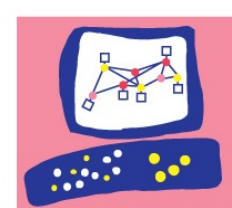


Defeating Sandbox Evasion

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

Sandbox Detection Evasion Tool

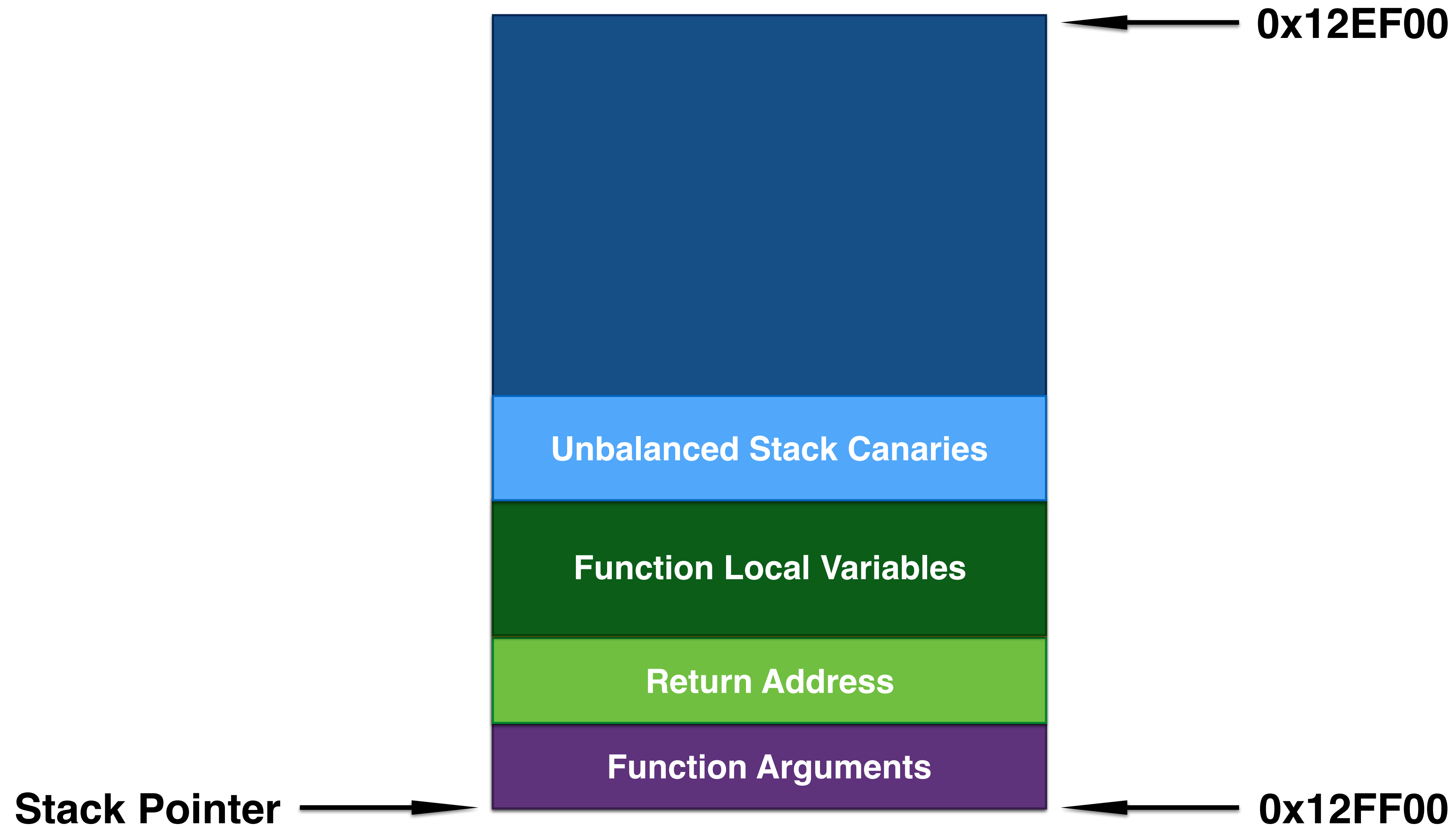
- Contains detection/evasion techniques for different environments
- Configurable through JSON files
- Developed for assessment of internal (your) virtual systems

