Evasions: CPU

Go back (..)

Contents

CPU detection methods used

Techniques in this group use specific processor instructions to either get particular information about CPU — or execute predefined instruction sequence which behaves differently in usual host OS and in virtual environment.

1. Check vendor ID string via CPUID instruction

The `CPUID` (https://x86.renejeschke.de/html/file_module_x86_id_45.html) instruction is an instruction that returns processor identification and feature information to `EBX`, `ECX`, `EDX`. The information received to these registers can be used to identify a vendor.

Code sample

```
__declspec(naked) void get_cpuid_vendor(char *vendor_i
d) {
  __asm {
    ; save non-volatile register
    push ebx

    ; nullify output registers
    xor ebx, ebx
    xor ecx, ecx
    xor edx, edx

    ; call cpuid with argument in EAX
    mov eax, 0x40000000
    cpuid

    ; store vendor_id ptr to destination
    mov edi, vendor_id

    ; move string parts to destination
    mov eax, ebx  ; part 1 of 3 from EBX
    stosd
    mov eax, ecx  ; part 2 of 3 from ECX
    stosd
    mov eax, edx  ; part 3 of 3 from EDX
    stosd

    ; restore saved non-volatile register
    pop ebx

    ; return from function
    retn
  }
}
```

Detections table

Check vendor ID string via CPUID instruction - returned in parts in EBX, ECX, EDX:

Detect	EAX as argument to CPUID	String
FreeBSD HV	0x40000000	bhyve bhyve
Hyper-V	0x40000000	Microsoft Hv
KVM	0x40000000	KVMKVMKVM
Parallels	0x40000000	prl hyperv
VirtualBox	0x40000000	VBoxVBoxVBox
VirtualPC	0x40000000	Microsoft Hv
VMware	0x40000000	VMwareVMware
Xen	0x40000000	XenVMMXenVMM

2. Check if being run in Hypervisor via CPUID instruction

An other way to detect if the program is being run in hypervisor is using the `CPUID` instruction in an other way.

Instead of setting `EAX` (the argument to `CPUID`) to be `0x40000000`, `EAX` is set to 1.

When `EAX` is set to 1, the 31st bit in `ECX` (`CPUID`'s returned value) is set, it indicates that the program is being run in Hypervisor.

Code sample (function **GetAdaptersAddresses**)

```
__declspec(naked) bool is_run_in_hypervisor() {
  __asm {
    ; nullify output register
    xor ecx, ecx

    ; call cpuid with argument in EAX
    mov eax, 1
    cpuid

    ; set CF equal to 31st bit in ECX
    bt ecx, 31

    ; set AL to the value of CF
    setc al

    ; return from function
    retn
  }
}
```

Detections table

Check if being run in Hypervisor (via CPUID)		
Detect	**EAX as argument to CPUID**	**Check of return value**
Hypervisor	1	31st bit in ECX - set if run in Hypervisor

3. Check for global tables location: IDT/GDT/LDT

This technique doesn't work on latest VMware releases (all Windows releases affected). However, it is described here for the sake of completeness.

This trick involves looking at the pointers to critical operating system tables that are typically relocated on a virtual machine. It's what called "Red Pill" and was first introduced (http://web.archive.org/web/20070325211649/http://www.invisiblethings.org/papers/redpill.html) by Joanna Rutkowska.

There is one Local Descriptor Table Register (LDTR), one Global Descriptor Table Register (GDTR), and one Interrupt Descriptor Table Register (IDTR) per CPU. They have to be moved to a different location when a guest operating system is running to avoid conflicts with the host.

On real machines the IDT, for example, is located lower in memory than it is on guest (i.e., virtual) machines.

Code sample

```
idt_vm_detect = ((get_idt_base() >> 24) == 0xff);
ldt_vm_detect = (get_ldt_base() == 0xdead0000);
gdt_vm_detect = ((get_gdt_base >> 24) == 0xff);

// sidt instruction stores the contents of the IDT
Register
// (the IDTR which points to the IDT) in a processor
register.
ULONG get_idt_base() {
    UCHAR idtr[6];
#if defined (ENV32BIT)
    _asm sidt idtr
#endif
    return *((unsigned long *)&idtr[2]);
}

// sldt instruction stores the contents of the LDT
Register
// (the LDTR which points to the LDT) in a processor
register.
ULONG get_ldt_base() {
    UCHAR ldtr[5] = "\xef\xbe\xad\xde";
#if defined (ENV32BIT)
    _asm sldt ldtr
#endif
    return *((unsigned long *)&ldtr[0]);
}

// sgdt instruction stores the contents of the GDT
Register
// (the GDTR which points to the GDT) in a processor
register.
ULONG get_gdt_base() {
    UCHAR gdtr[6];
#if defined (ENV32BIT)
    _asm sgdt gdtr
#endif
```

```
        return gdt = *((unsigned long *)&gdtr[2]);
    }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

4. Using exotic instructions to fool virtual emulators

This technique is described by this link (https://www.slideshare.net/Cyphort/mmw-antisandbox-techniques) (slide #37).

MMX instructions may be used as random instructions by malware. Sometimes such subsets of CPU instruction are not supported by emulators and thus exception is thrown instead of performing analysis.

Example:

5. Detecting environment via execution of illegal instructions (VirtualPC only)

The malware executes illegal instructions, which should generate exception on the real CPU but are executed normally - or in some different way - in virtual environment.

Information about CPU exceptions is provided by this link (https://wiki.osdev.org/Exceptions#Invalid_Opcode).

Code sample (variant 1, generating #ud exception)

```
push ebx
xor ebx, ebx
mov eax, 1
; the following 4 bytes below generate #ud exception
db 0x0F
db 0x3F
db 0x0D
db 0x00
test ebx, ebx
setz al
pop ebx
```

It should be emphasized that there are more than 1,000 combinations of

```
0x0F
0x3F
0xXX
0xYY
```

bytes that may be used by malware in order to detect VirtualPC enviroment.

Code sample (variant 2, executing illegal STI instruction)

```
// Taken here: https://pastebin.com/Nsv5B1yk
// http://waleedassar.blogspot.com
// http://www.twitter.com/waleedassar
// Use this code to detect if Windows XP is running
inside Virtual PC 2007
#include "stdafx.h"
#include "windows.h"
#include "stdio.h"

#define CONTEXT_ALL 0x1003F

int dummy(int);
unsigned long gf=0;

int __cdecl Handler(EXCEPTION_RECORD* pRec,void* est,u
nsigned char* pContext,void* disp)
{
    if(pRec->ExceptionCode==0xC0000096)  //Privileged
instruction
    {
        //----------------------Installing the
trick-------------------------------------
        *(unsigned long*)(pContext)=CONTEXT_ALL;/
*CONTEXT_DEBUG_REGISTERS|CONTEXT_FULL*/
        *(unsigned long*)(pContext+0x4)=(unsigned
long)(&dummy);
        *(unsigned long*)(pContext+0x8)=(unsigned
long)(&dummy);
        *(unsigned long*)(pContext+0xC)=(unsigned
long)(&dummy);
        *(unsigned long*)(pContext+0x10)=(unsigned lon
g)(&dummy);
        *(unsigned long*)(pContext+0x14)=0;
        *(unsigned long*)(pContext+0x18)=0x155; //
Enable the four DRx On-Execute
        //-----------------------------------------
-------------------------------------
        (*(unsigned long*)(pContext+0xB8))++;
```

```
        return ExceptionContinueExecution;
    }
    else if(pRec-
>ExceptionCode==EXCEPTION_SINGLE_STEP)
    {
        if(gf==1)
        {
            MessageBox(0,"Expected behavior (XP)","wal
iedassar",0);
            ExitProcess(0);
        }
        gf++;
        (*(unsigned long*)(pContext+0xC0))|
=0x00010000; //Set the RF (Resume Flag)
        return ExceptionContinueExecution;
    }
    return ExceptionContinueSearch;
}

int dummy(int x)
{
    x+=0x100;
    return x;
}

int main(int shitArg)
{
    unsigned long ver_=GetVersion();
    unsigned long major=ver_&0xFF;
    unsigned long minor=(ver_>>0x8)&0xFF;
    if(major==0x05 & minor==0x01) //Windows XP
    {
        unsigned long x=0;
        __asm
        {
            push offset Handler
            push dword ptr fs:[0x0]
            mov dword ptr fs:[0x0],esp
            STI; Triggers an exception(privileged inst
ruction)
```

```
        }
        dummy(0xFF);
        __asm
        {
            pop dword ptr fs:[0x0]
            pop ebx
        }
        MessageBox(0,"Virtual PC 2007 detected (XP)","
  waliedassar",0);
    }
    return 0;
}
```

Code sample (variant 3, resetting VirtualPC)

```
// Taken here: https://pastebin.com/exAK5XQx
// http://waleedassar.blogspot.com (@waleedassar)
// Executing "\x0F\xC7\xC8\x05\x00" in VirtualPC 2007
triggers a reset error.
#include "stdafx.h"
#include "windows.h"
#include "stdio.h"

bool flag=false;

int __cdecl Handler(EXCEPTION_RECORD* pRec,void* est,u
nsigned char* pContext,void* disp)
{
    if(pRec->ExceptionCode==0xC000001D  || pRec->Excep
tionCode==0xC000001E || pRec->ExceptionCode==0xC000000
5)
    {
        flag=true;
        (*(unsigned long*)(pContext+0xB8))+=5;
        return ExceptionContinueExecution;
    }
    return ExceptionContinueSearch;
}

int main(int argc, char* argv[])
{
    __asm
    {
        push offset Handler
        push dword ptr fs:[0x0]
        mov dword ptr fs:[0x0],esp
    }
    flag=false;
    __asm
    {
        __emit 0x0F
        __emit 0xC7
        __emit 0xC8
```

```
            __emit 0x05
            __emit 0x00
        }
        if(flag==false)
        {
            MessageBox(0,"VirtualPC detected","waliedassar
  ",0);
        }
        __asm
        {
            pop dword ptr fs:[0x0]
            pop eax
        }
        return 0;
  }
```

6. Detecting environment via IN instruction - backdoor port (VMware only)

This article (https://sites.google.com/site/chitchatvmback/backdoor) explains why backdoor port communication is used in VMware in the first place.

Code sample (variant 1)

```
bool VMWare::CheckHypervisorPort() const {
    bool is_vm = false;
    __try {
        __asm {
            push edx
            push ecx
            push ebx
            mov eax, 'VMXh'
            mov ebx, 0
            mov ecx, 10
            mov edx, 'VX'
            in eax, dx      // <- key point is here
            cmp ebx, 'VMXh'
            setz[is_vm]
            pop ebx
            pop ecx
            pop edx
        }
    }
    __except (EXCEPTION_EXECUTE_HANDLER) {
        is_vm = false;
    }
    return is_vm;
}
```

Code sample (variant 2)

