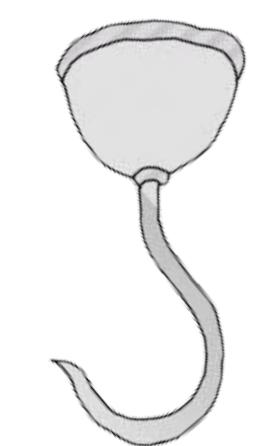
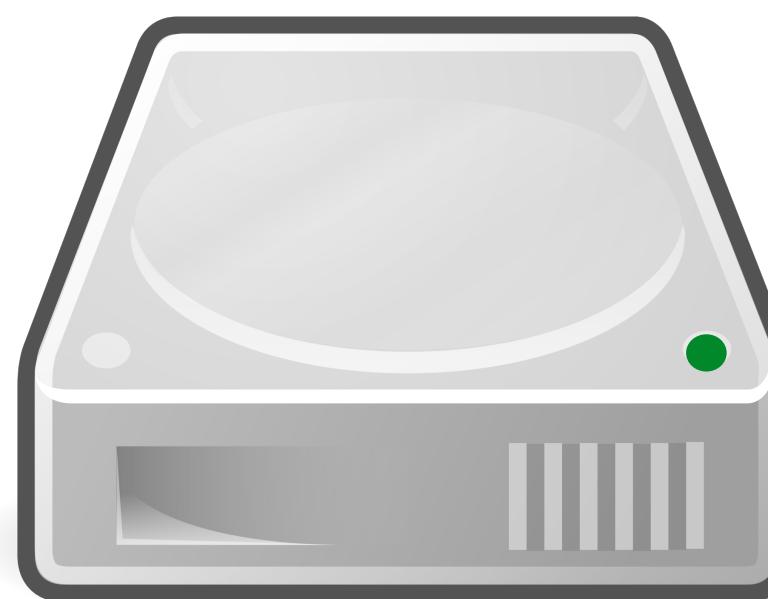
IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)

Fix

Virtual Environment

Drive Info

\.\PhysicalDriveX



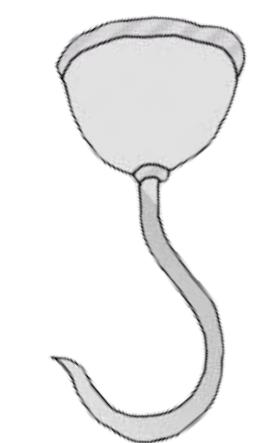
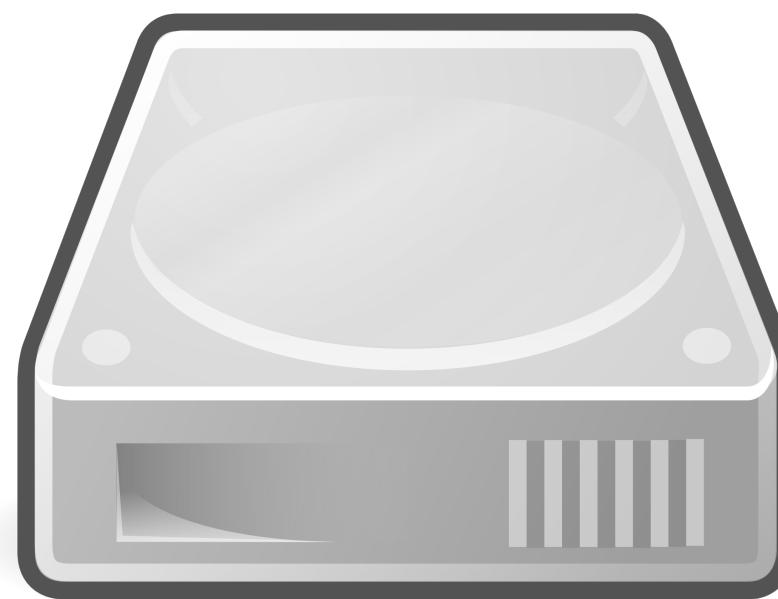
IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)