Cuckoo Sandbox



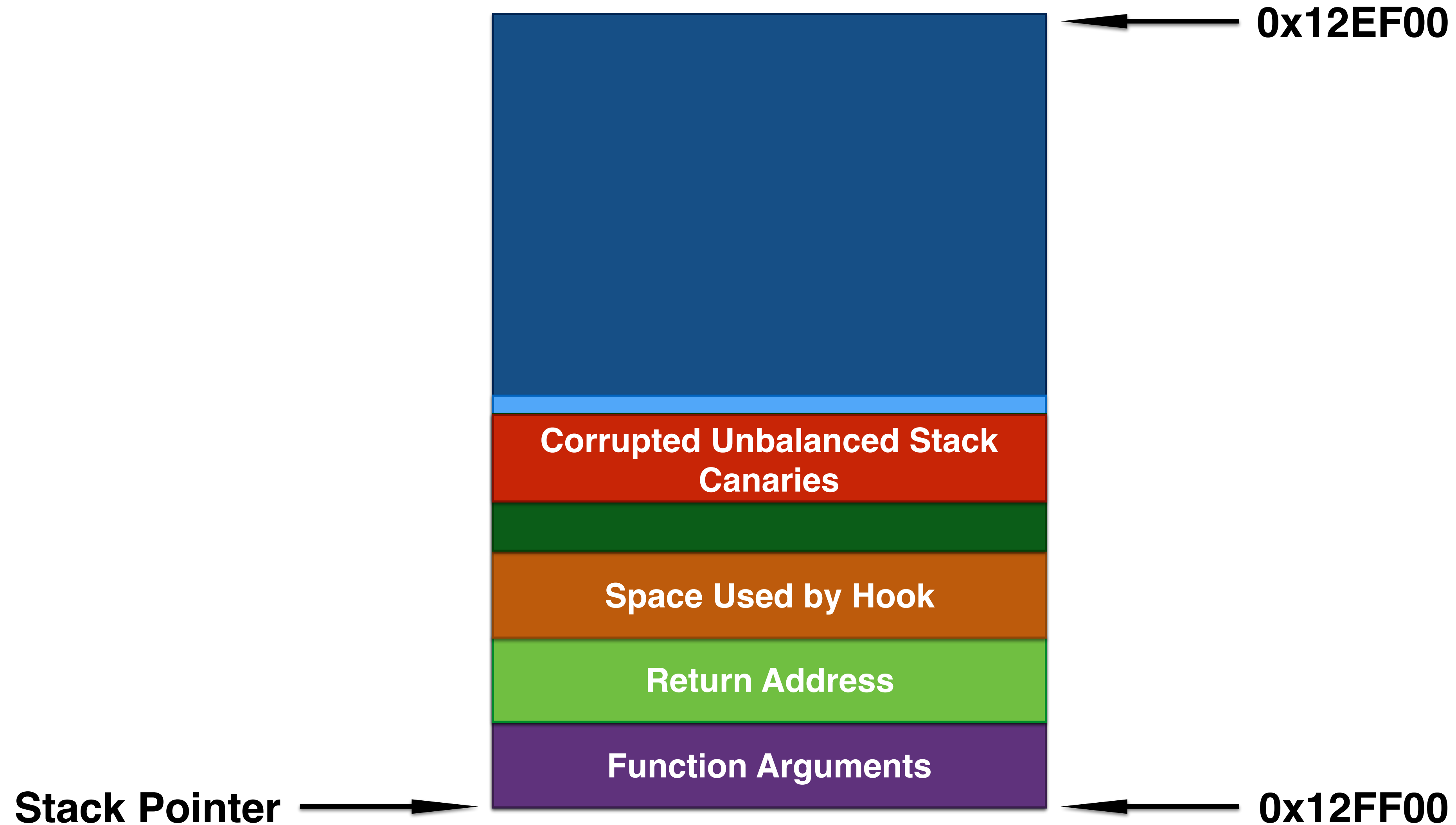
Cuckoo: Monitor

Unbalanced Stack: Normal Operation



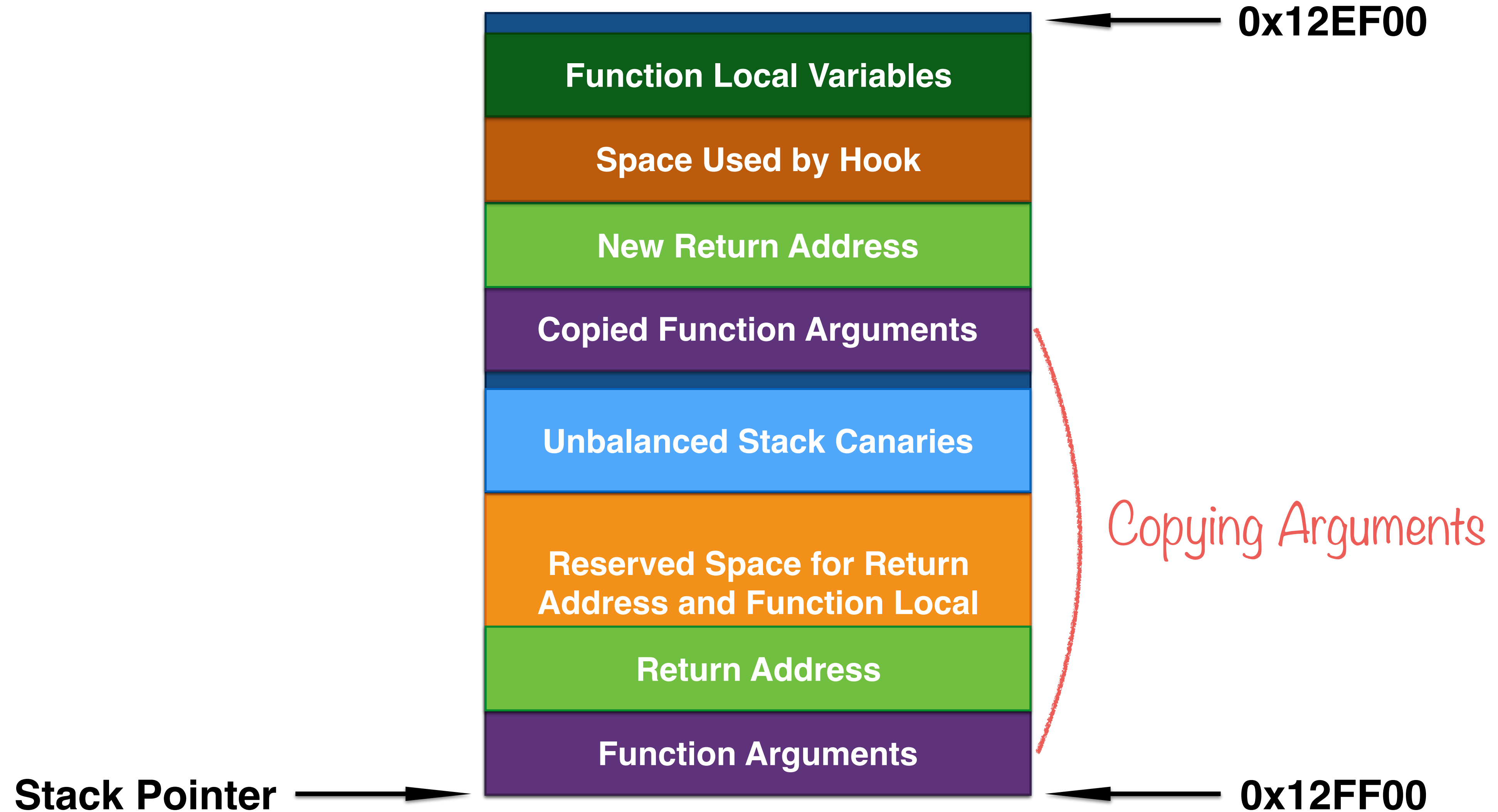
Cuckoo: Monitor

Unbalanced Stack: Problem Definition



Cuckoo: Monitor

Unbalanced Stack: Proposed Solution



Cuckoo: Monitor

Sleep Skip: Code (Main Logic)

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


Cuckoo: Monitor

Sleep Skip: Usage

sleep_skip0

- NtDelayExecution

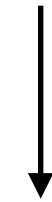
g_time_skipped

- NtQuerySystemTime
- GetTickCount
- GetLocalTime
- GetSystemTime
- GetSystemTimeAsFileTime

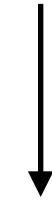
Cuckoo: Monitor

INFINITE Delay: Normal Operation

```
Sleep(INFINITE)
```



```
SleepEx(INFINITE, FALSE)
```



```
NtDelayExecution(FALSE, &DelayInterval)  
DelayInterval = 0x8000000000000000
```

Thread Sleeps Forever.