```
bool VMWare::CheckHypervisorPortEnum() const {
    bool is_vm = false;
    short ioports[] = { 'VX' , 'VY' };
    short ioport;
    for (short i = 0; i < _countof(ioports); ++i) {
        ioport = ioports[i];
        for (unsigned char cmd = 0; cmd < 0x2c; +
+cmd) {
            __try {
                __asm {
                    push eax
                    push ebx
                    push ecx
                    push edx
                    mov eax, 'VMXh'
                    movzx ecx, cmd
                    mov dx, ioport
                    in eax,
dx      // <- key point is here
                    pop edx
                    pop ecx
                    pop ebx
                    pop eax
                }
                is_vm = true;
                break;
            }
            __except (EXCEPTION_EXECUTE_HANDLER) {}
        }
        if (is_vm)
            break;
    }
    return is_vm;
}
```

Signature recommendations

No signature recommendations are provided for this evasion group as it's hard to track such a code being executed.

Countermeasures

Patch hypervisor. If it proves impossible — due to license issues or something else — patch VM config. Usually undocumented options help.

- vs CPUID instruction: refer to this article (http://vknowledge.net/2014/04/17/how-to-fake-a-vms-guest-os-cpuid/) for the example of such a patch
- vs IN instruction (VMware backdoor): take a look at these config changes (https://wasm.in/threads/izmenenie-raboty-backdoor-interfejsa-v-vmware.24564/#post-291532)

Credits

Credits go to open-source project from where code samples were taken and to independent researcher who shared his findings:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)
- @waleedassar (https://twitter.com/waleedassar)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Share this:

(http://www.reddit.com/submit?url=https://evasions.checkpoint.com/src/Evasions/techniques/cpu.html&title=Evasions:%20CPU - Evasion Techniques) (http://news.ycombinator.com/submitlink?u=https://evasions.checkpoint.com/src/Evasions/techniques/cpu.html&t=Evasions:%20CPU - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/cpu.html&text=Evasions:%20CPU - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/cpu.html&title=Evasions:%20CPU)

Evasions: Filesystem

Go back (..)

Contents

Filesystem detection methods

The principle of all the filesystem detection methods is the following: there are no such files and directories in usual host; however they exist in particular virtual environments and sandboxes. Virtual environment may be detected if such an artifact is present.

1. Check if specific files exist

This method uses the difference in files which are present in usual host system and virtual environments. There are quite a few file artifacts present in virtual environments which are specific for such kinds of systems. These files are not present on usual host systems where no virtual environment is installed.

Function used:

- GetFileAttributes // if attributes are invalid then no file exists

Code sample

```
BOOL is_FileExists(TCHAR* szPath)
{
    DWORD dwAttrib = GetFileAttributes(szPath);
    return (dwAttrib != INVALID_FILE_ATTRIBUTES) && !
(dwAttrib & FILE_ATTRIBUTE_DIRECTORY);
}

/*
Check against some of VMware blacklisted files
*/
VOID vmware_files()
{
    /* Array of strings of blacklisted paths */
    TCHAR* szPaths[] = {
        _T("system32\\drivers\\vmmouse.sys"),
        _T("system32\\drivers\\vmhgfs.sys"),
    };

    /* Getting Windows Directory */
    WORD dwlength = sizeof(szPaths) /
sizeof(szPaths[0]);
    TCHAR szWinDir[MAX_PATH] = _T("");
    TCHAR szPath[MAX_PATH] = _T("");
    GetWindowsDirectory(szWinDir, MAX_PATH);

    /* Check one by one */
    for (int i = 0; i < dwlength; i++)
    {
        PathCombine(szPath, szWinDir, szPaths[i]);
        TCHAR msg[256] = _T("");
        _stprintf_s(msg, sizeof(msg) / sizeof(TCHAR),
_T("Checking file %s: "), szPath);
        if (is_FileExists(szPath))
            print_results(TRUE, msg);
        else
            print_results(FALSE, msg);
```

```
        }
    }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Signature recommendations

If the following function contains its only argument from the table column `Path`:

- GetFileAttributes(path)

then it's an indication of application trying to use the evasion technique.

Detections table

Check if the following files exist:		
Detect	**Path**	**Details (if any)**
[general]	c:\[60 random hex symbols]	file unique to the PC used for encoding
	c:\take_screenshot.ps1	
	c:\loaddll.exe	
	c:\email.doc	
	c:\email.htm	
	c:\123\email.doc	
	c:\123\email.docx	
	c:\a\foobar.bmp	
	c:\a\foobar.doc	
	c:\a\foobar.gif	
	c:\symbols\aagmmc.pdb	
Parallels	c:\windows\system32\drivers\prleth.sys	Network Adapter
	c:\windows\system32\drivers\prlfs.sys	
	c:\windows\system32\drivers\prlmouse.sys	Mouse Synchronization Tool
	c:\windows\system32\drivers\prlvideo.sys	
	c:\windows\system32\drivers\prltime.sys	Time Synchronization Driver
	c:\windows\system32\drivers\prl_pv32.sys	Paravirtualization Driver

	c:\windows\system32\drivers\prl_paravirt_32.sys	Paravirtualization Driver
VirtualBox	c:\windows\system32\drivers\VBoxMouse.sys	
	c:\windows\system32\drivers\VBoxGuest.sys	
	c:\windows\system32\drivers\VBoxSF.sys	
	c:\windows\system32\drivers\VBoxVideo.sys	
	c:\windows\system32\vboxdisp.dll	
	c:\windows\system32\vboxhook.dll	
	c:\windows\system32\vboxmrxnp.dll	
	c:\windows\system32\vboxogl.dll	
	c:\windows\system32\vboxoglarrayspu.dll	
	c:\windows\system32\vboxoglcrutil.dll	
	c:\windows\system32\vboxoglerrorspu.dll	
	c:\windows\system32\vboxoglfeedbackspu.dll	
	c:\windows\system32\vboxoglpackspu.dll	
	c:\windows\system32\vboxoglpassthroughspu.dll	
	c:\windows\system32\vboxservice.exe	
	c:\windows\system32\vboxtray.exe	
	c:\windows\system32\VBoxControl.exe	
VirtualPC	c:\windows\system32\drivers\vmsrvc.sys	
	c:\windows\system32\drivers\vpc-s3.sys	
VMware	c:\windows\system32\drivers\vmmouse.sys	Pointing PS/2 Device Driver
	c:\windows\system32\drivers\vmnet.sys	
	c:\windows\system32\drivers\vmxnet.sys	PCI Ethernet Adapter
	c:\windows\system32\drivers\vmhgfs.sys	HGFS Filesystem Driver
	c:\windows\system32\drivers\vmx86.sys	
	c:\windows\system32\drivers\hgfs.sys	

2. Check if specific directories are present

This method uses the difference in directories which are present in usual host system and virtual environments. There are quite a few directory artifacts present in virtual environments which are specific for such kinds of systems. These directories are not present on usual host systems where no virtual environment is installed.

Function used:

- GetFileAttributes // if attributes are invalid then no file exists

Code sample

```
BOOL is_DirectoryExists(TCHAR* szPath)
{
    DWORD dwAttrib = GetFileAttributes(szPath);
    return (dwAttrib != INVALID_FILE_ATTRIBUTES) && (d
wAttrib & FILE_ATTRIBUTE_DIRECTORY);
}

/*
Check against VMware blacklisted directory
*/
BOOL vmware_dir()
{
    TCHAR szProgramFile[MAX_PATH];
    TCHAR szPath[MAX_PATH] = _T("");
    TCHAR szTarget[MAX_PATH] = _T("VMware\\");
    if (IsWoW64())

ExpandEnvironmentStrings(_T("%ProgramW6432%"), szProgr
amFile, ARRAYSIZE(szProgramFile));
    else
        SHGetSpecialFolderPath(NULL, szProgramFile, CS
IDL_PROGRAM_FILES, FALSE);
    PathCombine(szPath, szProgramFile, szTarget);
    return is_DirectoryExists(szPath);
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Signature recommendations

If the following function contains its only argument from the table column `Path`:

- GetFileAttributes(path)

then it's an indication of application trying to use the evasion technique.

Detections table

Check if the following files exist:	
Detect	**Path**
CWSandbox	c:\analysis
VirtualBox	%PROGRAMFILES%\oracle\virtualbox guest additions\
VMware	%PROGRAMFILES%\VMware\

3. Check if full path to the executable contains one of the specific strings

This method relies on peculiarities of launching executables inside virtual environments. Some environments launch executables from specific paths - and malware samples check these paths.

Functions used to get executable path:

- GetModuleFileName
- GetProcessImageFileNameA/W
- QueryFullProcessImageName

Code sample (function GetModuleFileName)

```
int gensandbox_path() {
    char path[500];
    size_t i;
    DWORD pathsize = sizeof(path);

    GetModuleFileName(NULL, path, pathsize);

    for (i = 0; i < strlen(path); i++) { /* case-
insensitive */
        path[i] = toupper(path[i]);
    }

    // some sample values from the table
    if (strstr(path, "\\SAMPLE") != NULL) {
        return TRUE;
    }
    if (strstr(path, "\\VIRUS") != NULL) {
        return TRUE;
    }
    if (strstr(path, "SANDBOX") != NULL) {
        return TRUE;
    }

    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Code sample (function QueryFullProcessImageName)

```
DWORD PID = 1337; // process ID of the target process
HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATIO
N, false, PID);
DWORD value = MAX_PATH;
char buffer[MAX_PATH];
QueryFullProcessImageName(hProcess, 0, buffer, &value)
;
printf("EXE Path: %s\n", buffer);
```

No signature recommendations

Signature recommendations are not provided as it's hard to say why exactly application wants to get its full path. Function calls may be hooked - and that's it, just general recommendation.

Detections table

Check if full path to the executable contains one of the following strings:	
Detect	**String**
[general]	\sample
	\virus
	sandbox

4. Check if the executable is run from specific directory

This method relies on peculiarities of launching executables inside virtual environments. Some environments launch executables from specific directories - and malware samples check these directories.

It's just a particular case of checking presence of specific strings in full application path, please refer to the section above for code sample and signature recommendations.

As this very method is pretty old and is not commonly used, the links to external sources are provided for the reference on this method:

- VB code sample (https://www.opensc.io/showthread.php?t=2343)
- python code sample (https://github.com/brad-accuvant/community-modified/blob/master/modules/signatures/antisandbox_joe_anubis_files.py)
- anti-emulation tricks (http://web.archive.org/web/20181222042516/www.woodmann.com/forum/showthread.php?12545-Anti-Emulation-Tricks)
- stub for C code (http://web.archive.org/web/20101026233743/http://evilcry.netsons.org/OC0/code/EmulationAwareness.c)

Detections table

Check if the executable is run from the following directories:	
Detect	**Path**
Anubis	c:\insidetm

5. Check if the executable files with specific names are present in physical disk drives' root

This method relies on peculiarities of virtual environments, in this case it’s presence of specific files in disk root root directories.