Fix

Virtual Environment

Drive Info

\.\PhysicalDriveX



IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)

DriveModelNumber

DriveSerialNumber

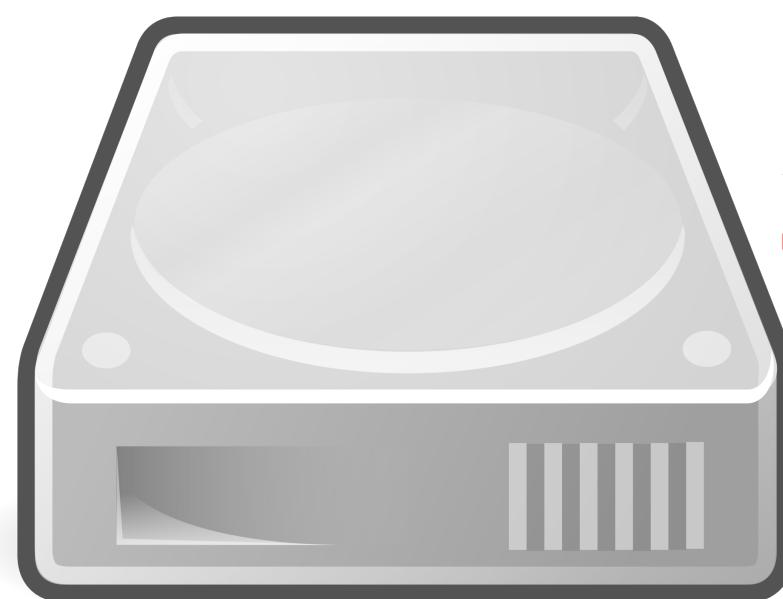
DriveControllerRevisionNumber

Fix

Virtual Environment

Drive Info

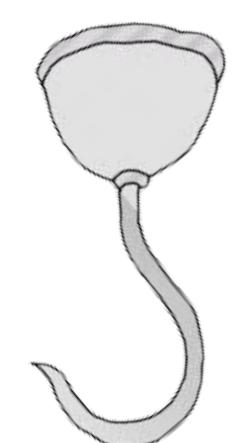
\.\PhysicalDriveX



DriveModelNumber

DriveSerialNumber

DriveControllerRevisionNumber



IOCTL_QUERY_STORAGE_PROPERTY
DFP_RECEIVE_DRIVE_DATA
(SMART_RCV_DRIVE_DATA)

PatchedDriveModelNumber

PatchedDriveSerialNumber

PatchedDriveControllerRevisionNumber

InviZzzible



InviZzzible

Idea Behind the Tool Creation

- Generic tool that covers many different virtual environment detection techniques
- Contains information how to fix positive detections
- Easy-extendable interface support for new virtual environments
- Cuckoo Sandbox detection/evasion techniques support
- Fully configurable tool that may be used for the internal virtual environments tests
- Support of many different common detection techniques, such as registry keys, devices, files presence, etc. through JSON configuration files
- User-friendly report about the checked environment

InviZzzible

Supported Environments

- Cuckoo Sandbox
- Virtual Box
- VMWare
- Generic

InviZzzible

Supported Generic Detection Types

- Registry keys
- Devices
- Files
- Processes
- MAC addresses
- Network adapters
- Disk names
- Drive Models
- Firmwares
- System objects content
- Processor vendors
- Windows
- Shared folders

InviZzzible

Configuration File Content (Registry Key)

```
"ControlSet001 Enum": {  
    "description": "Check if ControlSet001\\Enum subkeys have  
specific value",  
    "countermeasures": "Countermeasures",  
    "type": "registry",  
    "enabled": "yes",  
    "arguments": {  
        "check": "contains",  
        "recursive": "yes",  
        "hkey": "HKLM",  
        "key": "SYSTEM\\ControlSet001\\Enum",  
        "value_name": [ "DeviceDesc", "FriendlyName" ],  
        "value_data": "VMware"  
    }  
}
```

InviZzzible

Adding New Detection (Object)

- To add a new detection, we need to add a new entry in configuration file.
- No recompilation for the tool is needed.

```
"Device Object": {  
    "description": "Check if specific device object is present",  
    "countermeasures": "Countermeasures",  
    "type": "object",  
    "enabled": "yes",  
    "arguments": {  
        "directory" : "\\\Device",  
        "name": "vmmemctl"  
    }  
}
```

InviZzzible

Output Interface: Report

- User-friendly HTML generated file.
- Contains detection technique description.
- Contains information on how to fix environment for specific detection technique.