Cuckoo: Monitor

INFINITE Delay: Problem Definition

```
Sleep(INFINITE)
↓
SleepEx(INFINITE, FALSE)
↓
NtDelayExecutionHook(FALSE, &DelayInterval)
DelayInterval = 0x8000000000000000
↓
sleep_skip(&DelayInterval)
↓
NtDelayExecutionOrig(FALSE, &DelayInterval)
DelayInterval = 0xFFFFFFFFFFFF18FC
```

Thread Sleeps Only for 0.1ms => Cuckoo Detect.

Cuckoo: Monitor

INFINITE Delay: Proposed Solution

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

Cuckoo: Monitor

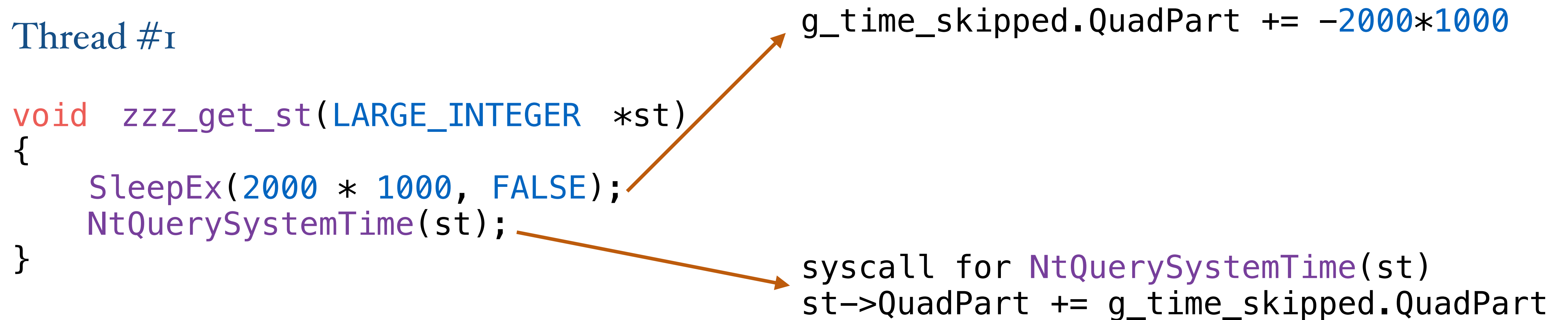
Delays Accumulation: Normal Operation

Thread #I

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

g_time_skipped.QuadPart += -2000*1000

syscall for NtQuerySystemTime(st)
st->QuadPart += g_time_skipped.QuadPart

The diagram illustrates the flow of data in the provided code. Two orange arrows originate from the code. The first arrow starts at the 'SleepEx(2000 * 1000, FALSE);' line and points to the 'g_time_skipped.QuadPart += -2000*1000' line. The second arrow starts at the 'NtQuerySystemTime(st);' line and points to the 'syscall for NtQuerySystemTime(st)' and 'st->QuadPart += g_time_skipped.QuadPart' lines.

Skipped delays are added to current time.

Cuckoo: Monitor

Delays Accumulation: Detection Technique

Thread #1

```
LARGE_INTEGER st_start, st_end;  
NtQuerySystemTime(&st_start);  
WaitForSomeObject(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 += -1000*60*60*24*3
```

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

Cuckoo: Monitor

Delays Skipping: Evasion Technique & Proposed Solution

Evasion Technique

- 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.