Function used:

- GetFileAttributes // if attributes are invalid then no file exists

Code sample (function GetModuleFileName)

```
int pafish_exists_file(char * filename) {
    DWORD res = INVALID_FILE_ATTRIBUTES;
    if (pafish_iswow64() == TRUE) {
        void *old = NULL;
        // Disable redirection immediately prior to
calling GetFileAttributes.
        if
(pafish_disable_wow64_fs_redirection(&old) ) {
            res = GetFileAttributes(filename);
            // Ignoring MSDN recommendation of exiting
if this call fails.
            pafish_revert_wow64_fs_redirection(old);
        }
    }
    else {
        res = GetFileAttributes(filename);
    }
    return (res != INVALID_FILE_ATTRIBUTES) ? TRUE : F
ALSE;
}

int gensandbox_common_names() {
    DWORD dwSize = MAX_PATH;
    char szLogicalDrives[MAX_PATH] = {0};
    DWORD dwResult = GetLogicalDriveStrings(dwSize,szL
ogicalDrives);
    BOOL exists;

    if (dwResult > 0 && dwResult <= MAX_PATH)
    {
        char* szSingleDrive = szLogicalDrives;
        char filename[MAX_PATH] = {0};
        while(*szSingleDrive)
        {
            if (GetDriveType(szSingleDrive) != DRIVE_R
EMOVABLE ) {
                snprintf(filename, MAX_PATH, "%ssample
.exe",szSingleDrive);
```

```
                exists = pafish_exists_file(filename);
                if (exists) return TRUE;

                snprintf(filename, MAX_PATH, "%smalwar
e.exe",szSingleDrive);
                exists = pafish_exists_file(filename);
                if (exists) return TRUE;
            }

            szSingleDrive += strlen(szSingleDrive) +
1;
        }
    }

    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Signature recommendations

If the following function contains its only argument from the table column `Path`:

- GetFileAttributes(path)

then it's an indication of application trying to use the evasion technique.

Detections table

Check if the executables with particular names are present in disk root:	
Detect	**Path**
[general]	malware.exe
	sample.exe

Countermeasures

Hook target functions and return appropriate results if indicators (files from tables) are checked.

Credits

Credits go to open-source projects from where code samples were taken:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)
- pafish project on github (https://github.com/a0rtega/pafish)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasion Techniques) in (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/filesystem.html&title=Evasions:%20Filesystem)

Evasions: Firmware tables

Go back (..)

Contents

Firmware tables detection methods

There are special memory areas used by OS which contain specific artifacts if OS is run under virtual environment. These memory areas may be dumped using different methods depending on the OS version.

Firmware tables are retrieved via SYSTEM_FIRMWARE_TABLE_INFORMATION object. It's defined the following way:

```
typedef struct _SYSTEM_FIRMWARE_TABLE_INFORMATION {
    ULONG ProviderSignature;
    SYSTEM_FIRMWARE_TABLE_ACTION Action;
    ULONG TableID;
    ULONG TableBufferLength;
    UCHAR TableBuffer[ANYSIZE_ARRAY];  // <- the
result will reside in this field
} SYSTEM_FIRMWARE_TABLE_INFORMATION, *PSYSTEM_FIRMWARE
_TABLE_INFORMATION;

// helper enum
typedef enum _SYSTEM_FIRMWARE_TABLE_ACTION
{
    SystemFirmwareTable_Enumerate,
    SystemFirmwareTable_Get
} SYSTEM_FIRMWARE_TABLE_ACTION, *PSYSTEM_FIRMWARE_TABL
E_ACTION;
```

1. Check if specific strings are present in Raw Firmware Table

Retrieved firmware table is scanned for the presence of particular strings.

Depending on Windows version different functions are used for this check. See code samples below.

1.1. Windows Vista+

Code sample

```
// First, SYSTEM_FIRMWARE_TABLE_INFORMATION object is
initialized in the following way:
SYSTEM_FIRMWARE_TABLE_INFORMATION *sfti =
    (PSYSTEM_FIRMWARE_TABLE_INFORMATION)HeapAlloc(GetP
rocessHeap(), HEAP_ZERO_MEMORY, Length);
sfti->Action = SystemFirmwareTable_Get;  // 1
sfti->ProviderSignature = 'FIRM';
sfti->TableID = 0xC0000;
sfti->TableBufferLength = Length;

// Then initialized SYSTEM_FIRMWARE_TABLE_INFORMATION
object is used as an argument for
// the system information call in the following way in
order to dump raw firmware table:
NtQuerySystemInformation(
    SystemFirmwareTableInformation,  // 76
    sfti,
    Length,
    &Length);
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Signature recommendations

If the function

- NtQuerySystemInformation

contains:

- 1st argument equal to 76 (SystemFirmwareTableInformation)
- 2nd argument has sfti->ProviderSignature field initialized to 'FIRM' and sfti->Action field initialized to 1

then it's an indication of application trying to use this evasion technique.

1.2. Windows XP

Code sample

```
// In case if OS version is Vista+ csrss.exe memory
space is read in order to dump raw firmware table:
hCSRSS = OpenProcess(PROCESS_QUERY_INFORMATION | PROCE
SS_VM_READ, FALSE, csrss_pid);

NtReadVirtualMemory(
     hCSRSS,
     0xC0000,
     sfti,
     RegionSize,
     &memIO);
```

Signature recommendations

If the following function contains PID of csrss.exe process as its 3rd argument:

- HANDLE hCSRSS = OpenProcess(..., csrss_pid)

and is followed by the call to the following function:

- NtReadVirtualMemory(hCSRSS, 0xC0000, ...)

which contains:

- 1st argument equal to csrss.exe handle
- 2nd argument equal to 0xC0000

then it’s an indication of application trying to use this evasion technique.

Detections table

Check if the following strings are present in Raw Firmware Table:	
Detect	**String**
Parallels	Parallels(R)

VirtualBox	Innotek
	Oracle
	VirtualBox
VirtualPC	S3 Corp.
VMware	VMware

2. Check if specific strings are present in Raw SMBIOS Firmware Table

Retrieved firmware table is scanned for the presence of particular strings.

Depending on Windows version different functions are used for this check. See code samples below.

2.1. Windows Vista+

Code sample

```
// SYSTEM_FIRMWARE_TABLE_INFORMATION object is
initialized in the following way:
SYSTEM_FIRMWARE_TABLE_INFORMATION *sfti =
    (PSYSTEM_FIRMWARE_TABLE_INFORMATION)HeapAlloc(GetP
rocessHeap(), HEAP_ZERO_MEMORY, Length);
sfti->Action = SystemFirmwareTable_Get; // 1
sfti->ProviderSignature = 'RSMB';
sfti->TableID = 0;
sfti->TableBufferLength = Length;

// Then initialized SYSTEM_FIRMWARE_TABLE_INFORMATION
object is used as an argument for
// the system information call in the following way in
order to dump raw firmware table:
NtQuerySystemInformation(
    SystemFirmwareTableInformation,  // 76
    sfti,
    Length,
    &Length);
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Signature recommendations

If the following function:

- NtQuerySystemInformation

contains:

- 1st argument equal to 76 (SystemFirmwareTableInformation)
- 2nd argument has sfti->ProviderSignature field initialized to 'RSMB' and sfti->Action field initialized to 1

then it's an indication of application trying to use this evasion technique.

2.2. Windows XP

Code sample

```
// In case if OS version is Vista+ csrss.exe memory
space is read in order to dump raw firmware table:
hCSRSS = OpenProcess(PROCESS_QUERY_INFORMATION | PROCE
SS_VM_READ, FALSE, csrss_pid);

NtReadVirtualMemory(
     hCSRSS,
     0xE0000,
     sfti,
     RegionSize,
     &memIO);
```

Signature recommendations

If the following function contains PID of csrss.exe process as its 3rd argument:

- HANDLE hCSRSS = OpenProcess(..., csrss_pid)

and is followed by the call to the following function:

- NtReadVirtualMemory(hCSRSS, 0xE0000, ...)

which contains:

- 1st argument equal to csrss.exe handle
- 2nd argument equal to 0xE0000

then it's an indication of application trying to use this evasion technique.

Detections table

Check if the following strings are present in Raw SMBIOS Firmware Table:	
Detect	**String**
Parallels	Parallels Software International
VirtualBox	Innotek
	Oracle
	VirtualBox
VirtualPC	VS2005R2
VMware	VMware, Inc.
	VMware

Countermeasures

- On systems older than Vista change memory content of csrss.exe at given addresses.
- On Vista+ OS hook NtQuerySystemInformation for retrieving SystemFirmwareTableInformation class and parse SFTI structure for provided field values.

Credits

Credits go to open-source project from where code samples were taken:

- VMDE project on github (https://github.com/hfiref0x/VMDE)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions: Generic OS queries

Go back (..)

Contents

Signature recommendations are general

Signature recommendations are general for each technique: hook the function used and track if it is called. It's pretty hard to tell why application wants to get user name, for example. It doesn't necessarily mean applying evasion technique. So the best what can be done in this situation is intercepting target functions and tracking their calls.

Detection via generic OS checks

Usual hosts have meaningful and non-standard usernames/computer names. Particular virtual environments assign some predefined names to default users as well as computer names. Other differences between host OS and VMs include RAM size, HDD size, quantity of monitors - and so on. While these may be not the most reliable ways to detect virtual environments, they are still commonly used in malware samples.

1. Check if the username is specific

Please note that checks are not case-sensitive.

Function used:

- GetUserNameA/W

Code sample

```
bool is_user_name_match(const std::string &s) {
    auto out_length = MAX_PATH;
    std::vector<uint8_t> user_name(out_length, 0);
    ::GetUserNameA((LPSTR)user_name.data(),
(LPDWORD)&out_length);

    return (!lstrcmpiA((LPCSTR)user_name.data(), s.c_s
tr()));
}
```

Code sample is taken from InviZzzible tool (https://github.com/CheckPointSW/InviZzzible)

Countermeasures

Change user name to non-suspicious one.

Detections table

Check if username is one of the following:	
Detect	**String**
[general]	admin
	andy
	honey
	john
	john doe
	malnetvm
	maltest
	malware
	roo
	sandbox
	snort
	tequilaboomboom
	test
	virus
	virusclone
	wilbert
Nepenthes	nepenthes
Norman	currentuser
ThreatExpert	username
Sandboxie	user
VMware	vmware

2. Check if the computer name is specific

Please note that checks are not case-sensitive.

Function used:

- GetComputerNameA/W

Code sample