CUCKOO

Detection Name	Type	Description	Detected	Countermeasures
UnbalancedStack	custom	Check if canaries at the top of the stack remains the same after function call.	NO	Stack adjusting before function call. Kernel-mode hooking.
DelaysAccumulation	custom	Check if delays accumulation is valid using get time functions.	YES	Complex.
InfiniteDelay	custom	Check if INFINITE delay is skipped due to sleep skipping.	NO	Adding conditional check for INFINITE delay.

InviZzzible

Output Interface: Console

- Console mode.
- Contains additional debug information.

```
*****
*****SandboxEvasion*****
*****
[INFO] MAIN: Initialize virtual environment detection modules...
[INFO] CUCKOO: Starting checks...
[INFO] CUCKOO: UnbalancedStack> Check if canaries at the top of the stack remains the same after function call.: 0

[INFO] CUCKOO: DelaysAccumulation> Check if delays accumulation is valid using get time functions.: 0

[INFO] CUCKOO: InfiniteDelay> Check if INFINITE delay is skipped due to sleep skipping.: 0

[INFO] CUCKOO: FunctionHooks> Check if functions are hooked.: 0

[INFO] CUCKOO: AgentArtifacts> Check if agent artifacts are present in the system.: 0

[INFO] CUCKOO: CuckoomonConfiguration> Check if cuckoomon configuration files are present in the system.: 0

[INFO] CUCKOO: WhitelistedProcess> Check if whitelisted process is not tracked (Attention: for the non-sandbox environment it will be always true)

[INFO] CUCKOO: EventName> Check if specific event name is present in the system (Relevant for CuckooMon).: 0

[INFO] CUCKOO: RaisedExceptions> Check if after raising specified number of exceptions, process exits with specific error code.: 0
```

InviZzzible

Output Interface: Domains

- Domain is generated per each detection with `.detected` or `.notdetected` TLD.
- Used for checking environments, where we do not have full access to the machine.
- Information about each detection technique may be found in Behavioral Analysis (Network Operations => DNS).

Domains

- CUCKOO.UnbalancedStack.notdetected
- CUCKOO.AgentArtifacts.notdetected
- CUCKOO.DelaysAccumulation.detected
- CUCKOO.FunctionHooks.detected

InviZzzible

Output Interface: File

- File is generated per each detection with `_detected` or `_notdetected` postfix.
- Used for checking environments, where we do not have full access to the machine.
- Information about each detection technique may be found in Behavioral Analysis (Files Operations) by looking for the `_detected` pattern.