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

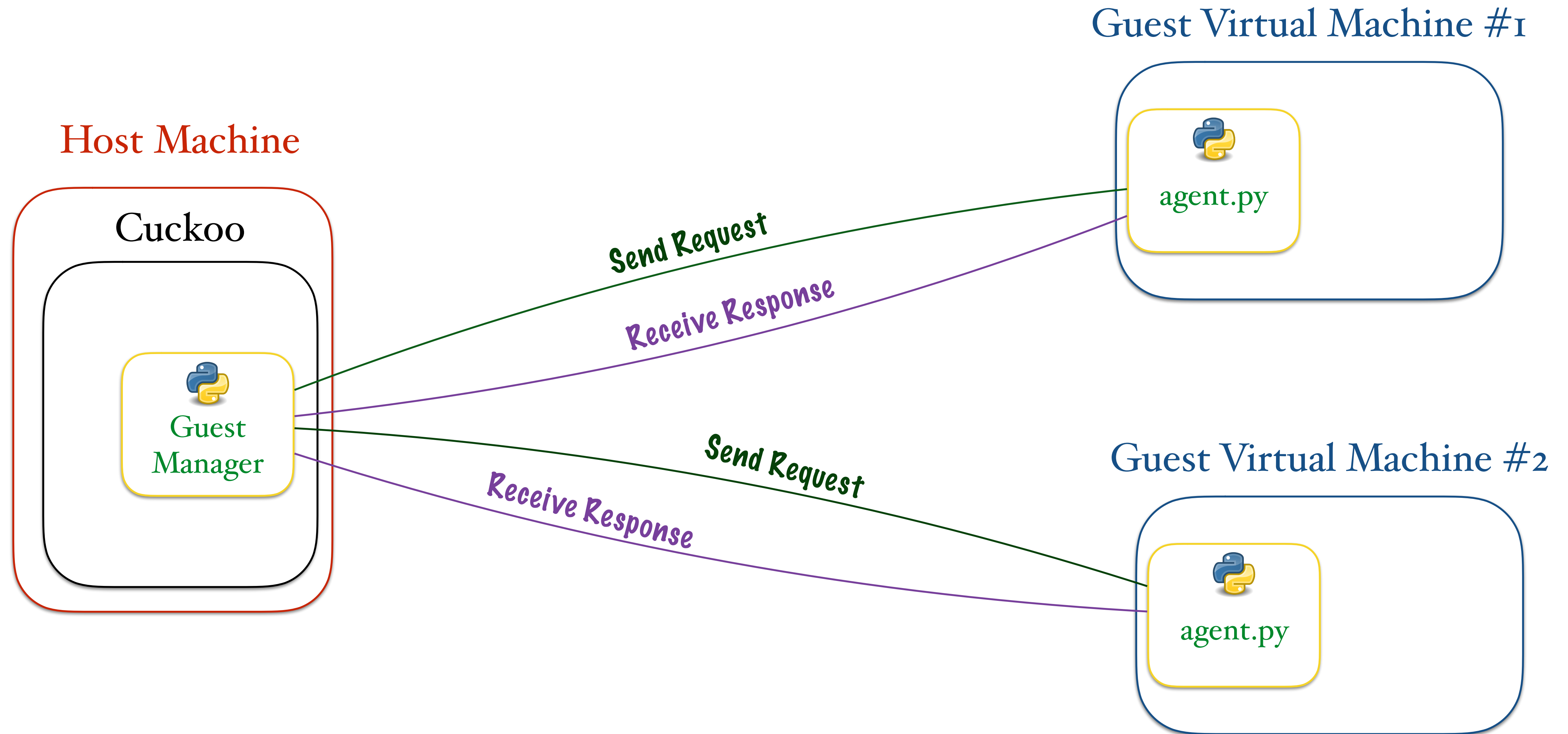
- All large delays are skipped.
- Small delays in limited set are skipped.

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.