```
bool is_computer_name_match(const std::string &s) {
    auto out_length = MAX_PATH;
    std::vector<uint8_t> comp_name(out_length, 0);
    ::GetComputerNameA((LPSTR)comp_name.data(), (LPDWO
RD)&out_length);

    return (!lstrcmpiA((LPCSTR)comp_name.data(), s.c_s
tr()));
}
```

Code sample is taken from InviZzzible tool (https://github.com/CheckPointSW/InviZzzible)

Countermeasures

Change computer name to non-suspicious one.

Detections table

Check if computer name is one of the following:	
Detect	**String**
[generic]	klone_x64-pc
	tequilaboomboom
Anubis	TU-4NH09SMCG1HC
	InsideTm

3. Check if the host name is specific

Please note that checks are not case-sensitive.

Function used:

- GetComputerNameExA/W

Code sample

```
bool is_host_name_match(const std::string &s) {
    auto out_length = MAX_PATH;
    std::vector<uint8_t> dns_host_name(out_length, 0);
    ::GetComputerNameExA(ComputerNameDnsHostname, (LPS
TR)dns_host_name.data(), (LPDWORD)&out_length);

    return (!lstrcmpiA((LPCSTR)dns_host_name.data(),
s.c_str()));
}
```

Code sample is taken from InviZzzible tool (https://github.com/CheckPointSW/InviZzzible)

Countermeasures

Change host name to non-suspicious one.

Detections table

Check if host name is one of the following:	
Detect	**String**
[generic]	SystemIT

4. Check if the total RAM is low

Functions used to get executable path:

- GetMemoryStatusEx

Code sample

```
BOOL memory_space()
{
    DWORDLONG ullMinRam = (1024LL * (1024LL * (1024LL
* 1LL))); // 1GB

    MEMORYSTATUSEX statex = {0};
    statex.dwLength = sizeof(statex);
    GlobalMemoryStatusEx(&statex); // calls
NtQuerySystemInformation

    return (statex.ullTotalPhys < ullMinRam) ? TRUE :
FALSE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Countermeasures

Patch/hook NtQuerySystemInformation to return new number of PhysicalPages in SystemBasicInformation.

Tip: in this case its 1st argument is equal to 2 - SystemPerformanceInformation enum value.

Alternatively, patch NumberOfPhysicalPages in KUSER_SHARED_DATA.

5. Check if the screen resolution is non-usual for host OS

The following set of functions is used:

- GetDesktopWindow
- GetWindowRect

Alternatively:

- GetSystemMetrics
- SystemParametersInfo

- GetMonitorInfo

Code sample

Take a look at this StackOverflow thread (https://stackoverflow.com/questions/4631292/how-detect-current-screen-resolution).

Countermeasures

Change screen resolution for it to match the resolution of usual host (1600x900, for example).

6. Check if the number of processors is low

Function used:

- GetSystemInfo

Besides this function numbers of processors can be obtained from PEB, via either asm inline or intrinsic function, see code samples below. It can be also obtained (ActiveProcessorCount flag) from the KUSER_SHARED_DATA structure.

Code sample (variant 1, al-khaser project)

```
BOOL NumberOfProcessors()
{
#if defined (ENV64BIT)
        PULONG ulNumberProcessors = (PULONG)(__readgsq
word(0x30) + 0xB8);
#elif defined(ENV32BIT)
        PULONG ulNumberProcessors = (PULONG)(__readfsd
word(0x30) + 0x64);
#endif

    if (*ulNumberProcessors < 2)
        return TRUE;
    else
        return FALSE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Code sample (variant 2, al-khaser project, asm inline)

```
__declspec(naked)
DWORD get_number_of_processors() {
    __asm {
        ; get pointer to Process Environment Block (PE
B)
        mov eax, fs:0x30

        ; read the field containing target number
        mov eax, [eax + 0x64]

        ; return from function
        retn
    }
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Code sample (variant 3, pafish project)

```
int gensandbox_one_cpu_GetSystemInfo() {
    SYSTEM_INFO si;
    GetSystemInfo(&si);
    return si.dwNumberOfProcessors < 2 ? TRUE : FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Code sample (variant 4)

```
__declspec(naked)
    DWORD get_number_of_active_processors() {
    __asm {
        mov eax, 0x7ffe0000  ; KUSER_SHARED_DATA struc
ture fixed address
        mov eax, byte ptr [eax+0x3c0] ; checking Activ
eProcessorCount
        retn  ; return from function
    }
}
```

Countermeasures

Assign two or more cores for Virtual Machine.

As an alternative solution, patch/hook NtCreateThread to assign specific core for each new thread.

7. Check if the quantity of monitors is small

Functions used:

- EnumDisplayMonitors
- GetSystemMetrics (SM_MONITOR)

Code sample

```
BOOL CALLBACK MonitorEnumProc(HMONITOR hMonitor, HDC h
dcMonitor, LPRECT lprcMonitor, LPARAM dwData)
{
    int *Count = (int*)dwData;
    (*Count)++;
    return TRUE;
}

int MonitorCount()
{
    int Count = 0;
    if (EnumDisplayMonitors(NULL, NULL, MonitorEnumPro
c, (LPARAM)&Count))
        return Count;
    return -1; // signals an error
}
```

Credits for this code sample: StackOverflow forum (https://stackoverflow.com/questions/7767036/how-do-i-get-the-number-of-displays-in-windows)

Countermeasures

Add at least one monitor to virtual environment.

8. Check if the hard disk drive size and free space are small

Functions used:

- DeviceIoControl(..., IOCTL_DISK_GET_LENGTH_INFO, ...)
- GetDiskFreeSpaceExA/W

Code sample (checking drive total size)

```
int gensandbox_drive_size() {
    GET_LENGTH_INFORMATION size;
    DWORD lpBytesReturned;

    HANDLE drive = CreateFile("\\\\.\
\PhysicalDrive0", GENERIC_READ, FILE_SHARE_READ,
NULL, OPEN_EXISTING, 0, NULL);
    if (drive == INVALID_HANDLE_VALUE) {
        // Someone is playing tricks. Or not enough
privileges.
        CloseHandle(drive);
        return FALSE;
    }
    BOOL result = DeviceIoControl(drive, IOCTL_DISK_GE
T_LENGTH_INFO, NULL, 0, &size, sizeof(GET_LENGTH_INFOR
MATION), &lpBytesReturned, NULL);
    CloseHandle(drive);

    if (result != 0) {
        if (size.Length.QuadPart / 1073741824 <= 60) /
* <= 60 GB */
        return TRUE;
    }

    return FALSE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Code sample (checking drive free space)

```
int gensandbox_drive_size2() {
    ULARGE_INTEGER total_bytes;

    if (GetDiskFreeSpaceExA("C:\\", NULL,
&total_bytes, NULL))
    {
        if (total_bytes.QuadPart / 1073741824 <= 60) /
* <= 60 GB */
        return TRUE;
    }

    return FALSE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Countermeasures

Against checking disk size: filter IRP device control requests to \\Device\\HarddiskN with specific CTL-codes:

- DRIVE_GEOMETRY_EX
- DRIVE_LAYOUT_EX
- PARTITION_INFO_EX

Against checking free space: patch/hook NtQueryVolumeInformationFile to process these classes:

- FileFsSizeInformation
- FileFsFullSizeInformation

in case if handle points to \\Device\\HarddiskVolumeN.

9. Check if the system uptime is small

Function used:

- GetTickCount
- GetTickCount64
- NtQuerySystemInformation

Code sample

```
bool Generic::CheckSystemUptime() const {
    const DWORD uptime = 1000 * 60 * 12; // 12 minutes
    return GetTickCount() < uptime;
}
```

Code sample is taken from InviZzzible tool (https://github.com/CheckPointSW/InviZzzible)

Code sample

```
#define MIN_UPTIME_MINUTES 12
BOOL uptime_check()
{
    ULONGLONG uptime_minutes = GetTickCount64() / (60 * 1000);
    return uptime_minutes < MIN_UPTIME_MINUTES;
}
```

Code sample

```
BOOL uptime_check2()
{
    SYSTEM_TIME_OF_DAY_INFORMATION  SysTimeInfo;
    ULONGLONG uptime_minutes;
    NtQuerySystemInformation(SystemTimeOfDayInformatio
n, &SysTimeInfo, sizeof(SysTimeInfo), 0);
    uptime_minutes =
(SysTimeInfo.CurrentTime.QuadPart - SysTimeInfo.BootTi
me.QuadPart) / (60 * 1000 * 10000);
    return uptime_minutes < MIN_UPTIME_MINUTES;
}
```

Countermeasures

- Adjust KeBootTime value
- Adjust SharedUserData->TickCount, SharedUserData->TickCoundLowDeprecated values

10. Check if the OS was boot from virtual hard disk (Win8+)

Function used:

- IsNativeVhdBoot // false on host OS, true within VM

Code sample (excerpt from malware)

Take a look at the excerpt from malware here (https://github.com/a0rtega/pafish/issues/46).

Code sample (pafish project)

```
int gensandbox_IsNativeVhdBoot() {
    BOOL isnative = FALSE;

    IsNativeVhdBoot fnnative = (IsNativeVhdBoot) GetPr
ocAddress(
        GetModuleHandleA("kernel32"), "IsNativeVhdBoot
");

    /* IsNativeVhdBoot always returns 1 on query
success */
    if (fnnative)
        fnnative(&isnative);

    return (isnative) ? TRUE : FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Countermeasures

Hook IsNativeVhdBoot and change its result to the one required.

Countermeasures

Countermeasures are present in appropriate sub-sections, see above.

Credits

Credits go to open-source projects from where code samples were taken:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)

- pafish project on github (https://github.com/a0rtega/pafish)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions: Global OS Objects

Go back (..)

Contents

Global objects detection methods

The principle of all the global objects detection methods is the following: there are no such objects in usual host; however they exist in particular virtual environments and sandboxes. Virtual environment may be detected if such an artifact is present.

1. Check for specific global mutexes

This method checks for particular mutexes which are present in virtual environments but not in usual host systems.