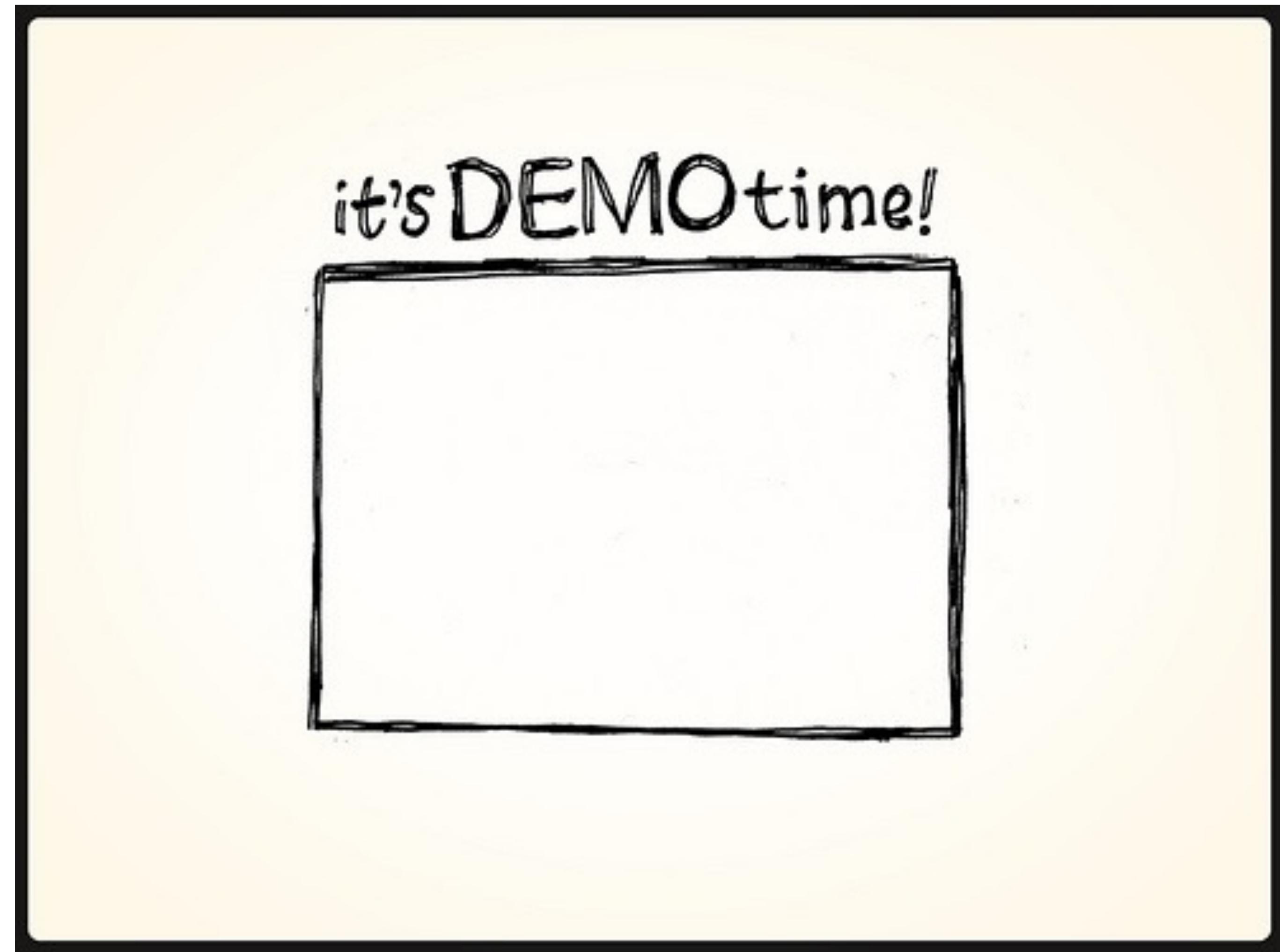
File Names

- `VMWARE_VideoBiosVersionRegKey_notdetected`
- `CUCKOO_TaskSchedulerProcess_detected`
- `VMWARE_Processes_detected`
- `CUCKOO_FunctionHooks_detected`

Summary

- Many new detection/evasion techniques for Cuckoo Sandbox were introduced.
- An easy-extendable tool that supports multiple virtual environments was created:
 - Easy interface for adding new environments
 - Addition of new detection techniques through JSON configurable files
 - Four output interfaces are supported: user-friendly HTML report, console, domains and files

Demo



Thanks And Credits

Aliaksandr Trafimchuk

Alexey Bukhteyev

Q&A

<https://github.com/CheckPointSW/InviZzzible>

Alexander Chaillytko

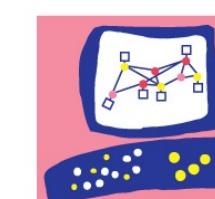
Team Leader

alexanderc@checkpoint.com

Stanislav Skuratovich

Malware Researcher

stanislavsk@checkpoint.com



Check Point®
SOFTWARE TECHNOLOGIES LTD.