Cuckoo: Sandbox

Host Guest: Communication Architecture



Cuckoo: Sandbox

Agent Module: Description

Features

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

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

Detection Technique

1. Enumerate all LISTENING sockets.
2. Send crafted packet and wait for specific 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>
```

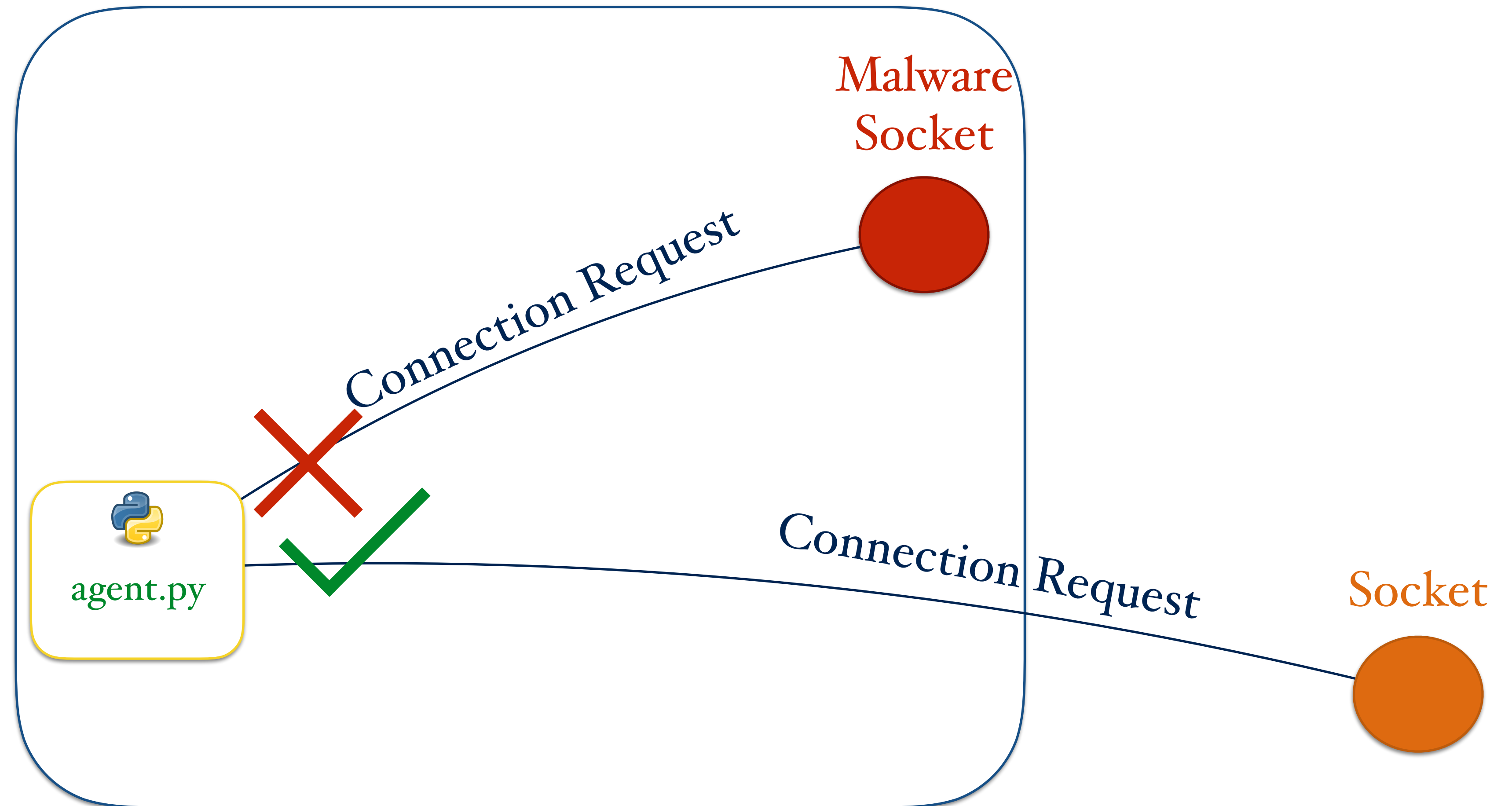
Response Regex

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

Cuckoo: Sandbox

Agent Module: Proposed Solution

Guest Virtual Machine



Cuckoo: Sandbox

Analyzer Module: Description

Features

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

Implementation details

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

Cuckoo: Sandbox

Analyzer Module: Description (Cont.)

```
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 & Proposed Solution

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

- Use randomly generated name for the Analyzer module.
- Use randomly generated name for the configuration file and pass it to the Analyzer module using the argument.

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)
```

Cuckoo: Sandbox

PID Tracking: Evasion Technique

```
# 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."
```

process_list already contains pid, so
Monitor code will not be injected there



Cuckoo: Sandbox

PID Tracking: Proposed Solution

```
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.remove_pid(int(data))
```

Cuckoo: Monitor

Task Scheduler: Description

- Used for the 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

Cuckoo: Monitor

Task Scheduler: Evasion Technique

Process #1

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

```
task.start_task(NOW);
```

Tasks Database

Task #1
Task #2

...
Task #n
Task SE_Task

Process #2 (SE_Task)

```
perform_malicious_activity();
```

Process created as Task objects is not monitored.

Cuckoo: Monitor

Whitelisted Processes: Initialization

```
BOOL WINAPI 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\\dumpprep.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;
}
```

Cuckoo: Monitor

Whitelisted Processes: Evasion Technique & Proposed Solution

Process #1

```
HANDLE hP, hT;
```

```
wchar_t pn;
```

```
pn = choose_from_whtl_list();
```

```
hP, hT = CreateProcess(pn, SUSPENDED);
```

```
InjectMaliciousCode(hP);
```

```
ResumeThread(hT);
```

Process #2 (Whitelisted Name)

```
perform_malicious_activity();
```

Whitelisted Process is not monitored.

Proposed Solution

- Remove all processes' paths from the whitelist.

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);
    }
}
```

Cuckoo: Monitor

Exceptions Number: Problem Definition

RtlDispatchException Hook

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

Cuckoo: Monitor

Exceptions Number: Detection Technique

Process #1

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

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

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: Description

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

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.