Functions used:

- CreateMutexA/W

- OpenMutexA/W

Code sample

```
// usage sample:
supMutexExist(L"Sandboxie_SingleInstanceMutex_Control")
; // sample value from the table below


BOOL supMutexExist(_In_ LPWSTR lpMutexName)
{
    DWORD dwError;
    HANDLE hObject = NULL;
    if (lpMutexName == NULL) {
        return FALSE;
    }

    SetLastError(0);
    hObject = CreateMutex(NULL, FALSE, lpMutexName); /
/ define around A or W function version
    dwError = GetLastError();

    if (hObject) {
        CloseHandle(hObject);
    }

    return (dwError == ERROR_ALREADY_EXISTS);
}
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Signature recommendations

If the following function contains 3rd argument from the table column `Name`:

- CreateMutexA/W(..., ..., registry_path)
- OpenMutexA/W(..., ..., registry_path)

then it’s an indication of application trying to use the evasion technique.

Detections table

Check if the following global mutexes exist:	
Detect	**Name**
DeepFreeze	Frz_State
Sandboxie	Sandboxie_SingleInstanceMutex_Control
	SBIE_BOXED_ServiceInitComplete_Mutex1
VirtualPC	MicrosoftVirtualPC7UserServiceMakeSureWe'reTheOnlyOneMutex

Note: DeepFreeze is an application restoring the system on each reboot.

2. Check for specific virtual devices

This method checks for particular virtual devices which are present in virtual environments but not in usual host systems.

Function used:

- NtCreateFile

Code sample

```
// usage sample:
HANDLE hDummy = NULL;
supOpenDevice(L"\\Device\\Null", GENERIC_READ,
&hDummy); // sample values from the table below


BOOL supOpenDevice(
    _In_ LPWSTR lpDeviceName,
    _In_ ACCESS_MASK DesiredAccess,
    _Out_opt_ PHANDLE phDevice)
{
    OBJECT_ATTRIBUTES attr;
    IO_STATUS_BLOCK iost;
    UNICODE_STRING uDevName;
    HANDLE hDevice;
    NTSTATUS Status;

    if (phDevice) {
        *phDevice = NULL;
    }
    if (lpDeviceName == NULL) {
        return FALSE;
    }

    hDevice = NULL;
    RtlSecureZeroMemory(&uDevName, sizeof(uDevName));
    RtlInitUnicodeString(&uDevName, lpDeviceName);
    InitializeObjectAttributes(&attr, &uDevName, OBJ_C
ASE_INSENSITIVE, 0, NULL);

    Status = NtCreateFile(&hDevice, DesiredAccess, &at
tr, &iost, NULL, 0,
        0, FILE_OPEN, 0, NULL, 0);
    if (NT_SUCCESS(Status)) {
        if (phDevice != NULL) {
            *phDevice = hDevice;
        }
    }
```

```
    return NT_SUCCESS(Status);
}
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Signature recommendations

If the following function contains 3rd argument with its field `ObjectName->Buffer` from the table column `Name`:

- NtCreateFile(..., ..., attr, ...)

then it's an indication of application trying to use the evasion technique.

3rd argument is of the following type:

```
typedef struct _OBJECT_ATTRIBUTES {
    ULONG Length;
    HANDLE RootDirectory;
    PUNICODE_STRING ObjectName;
    ULONG Attributes;
    PVOID SecurityDescriptor;
    PVOID SecurityQualityOfService;
} OBJECT_ATTRIBUTES;
```

Detections table

Check if the following virtual devices exist:	
Detect	**Path**
VirtualBox	\\.\VBoxMiniRdDN
	\\.\VBoxMiniRdrDN
	\\.\VBoxGuest
	\\.\VBoxTrayIPC
	\\.\VBoxMouse
	\\.\VBoxVideo
VMware	\\.\HGFS

	\\.\vmci

3. Check for specific global pipes

Pipes are just a particular case of virtual devices, please refer to the previous section for code sample and signature recommendations.

Detections table

Check if the following global pipes exist:	
Detect	**String**
VirtualBox	\\.\pipe\VBoxMiniRdDN
	\\.\pipe\VBoxTrayIPC

4. Check for global objects

This method checks for particular global objects which are present in virtual environments but not in usual host systems.

Functions used:

- NtOpenDirectoryObject
- NtQueryDirectoryObject

Code sample

```
// usage sample:
supIsObjectExists(L"\\Driver", L"SbieDrv"); // sample
values from the table below


typedef struct _OBJECT_DIRECTORY_INFORMATION {
    UNICODE_STRING Name;
    UNICODE_STRING TypeName;
} OBJECT_DIRECTORY_INFORMATION, *POBJECT_DIRECTORY_INF
ORMATION;

BOOL supIsObjectExists(
    _In_ LPWSTR RootDirectory,
    _In_ LPWSTR ObjectName)
{
    OBJSCANPARAM Param;
    if (ObjectName == NULL) {
        return FALSE;
    }

    Param.Buffer = ObjectName;
    Param.BufferSize = (ULONG)_strlen_w(ObjectName);

    return NT_SUCCESS(supEnumSystemObjects(RootDirecto
ry, NULL, supDetectObjectCallback, &Param));
}

NTSTATUS NTAPI supDetectObjectCallback(
    _In_ POBJECT_DIRECTORY_INFORMATION Entry,
    _In_ PVOID CallbackParam)
{
    POBJSCANPARAM Param =
(POBJSCANPARAM)CallbackParam;
    if (Entry == NULL) {
        return STATUS_INVALID_PARAMETER_1;
    }
    if (CallbackParam == NULL) {
        return STATUS_INVALID_PARAMETER_2;
```

```
    }
    if (Param->Buffer == NULL || Param->BufferSize ==
0) {
        return STATUS_MEMORY_NOT_ALLOCATED;
    }
    if (Entry->Name.Buffer) {
        if (_strcmpi_w(Entry->Name.Buffer, Param->Buff
er) == 0) {
            return STATUS_SUCCESS;
        }
    }

    return STATUS_UNSUCCESSFUL;
}

NTSTATUS NTAPI supEnumSystemObjects(
    _In_opt_ LPWSTR pwszRootDirectory,
    _In_opt_ HANDLE hRootDirectory,
    _In_ PENUMOBJECTSCALLBACK CallbackProc,
    _In_opt_ PVOID CallbackParam)
{
    BOOL cond = TRUE;
    ULONG ctx, rlen;
    HANDLE hDirectory = NULL;
    NTSTATUS status;
    NTSTATUS CallbackStatus;
    OBJECT_ATTRIBUTES attr;
    UNICODE_STRING sname;
    POBJECT_DIRECTORY_INFORMATION objinf;

    if (CallbackProc == NULL) {
        return STATUS_INVALID_PARAMETER_4;
    }
    status = STATUS_UNSUCCESSFUL;

    __try {
        // We can use root directory.
        if (pwszRootDirectory != NULL) {
            RtlSecureZeroMemory(&sname, sizeof(sname))
;
```

```
            RtlInitUnicodeString(&sname, pwszRootDirec
tory);
            InitializeObjectAttributes(&attr, &sname,
OBJ_CASE_INSENSITIVE, NULL, NULL);

            status =
NtOpenDirectoryObject(&hDirectory, DIRECTORY_QUERY, &a
ttr);
            if (!NT_SUCCESS(status)) {
                return status;
            }
        }
        else {
            if (hRootDirectory == NULL) {
                return STATUS_INVALID_PARAMETER_2;
            }
            hDirectory = hRootDirectory;
        }

        // Enumerate objects in directory.
        ctx = 0;
        do {
            rlen = 0;
            status =
NtQueryDirectoryObject(hDirectory, NULL, 0, TRUE, FALS
E, &ctx, &rlen);
            if (status != STATUS_BUFFER_TOO_SMALL)
                    break;
            objinf = HeapAlloc(GetProcessHeap(), HEAP_
ZERO_MEMORY, rlen);
            if (objinf == NULL)
                break;

            status =
NtQueryDirectoryObject(hDirectory, objinf, rlen,
TRUE, FALSE, &ctx, &rlen);
            if (!NT_SUCCESS(status)) {
                HeapFree(GetProcessHeap(), 0, objinf);
                break;
            }
```

```
            CallbackStatus = CallbackProc(objinf, Call
  backParam);
            HeapFree(GetProcessHeap(), 0, objinf);
            if (NT_SUCCESS(CallbackStatus)) {
                status = STATUS_SUCCESS;
                break;
            }
        } while (cond);

        if (hDirectory != NULL) {
            NtClose(hDirectory);
        }
    }
    __except (EXCEPTION_EXECUTE_HANDLER) {
        status = STATUS_ACCESS_VIOLATION;
    }

    return status;
}
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Detections table

Check if the following global objects exist:		
Detect	**Path**	**Object**
Hyper-V	VmGenerationCounter	\Device
Parallels	prl_pv	\Device
	prl_tg	\Device
	prl_time	\Device
Sandboxie	SandboxieDriverApi	\Device
	SbieDrv	\Driver
	SbieSvcPort	\RPC Control
VirtualBox	VBoxGuest	\Device
	VBoxMiniRdr	\Device
	VBoxVideo	\Driver
	VBoxMouse	\Driver
VirtualPC	VirtualMachineServices	\Device
	1-driver-vmsrvc	\Driver

VMware	vmmemctl	\Device

5. Check for object directory (Sandboxie only)

This method checks for particular object directory which is present in Sandboxie virtual environment but not in usual host systems.

Function used:

- GetFileAttributes

Code sample

```
#define DIRECTORY_QUERY (0x0001)
#define OBJ_CASE_INSENSITIVE 0x00000040L
#define DIRECTORY_SANDBOXIE L"\\Sandbox"

int check_if_obj_dir_present() {
    OBJECT_ATTRIBUTES attr;
    UNICODE_STRING ustrName;
    HANDLE hObject = NULL;

    RtlSecureZeroMemory(&ustrName, sizeof(ustrName));
    RtlInitUnicodeString(&ustrName, DIRECTORY_SANDBOXI
E);
    InitializeObjectAttributes(&attr, &ustrName, OBJ_C
ASE_INSENSITIVE, NULL, NULL);

    if (NT_SUCCESS(NtOpenDirectoryObject(&hObject, DIR
ECTORY_QUERY, &attr))) {
        NtClose(hObject);
        return TRUE;
    }

    return FALSE;
}
```

Credits for this code sample: VMDE project (https://github.com/hfiref0x/VMDE)

Signature recommendations

If the following function contains 3rd argument with its field "ObjectName->Buffer" from the table column `Name`:

- NtOpenDirectoryObject(..., ..., attr, ...)

then it's an indication of application trying to use the evasion technique.

3rd argument is of the following type:

```
typedef struct _OBJECT_ATTRIBUTES {
    ULONG Length;
    HANDLE RootDirectory;
    PUNICODE_STRING ObjectName;
    ULONG Attributes;
    PVOID SecurityDescriptor;
    PVOID SecurityQualityOfService;
} OBJECT_ATTRIBUTES;
```

Detections table

Check if the following object directory exists:	
Detect	**Path**
Sandboxie	\Sandbox

6. Check if virtual registry is present in OS (Sandboxie only)

This method checks for virtual registry which is present in Sandboxie virtual environment but not in usual host systems.

Application opens registry key \REGISTRY\USER. It uses the following function in order to check real object name:

```
NtQueryObject(
    hUserKey,
    ObjectNameInformation,
    oni, // OBJECT_NAME_INFORMATION object
    Size,
    NULL);
```

If received OBJECT_NAME_INFORMATION object name does not equal to the "\REGISTRY\USER", then application assumes that it runs inside Sandboxie environment.

Signature recommendations

If the following function is used for opening \REGISTRY\USER:

- NtOpenKey

and is followed by the call of the following function with its 1st argument being the handle of \REGISTRY\USER key:

- NtQueryObject(hUserKey, ...)

then it's an indication of application trying to use the evasion technique.

Countermeasures

Hook target functions and return appropriate results if indicators (objects from tables) are triggered. In some cases stopping appropriate device may help – but it's not a universal counter-action: not all global objects are devices.

Credits

Credits go to open-source project from where code samples were taken:

- VMDE project on github (https://github.com/hfiref0x/VMDE)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions: Hardware

Go back (..)

Contents

Hardware info detection methods

Virtual environments emulate hardware devices and leave specific traces in their descriptions - which may be queried and the conclusion about non-host OS made.

1. Check if HDD has specific name

Functions used:

- SetupDiGetClassDevs
- SetupDiEnumDeviceInfo
- SetupDiGetDeviceRegistryProperty

Code sample

```
hDevs = SetupDiGetClassDevs(
    &guid,  // GUID_DEVCLASS(DEVINTERFACE)_DISKDRIVE
    NULL,
    NULL,
    DIGCF_PRESENT);

SetupDiEnumDeviceInfo(
    hDevsInfo,
    0,
    &devinfo);  // PSP_DEVINFO_DATA

SetupDiGetDeviceRegistryProperty(
    hDevs,
    &devinfo,
    SPDRP_FRIENDLYNAME,
    &dword_1,
    szFriendlyName,  // HDD name will be here
    dFriendlyNameSize,
    &dword_2);
```

Detections table

Check if hard disk drive has one of the following names:	
Detect	**Name**
QEMU	QEMU
VirtualBox	VBOX
VirtualPC	VIRTUAL HD
VMware	VMware

2. Check if HDD Vendor ID has specific value

The following function is used:

- DeviceIoControl(..., IOCTL_STORAGE_QUERY_PROPERTY, ...)

```
Code sample
```

```
bool GetHDDVendorId(std::string& outVendorId) {
    HANDLE hDevice = CreateFileA(_T("\\\\.\\PhysicalDr
ive0"),
                                 0,
                                 FILE_SHARE_READ | FIL
E_SHARE_WRITE,
                                 0,
                                 OPEN_EXISTING,
                                 0,
                                 0);
    if (hDevice == INVALID_HANDLE_VALUE)
        return false;

    STORAGE_PROPERTY_QUERY storage_property_query = {}
;
    storage_property_query.PropertyId = StorageDeviceP
roperty;
    storage_property_query.QueryType = PropertyStandar
dQuery;
    STORAGE_DESCRIPTOR_HEADER storage_descriptor_heade
r = {};
    DWORD BytesReturned = 0;

    if (!DeviceIoControl(hDevice, IOCTL_STORAGE_QUERY_
PROPERTY,
                         &storage_property_query, size
of(storage_property_query),
                         &storage_descriptor_header, s
izeof(storage_descriptor_header),
                         &BytesReturned, )) {
        printf("DeviceIoControl() for size query
failed\n");
        CloseHandle(hDevice);
        return false;
    }
    if (!BytesReturned) {
        CloseHandle(hDevice);
        return false;
```

```
    }

    std::vector<char> buff(storage_descriptor_header.S
ize); //_STORAGE_DEVICE_DESCRIPTOR
    if (!DeviceIoControl(hDevice, IOCTL_STORAGE_QUERY_
PROPERTY,
                         &storage_property_query, size
of(storage_property_query),
                         buff.data(), buff.size(),
0)) {
        CloseHandle(hDevice);
        return false;
    }

    CloseHandle(hDevice);

    if (BytesReturned) {
        STORAGE_DEVICE_DESCRIPTOR* device_descriptor
= (STORAGE_DEVICE_DESCRIPTOR*)buff.data();
        if (device_descriptor->VendorIdOffset)
            outVendorId = &buff[device_descriptor->Ven
dorIdOffset];

        return true;
    }

    return false;
}
```

Detections table

Check if HDD Vendor ID is one of the following:	
Detect	**Name**
VirtualBox	VBOX
VMware	vmware

3. Check if audio device is absent

This technique was extracted from TeslaCrypt malware sample and was described in this Joe Security blog post (https://www.joesecurity.org/blog/6933341622592617830).

Code sample

```
void AudioEvasion() {
  PCWSTR wszfilterName = L"audio_device_random_name";

  if (FAILED(CoInitialize(NULL)))
    return;

  IGraphBuilder *pGraph = nullptr;
  if (FAILED(CoCreateInstance(CLSID_FilterGraph,
NULL, CLSCTX_INPROC_SERVER, IID_IGraphBuilder,
(void**)&pGraph)))
    return;

  if (E_POINTER != pGraph->AddFilter(NULL, wszfilterNa
me))
    ExitProcess(-1);

  IBaseFilter *pBaseFilter = nullptr;
  CoCreateInstance(CLSID_AudioRender, NULL, CLSCTX_INP
ROC_SERVER, IID_IBaseFilter, (void**)&pBaseFilter);

  pGraph->AddFilter(pBaseFilter, wszfilterName);

  IBaseFilter *pBaseFilter2 = nullptr;
  pGraph->FindFilterByName(wszfilterName, &pBaseFilter
2);
  if (nullptr == pBaseFilter2)
    ExitProcess(1);

  FILTER_INFO info = { 0 };
  pBaseFilter2->QueryFilterInfo(&info);
  if (0 != wcscmp(info.achName, wszfilterName))
    return;

  IReferenceClock *pClock = nullptr;
  if (0 != pBaseFilter2->GetSyncSource(&pClock))
    return;
  if (0 != pClock)
    return;
```

```
    CLSID clsID = { 0 };
    pBaseFilter2->GetClassID(&clsID);
    if (clsID.Data1 == 0)
      ExitProcess(1);

    if (nullptr == pBaseFilter2)
      ExitProcess(-1);

    IEnumPins *pEnum = nullptr;
    if (0 != pBaseFilter2->EnumPins(&pEnum))
      ExitProcess(-1);

    if (0 == pBaseFilter2->AddRef())
      ExitProcess(-1);
  }
```

4. Check if CPU temperature information is available

This technique was extracted from GravityRAT malware and is described by this link (https://blog.talosintelligence.com/2018/04/gravityrat-two-year-evolution-of-apt.html).

Code sample (Windows cmd command)

```
wmic /namespace:\\root\WMI path
MSAcpi_ThermalZoneTemperature get CurrentTemperature
```

5. Check physical display adapter for IDirect3D9 interface

This method checks physical display adapters present in the system when the IDirect3D9 interface was instantiated. It works on all Windows versions starting from Windows XP.

Functions used:

- Direct3DCreate9 - called from `d3d9.dll` library
- GetAdapterIdentifier - called via IDirect3D9 interface

Code sample

```
#include <d3d9.h>

// https://github.com/qt/qtbase/blob/dev/src/plugins/
platforms/windows/qwindowsopengltester.cpp#L124

void detect() {
    typedef IDirect3D9* (WINAPI* PtrDirect3DCreate9)(U
INT);

    HMODULE d3d9lib = ::LoadLibraryA("d3d9");
    if (!d3d9lib)
        return;

    PtrDirect3DCreate9 direct3DCreate9 = (PtrDirect3DC
reate9)GetProcAddress(d3d9lib, "Direct3DCreate9");
    if (!direct3DCreate9)
        return;

    IDirect3D9* direct3D9 = direct3DCreate9(D3D_SDK_VE
RSION);
    if (!direct3D9)
        return;

    D3DADAPTER_IDENTIFIER9 adapterIdentifier;
    const HRESULT hr = direct3D9-
>GetAdapterIdentifier(0, 0, &adapterIdentifier);
    direct3D9->Release();

    if (SUCCEEDED(hr)) {
        printf("VendorId:    0x%x\n", adapterIdentifie
r.VendorId);
        printf("DeviceId:    0x%x\n", adapterIdentifie
r.DeviceId);
        printf("Driver:      %s\n",
adapterIdentifier.Driver);
        printf("Description: %s\n",
adapterIdentifier.Description);
```

```
    }
  }
```

Credits for this code sample go to elsamuko (https://gist.github.com/elsamuko/d3049d52ca235112c99ac3ee30282846) who pointed it out.

Example of output on a usual host machine is provided below:

```
VendorId:    0x10de
DeviceId:    0x103c
Driver:      nvldumdx.dll
Description: NVIDIA Quadro K5200
```

And here is an example of output on a virtual machine (VMware):

```
VendorId:    0x15ad
DeviceId:    0x405
Driver:      vm3dum64_loader.dll
Description: VMware SVGA 3D
```

Examined fields are named after the corresponding fields of D3DADAPTER_IDENTIFIER9 structure. Malware can compare values in these fields to the ones which are known to be present inside the virtual machine and if match is found, then it draws the conclusion that it’s run under virtual machine.

Detections table

Check if the following values are present in the fields of D3DADAPTER_IDENTIFIER9 structure:			
Detect	**Structure field**	**Value**	**Comment**
VMware	VendorId	0x15AD	
	DeviceId	0x405	Only when used in combination with VendorId related to VMware (0x15AD)
	Driver	vm3dum.dll	

	Driver	vm3dum64_loader.dll	
	Description	VMware SVGA 3D	

Signature recommendations

Signature recommendations are general for each technique: hook the function used and track if it is called. It's pretty hard to tell why application wants to get HDD name, for example. It doesn't necessarily mean applying evasion technique. So the best what can be done in this situation is intercepting target functions and tracking their calls.

Countermeasures

- versus HDD checks: rename HDD so that it's not detected by specific strings;
- versus audio device check: add audio device;
- versus CPU temperature check: add stub to hypervisor to output some meaningful information;
- versus physical display adapter check: set up hook on a function GetAdapterIdentifier from d3d9.dll, check if the queried adapter is related to DirectX and replace return values.

Go back (..)

---

twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/hardware.html&text=Evasions:%20Hardware - Evasion Techniques) in (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/hardware.html&title=Evasions:%20Hardware)

Evasions: Hooks

Go back (..)

Contents

Hooks detection methods

Techniques described here make use of hooks either to detect user presence or as means to be checked whether some unusual-for-host-OS hooks installed.

1. Check whether hooks are set within system functions

Malware reads memory at specific addresses to check if Windows API functions are hooked.
This method is based on the fact, that emulation environments are most likely to hook these functions to be able to gather data and statistics during an emulation.

Popular functions to be checked:

- ReadFile
- DeleteFile
- CreateProcessA/W

Reading memory is accomplished via the following functions:

- ReadProcessMemory
- NtReadVirtualMemory

Then different algorithms may be used for checking:

- Comparing first two bytes with \x8B\xFF (mov edi, edi) – typical prologue start for kernel32 functions.
- Comparing first N bytes with \xCC - software breakpoint (int 3), not connected with hooks directly but still a suspicious behavior.
- Comparing first N bytes with \xE9 (call) or with \xEB (jmp instruction) – typical instructions for redirecting execution.
- Checking for push/ret combo for execution redirection.

and so on.

It’s pretty tricky to count for every possible comparison so general indication of something unusual in application’s behavior is reading memory where OS libraries reside. If to be more precise: reading memory where “interesting” functions are situated.

This atricle (https://0x00sec.org/t/defeating-userland-hooks-ft-bitdefender/12496) explains how to detect user-mode hooks and remove them. The following code samples are taken from the article.

Example of hook detection