Virtual Environment

Virtual Environment

Date/Time Tampering: Description

Description

- In real environment system, time and web time should be similar with regard of time zone.
- In virtual environment, sleep functions may be hooked to minimize execution time.
- In 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.

Virtual Environment

Date/Time Tampering: Detection Technique (Static Response)

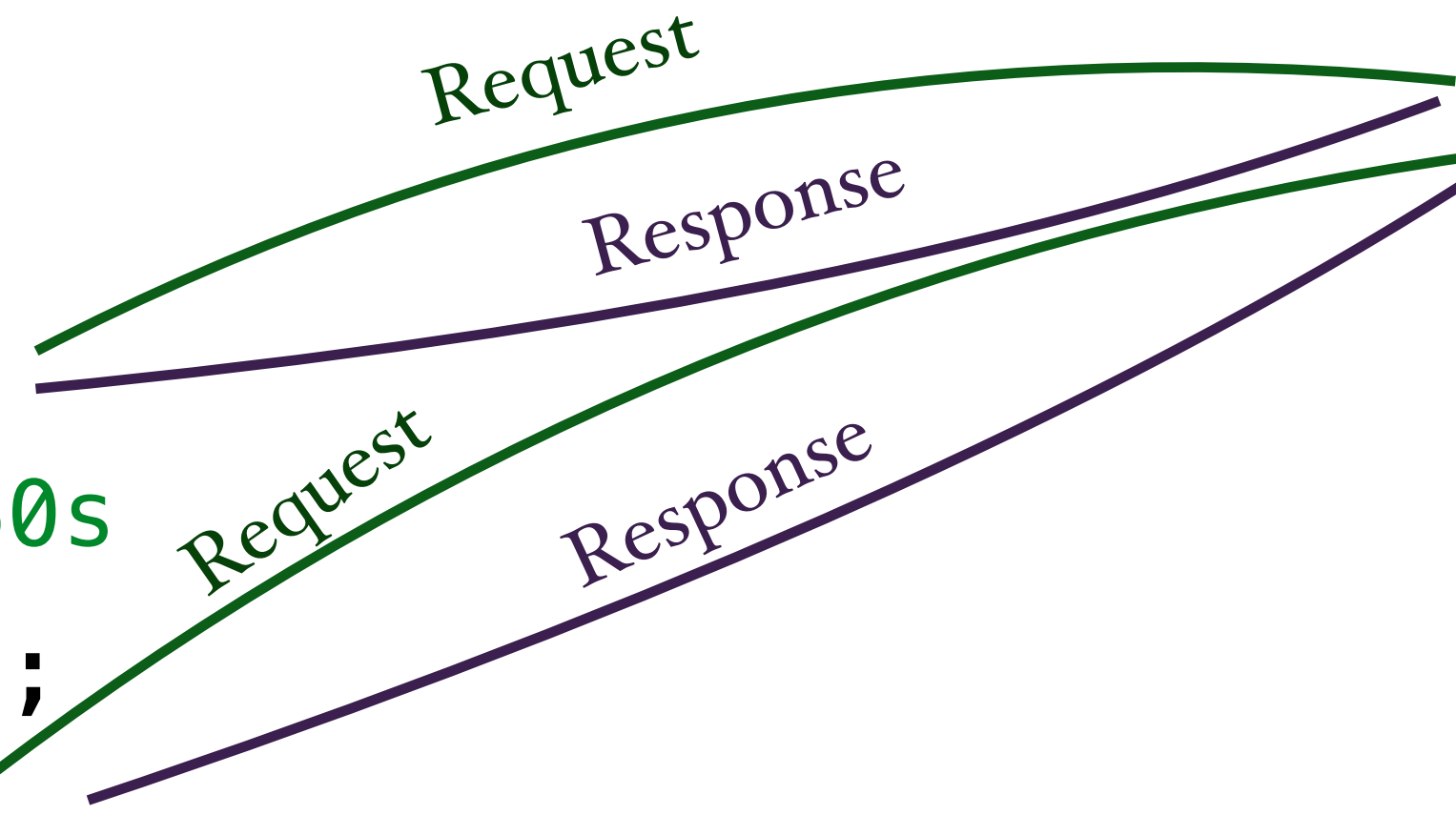
Process #1

```
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

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 patched.
- The memory under `PEB` may be patched.

Virtual Environment

Number Of Processors/Cores: Detection Technique

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 Table: Description

Description

- Technique based on raw and SMBIOS firmware tables content was implemented in VMDE tool.
- 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.

Virtual Environment

Firmware Table: Proposed Solution

Proposed Solution

- Systems older than Vista (`NtReadVirtualMemory` service splicing)
 - Check if read address equals to `0xC0000` or `0xE0000`.
 - Modify the buffer that is returned to user space.
- Vista and higher systems (`NtQuerySystemInformation` hook)
 - Check if `SystemInformationClass` is equal to `SystemFirmwareTableInformation`.
 - Parse the SFTI structure for `ProviderSignature` and `TableId`.
 - Check if the `ProviderSignature` is 'FIRM' or 'RSMB'.
 - Call the original `NtQuerySystemInformation` routine.
 - Modify the buffer that is returned to user space.

Sandbox Detection Evasion Tool



Sandbox Detection Evasion Tool

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

Sandbox Detection Evasion Tool

Supported Environments

- Cuckoo Sandbox
- Virtual Box
- VMWare
- Generic

Sandbox Detection Evasion Tool

Supported Generic Detection Types

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

Sandbox Detection Evasion Tool

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"  
  }  
}
```