```
HOOK_TYPE IsHooked(LPCVOID lpFuncAddress, DWORD_PTR *d
wAddressOffset) {
    LPCBYTE lpBytePtr = (LPCBYTE)lpFuncAddress;

    if (lpBytePtr[0] == 0xE9) {
        *dwAddressOffset = 1;
        return HOOK_RELATIVE;    // E9 jmp is
relative.
    } else if (lpBytePtr[0] == 0x68 &&  lpBytePtr[5] =
= 0xC3) {
        *dwAddressOffset = 1;
        return HOOK_ABOLSUTE;    // push/ret is
absolute.
    }

    return HOOK_NONE;            // No hook.
}

LPVOID lpFunction = ...;
DWORD_PTR dwOffset = 0;
LPVOID dwHookAddress = 0;

HOOK_TYPE ht = IsHooked(lpFunction, &dwOffset);
if (ht == HOOK_ABSOLUTE) {
    // 1. Get the pointer to the address (lpFunction +
dwOffset)
    // 2. Cast it to a DWORD pointer
    // 3. Dereference it to get the DWORD value
    // 4. Cast it to a pointer
    dwHookAddress = (LPVOID)(*(LPDWORD)((LPBYTE)lpFunc
tion + dwOffset));
} else if (ht == HOOK_RELATIVE) {
    // 1. Get the pointer to the address (lpFunction +
dwOffset)
    // 2. Cast it to an INT pointer
    // 3. Dereference it to get the INT value (this
can be negative)
    INT nJumpSize = (*(PINT)((LPBYTE)lpFunction  + dwO
```

```
ffset);
    // 4. E9 jmp starts from the address AFTER the jmp
instruction
    DWORD_PTR dwRelativeAddress = (DWORD_PTR)
((LPBYTE)lpFunction + dwOffset + 4));
    // 5. Add the relative address and jump size
    dwHookAddress = (LPVOID)(dwRelativeAddress + nJump
Size);
}
```

Example of unhooking functions

```
// Parse the PE headers.
PIMAGE_DOS_HEADER pidh = (PIMAGE_DOS_HEADER)lpMapping;
PIMAGE_NT_HEADERS pinh = (PIMAGE_NT_HEADERS)((DWORD_PT
R)lpMapping + pidh->e_lfanew);

// Walk the section headers and find the .text
section.
for (WORD i = 0; i < pinh-
>FileHeader.NumberOfSections; i++) {
    PIMAGE_SECTION_HEADER pish = (PIMAGE_SECTION_HEADE
R)((DWORD_PTR)IMAGE_FIRST_SECTION(pinh) +
                                ((DWORD_PTR)IMAGE_SIZ
EOF_SECTION_HEADER * i));
    if (!strcmp(pish->Name, ".text")) {
        // Deprotect the module's memory region for
write permissions.
        DWORD flProtect = ProtectMemory(
            (LPVOID)((DWORD_PTR)hModule + (DWORD_PTR)p
ish->VirtualAddress),    // Address to protect.
            pish-
>Misc.VirtualSize,                    // Size to
protect.

PAGE_EXECUTE_READWRITE                          //
Desired protection.
        );

        // Replace the hooked module's .text section
with the newly mapped module's.
        memcpy(
            (LPVOID)((DWORD_PTR)hModule + (DWORD_PTR)p
ish->VirtualAddress),
            (LPVOID)((DWORD_PTR)lpMapping +
(DWORD_PTR)pish->VirtualAddress),
            pish->Misc.VirtualSize
        );

        // Reprotect the module's memory region.
```

```
            flProtect = ProtectMemory(
                (LPVOID)((DWORD_PTR)hModule + (DWORD_PTR)p
    ish->VirtualAddress),    // Address to protect.
                pish-
    >Misc.VirtualSize,                          // Size to
    protect.

    flProtect                                       //
    Revert to old protection.
            );
        }
    }
```

2. Check user clicks via mouse hooks

This technique is described by this link (https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/pf/file/fireeye-hot-knives-through-butter.pdf) (p.4, p. 7).

Malware sets mouse hook to detect a click (or more) if it occurs. If it's the case malware treats the host a usual one, i.e., with end user behind the screen - not a virtual environment. If no mouse click is detected then it's very likely a virtual environment.

Functions used:

- SetWindowsHookExA/W (WH_MOUSE_LL, ...)
- GetAsyncKeyState

Code sample (**SetWindowsHookExA**)

```
HHOOK g_hhkMouseHook = NULL;

LRESULT CALLBACK mouseHookProc(int nCode, WPARAM wPara
m, LPARAM lParam)
{
  switch (wParam)
  {
  case WM_MOUSEMOVE:
    // ...
    break;
  case WM_NCLBUTTONDOWN:
    // ...
    break;
  case WM_LBUTTONUP:
    UnhookWindowsHookEx(g_hhkMouseHook);
    CallMaliciousCode();
    ExitProcess(0);
  }
  return CallNextHookEx(g_hhkMouseHook, nCode,
wParam, lParam);
}

g_hhkMouseHook = SetWindowsHookEx(WH_MOUSE_LL, mouseHo
okProc, GetModuleHandleA(NULL), NULL);
```

Code sample (**GetAsyncKeyState**)

```
std::thread t([]()
{
  int count = 0;
  while (true)
  {
    if (GetAsyncKeyState(VK_LBUTTON) || GetAsyncKeySta
te(VK_RBUTTON) || GetAsyncKeyState(VK_MBUTTON))
    {
      if (++count == 2)
        break;
    }
    Sleep(100);
  }
  CallMaliciousCode();
});
t.join();
```

3. Check for incorrectly hooked functions

There are more than 400 Native API functions (or Nt-functions) in `ntdll.dll` that are usually hooked in sandboxes. In such a large list, there is enough space for different kinds of mistakes. We checked the hooked Nt-functions in popular sandboxes and found several issues. On of them is a lack of necessary checks for arguments in a hooked function. This case is described our article “Timing: Call a potentially hooked delay function with invalid arguments evasions (timing.html#call-hooked-function-with-invalid-arguments)”

Another issue we found is a discrepancy in the number of arguments in a hooked and an original function. If a function is hooked incorrectly, in kernel mode this may lead an operating system to crash. Incorrect user-mode hooks are not as critical. However, they may lead an analyzed application to crash or can be easily detected. For example, let’s look at the `NtLoadKeyEx` function. It was first

introduced in Windows Server 2003 and had only 4 arguments. Starting from Windows Vista up to the latest version of Windows 10, it has 8 arguments:

```
; Exported entry 318. NtLoadKeyEx
; Exported entry 1450. ZwLoadKeyEx
; __stdcall NtLoadKeyEx(x, x, x, x, x, x, x, x)
public _NtLoadKeyEx@32
```

However, in the Cuckoo monitor, the NtLoadKeyEx declaration still has only 4 arguments (https://github.com/cuckoosandbox/monitor/blob/8c419e6216f379e01ea0caa3a71142543e10fc04/sigs/registry_native.rst#ntloadkeyex):

```
*  POBJECT_ATTRIBUTES TargetKey
*  POBJECT_ATTRIBUTES SourceFile
** ULONG Flags flags
** HANDLE TrustClassKey trust_class_key
```

We found this legacy prototype used in other sources as well. For example, CAPE monitor (https://github.com/kevoreilly/capemon/blob/a3fe72ad9d3f9cd45aa2f5d503a5328ab1f9e442/hooks.h#L710) has the same issue:

```
extern HOOKDEF(NTSTATUS, WINAPI, NtLoadKeyEx,
    __in      POBJECT_ATTRIBUTES TargetKey,
    __in      POBJECT_ATTRIBUTES SourceFile,
    __in      ULONG Flags,
    __in_opt  HANDLE TrustClassKey
);
```

Therefore, if a sandbox uses any recent Windows OS, this function is hooked incorrectly. After the call to the incorrectly hooked function, the stack pointer value becomes

invalid. Therefore, a totally “legitimate” call to the RegLoadAppKeyW function, which calls NtLoadKeyEx, leads to an exception. This fact can be used to evade Cuckoo and CAPE sandbox with just a single call to the RegLoadAppKeyW function.

Code sample

```
RegLoadAppKeyW(L"storage.dat", &hKey, KEY_ALL_ACCESS,
0, 0);
// If the application is running in a sandbox an
exception will occur
// and the code below will not be executed.