Sandbox Detection Evasion Tool

Adding New Detection (Object)

- To add 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"  
  }  
}
```

Sandbox Detection Evasion Tool

New Virtual Environment Support

- To add new environment, the following template should be implemented (all generic detection types are supported by default).

```
class VMWare: VEDetection {
public:
    VMWare(const json_tiny &j) : VEDetection(j) {
        module_name = std::string("VMWARE");
    }
    virtual VMWare() {}

    static VEDetection* create_instance(const json_tiny &j);

    // overridden
    virtual void CheckAllCustom();

    // custom methods implementation go here. Should be called from CheckAllCustom
};
```

Sandbox Detection Evasion Tool

Output Interfaces

- **Report**
 - User-friendly HTML generated file.
 - Contains detection technique description.
 - Contains information on how to fix environment for specific detection technique.
- **Console**
 - Console mode.
 - Contains additional debug information.
- **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.

Sandbox Detection Evasion Tool

Report

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.	NO	Usage of per-thread delays accumulation.
InfiniteDelay	custom	Check if INFINITE delay is skipped due to sleep skipping.	NO	Adding conditional check for INFINITE delay.

VMWARE

Detection Name	Type	Description	Detected	Countermeasures
HypervisorPort	custom	Check if hypervisor port returns specific value.	YES	Disable hypervisor port.
DeviceNPF_NDIS	custom	Check if access to '\\.\NPF_NdisWanIp' device returns specific error code.	YES	Disable access to '\\.\NPF_NdisWanIp' device for the not-trusted porcesses.
HypervisorBit	custom	Check if hypervisor bit is enabled.	YES	cpuid instruction mask.
VMWare Tools Registry Key	registry	Check if `HKLM\SOFTWARE\VMware, Inc.\VMware Tools' Registry Key is present.	YES	Remove `HKLM\SOFTWARE\VMware, Inc.\VMware Tools' Registry Key.

Summary

- Many new detection/evasion techniques for Cuckoo Sandbox were introduced.
- We hope all of them will be fixed ASAP, as we are in contact with Cuckoo Sandbox developers and willing to share information.
- 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
 - Three output interfaces are supported: user-friendly HTML report, console, files

Q&A

<https://github.com/CheckPointSW/VB2016-sandbox-evasion>

Alexander Chailytko

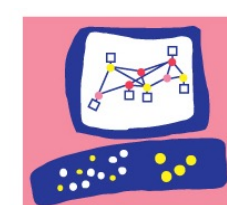
Team Leader

alexanderc@checkpoint.com

Stanislav Skuratovich

Malware Researcher

stanislavsk@checkpoint.com



Check Point[®]
SOFTWARE TECHNOLOGIES LTD.