// Some legitimate code that works with hKey to
distract attention goes here
// ...
RegCloseKey(hKey);
// Malicious code goes here
// ...
```

Instead of using RegLoadAppKeyW, we can call the NtLoadKeyEx function directly and check the ESP value after the call.

Code sample

```
__try
{
    _asm mov old_esp, esp
    NtLoadKeyEx(&TargetKey, &SourceFile, 0, 0, 0, KEY_
ALL_ACCESS, &hKey, &ioStatus);
    _asm mov new_esp, esp
    _asm mov esp, old_esp
    if (old_esp != new_esp)
        printf("Sandbox detected!");
}
__except (EXCEPTION_EXECUTE_HANDLER)
{
    printf("Sandbox detected!");
}
```

Signature recommendations

No signature recommendations are provided for this evasion group as it's hard to make a difference between the code which aims for some evasion technique and the one which is "legally used".

Countermeasures

- versus function hook checks: set kernel mode hooks; second solution is to use stack routing to implement function hooking;
- versus mouse click checks via hooks: use mouse movement emulation module.
- versus incorrect function hooks: ensure all the hooked function have the same number of arguments as the original functions

Credits

Credits go to user dtm from 0x00sec.org (https://0x00sec.org/) forum.

Due to modular code structure of the Check Point's tool called InviZzzible it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions: Human-like behavior

Go back (..)

Contents

Human-like behavior detection methods

All the techniques described in this group make use of the fact that certain actions are performed differently by a user and by a virtual environment.

1. General detection methods via registry

The registry is a storage for different pieces of information. For example, recently opened URLs and documents, and software installation notes are stored here. All of these may be used to determine if the machine is operated by a human user and is not a sandbox.

1.1. Check the number of recently opened documents

It's hard to imagine a typical host system where the user does not open any documents. Therefore, the lack of recently opened documents indicates this is likely a virtual environment.

Code sample (VB)

```
Public Function DKTxHE() As Boolean
DKTxHE = RecentFiles.Count < 3
End Function
```

This code sample was taken from SentinelOne article (https://www.sentinelone.com/blog/anti-vm-tricks/)

1.2. Check if the browser history contains at least 10 URLs

It's hard to imagine a typical host system where the user does not browse the Internet. Therefore, if there are fewer than 10 URLs in the browser history, this is likely a sandbox or VM.

Code sample (for Chrome)

```
bool chrome_history_evasion(int min_websites_visited
= 10)
{
  sqlite3 *db;
  int rc;
  bool vm_found = false;

  rc = sqlite3_open("C:\\Users\\<USER_NAME>\\AppData\
\Local\\Google\\Chrome\\User Data\\Default\\History",
&db);
  if (!rc)
  {
    char **results = nullptr;
    char *error = nullptr;
    int rows, columns;

    rc = sqlite3_get_table(db, "SELECT DISTINCT title
FROM urls;", &results, &rows, &columns, &error);
    if (!rc)
      vm_found = rows < min_websites_visited;
    sqlite3_free_table(results);
  }

  sqlite3_close(db);
  return vm_found;
}
```

1.3. Check if certain software packages were installed

If the system is only used for simulation purposes then it is likely to have many fewer installed software packages than a usual user's work machine. The installed packages may be specific to emulation purposes, not the ones that are usually used by human operators. Therefore, the list of installed packages may be compared to the list of commonly used applications to determine if it's a sandbox.

Code sample (PowerShell)

```
Get-ItemProperty HKLM:
\Software\Microsoft\Windows\CurrentVersion\Uninstall\*
| Format-Table -AutoSize | Measure-Object -Line
```

1.4. Countermeasures

Countermeasures are simple:

- open few documents to update the recent history
- open few internet URLs to create a browsing history
- install some lightweight software (like Notepad++)

2. Check for user presence at the moment of executing a process

The following sub-group leverages the differences between a user's interaction with the machine and the actions of a virtual environment.

2.1. Check the mouse movement

This method relies on the fact that a user frequently moves the mouse during actual work.

Some sandboxes and antivirus virtual machines have a static cursor position because they do not emulate any user activity while automatically running the files.

Code sample

```
int gensandbox_mouse_act() {
    POINT position1, position2;

    GetCursorPos(&position1);
    Sleep(2000);
    GetCursorPos(&position2);

    if ((position1.x == position2.x) && (position1.y =
= position2.y))
        // No mouse activity during the sleep.
        return TRUE;
    else
        return FALSE;
}
```

This code sample was taken from pafish project (https://github.com/a0rtega/pafish)

Such a short delay of only 2 seconds implies that the user should be active at the moment of infection.

More sophisticated checks rely on detection of not only the mouse movement per se but the pattern of such movement. The following example is taken from the research of LummaC2 Stealer (https://outpost24.com/blog/lummac2-anti-sandbox-technique-trigonometry-human-detection/) conducted by Outpost24.

First, malware captures mouse movements with the delay of 50 msec between them.

Second, the vectors are drawn out of paired captured positions.

Next, the angles are calculated between the corresponding vectors.

Finally, the angles are compared with the 45.0º threshold value, and if any of the angles is bigger than this hardcoded value, malware treats the result as being suspicious and does not execute the malicious code.

Countermeasures

Implement the module for mouse movement during a sample emulation. Make sure to come up with a more delicate way of interacting with the mouse cursor rather than just random movements all around the screen, so that it resembles the behavior of a human being.

2.2. Check via a request for user interaction

Some malware samples contain a GUI installer which requires user interaction. For example, the user must click the “Install” or “Next” buttons. Therefore, the malware may take no action unless the button is clicked. Standard sandboxes like Cuckoo have a module which simulates user activity. It searches for and clicks buttons with the captions mentioned above.

To prevent auto-clicking, a malware sample may create buttons with a class name that differs from “Button” or with a different caption (not “Install” or “Next”). This way the sandbox can’t detect and click the button.

Code sample

```
   // we use extended style flags to make a static
look like a button
   HWND hButton = CreateWindowExW(
       WS_EX_DLGMODALFRAME | WS_EX_WINDOWEDGE,  //
extended style flags
       TEXT("static"),         // class "static"
instead of "button"
       TEXT("Real next"),      // caption different
from “Install” or “Next”
       WS_VISIBLE | WS_CHILD | WS_GROUP |
SS_CENTER,  // usual style flags
       10, 10, 80, 25,         // arbitrary position
and size, may be any
       hWnd,                   // parent window
       NULL,                   // no menu
       NULL,                   // a handle to the
instance of the module to be associated with the
window
       NULL);                  // pointer to custom
value is not required
```

Countermeasures

Check for controls other than the buttons and examine their properties. For example, if the “Install” text is linked with the “static” control (not with “button”), this may indicate that the evasion technique is applied. Therefore, such a static control may be clicked.

2.3. Evasion technique for the Cuckoo human-interaction module

Suppose that the malware installer window has a button with the “Install” caption or something similar. It can be found by the human-interaction module of a sandbox but it’s invisible to an actual user (one-pixel size, hidden, etc.).

The real installation button has an empty or fake caption and the window class “Static”, so it can’t be detected by the auto-clicking module. In addition, the malware may take some mock action if the invisible button is clicked.

Code sample

```
    HWND hWnd = CreateWindow(
        TEXT("Button"),         // class "button"
        TEXT("Next"),           // caption is
“Install” or “Next”

NULL,                   // style flags are not
required, the control is invisible
        1, 1, 1, 1,             // the control is
created of 1x1 pixel size
        hParentWnd,             // parent window
        NULL,                   // no menu
        NULL,                   // a handle to the
instance of the module to be associated with the
window
        NULL);                  // pointer to custom
value is not required
```

Countermeasures

Check for controls other than buttons and examine their properties. If there is a button of 1x1 pixel size or the button is invisible, this may be an indication of evasion technique applied. Therefore, such a control should not be clicked.

2.4. No suspicious actions until a document is scrolled down

Malware payloads which reside in Office documents (namely, *.docm *.docx) don't do anything until the document is scrolled to a certain page (second, third, etc.). A human user usually scrolls through the document while a virtual environment will likely not perform this step.

Example from FireEye report (https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/pf/file/fireeye-hot-knives-through-butter.pdf) (p. 6-7):

RTF documents consist of normal text, control words, and groups. Microsoft's RTF specification includes a shape-drawing function, which in turn includes a series of properties using the following syntax:

`{\sp{\sn propertyName}{\sv propertyValueInformation}}`

In this code, `\sp` is the control word for the drawing property, `\sn` is the property name, and `\sv` contains information about the property value. The code snippet in the image below exploits a CVE-2010-3333 vulnerability (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3333) that occurs when using an invalid `\sv` value for the pFragments shape property:

A closer look at the exploit code, as shown in the next image, reveals a series of paragraph marks (`./par`) that appears before the exploit code:

The repeated paragraph marks push the exploit code to the second page of the RTF document. Therefore, the malicious code does not execute unless the document scrolls down to bring the exploit code up into the active window – more likely to be a deliberate act by a human user than simulated movement in a virtual machine.

When the RTF is scrolled down to the second page, only then is the exploit code triggered and the payload is downloaded.

In a sandbox, where any mouse activity is random or preprogrammed, the RTF document's second page never appears. Therefore, the malicious code never executes, and nothing seems amiss in the sandbox analysis.

Countermeasures

Find a window with the document and send the WM_VSCROLL message there. Alternatively, send the WM_MOUSEWHEEL message as described here (https://stackoverflow.com/questions/60203135/set-delta-in-a-wm-mousewheel-message-to-send-with-postmessage).

2.5. Check user activity via GetLastInputInfo

User activity can be checked with the call to the `GetLastInputInfo` function

Although Agent Tesla v3 performs this check, it does so incorrectly. Compare the code of Agent Tesla v3 with the correct technique implementation below.

Evasion technique as implemented in Agent Tesla v3. This function is called after a delay of 30 seconds.

As measured time values are in milliseconds, the difference between them cannot be larger than 30000 (30 seconds). This means that with division by 1000.0, the resulting value cannot be larger than 30. In turn, this indicates that a comparison with 600 always leads to a result in which the sandbox is undetected.

The correct implementation is provided below.

Code sample

```
    bool sandbox_detected = false;

    Sleep(30000);

    DWORD ticks = GetTickCount();

    LASTINPUTINFO li;
    li.cbSize = sizeof(LASTINPUTINFO);
    BOOL res = GetLastInputInfo(&li);

    if (ticks - li.dwTime > 6000)
    {
        sandbox_detected = true;
    }
```

Countermeasures

Implement the module for mouse movement during a sample emulation.

Countermeasures

Countermeasures for chapter 1 are given in the corresponding section. Countermeasures for chapter 2 are given in place in the appropriate sections.

Signature recommendations

Signature recommendations are not provided for this class of techniques as the methods described in this chapter do not imply their usage for evasion purposes. It is hard to differentiate between the code meant for evasion and code designed for non-evasion purposes.

Credits

Open-source project from where code samples were taken:

- pafish project on Github (https://github.com/a0rtega/pafish)

Companies from where certain examples were taken:

- FireEye (https://www.fireeye.com)
- SentinelOne (https://www.sentinelone.com/)

Go back (..)

Evasions: Network

Go back (..)

Contents

Network detection methods

Evasion techniques in this group are related to network in this or that sense. Either network-related functions are used or network parameters are checked — if they are different from that of usual host OS then virtual environment is likely detected.

1. Specific network properties

Vendors of different virtual environments hard-code some values (MAC address) and names (network adapter) for their products — due to this fact such environments may be detected via checking properties of appropriate objects.

1.1. Check if MAC address is specific

Functions used:

- GetAdaptersAddresses(AF_UNSPEC, ...)
- GetAdaptersInfo

Code sample (function **GetAdaptersAddresses**)

```
int pafish_check_mac_vendor(char * mac_vendor) {
    unsigned long alist_size = 0, ret;
    ret = GetAdaptersAddresses(AF_UNSPEC, 0, 0, 0, &al
ist_size);

    if (ret == ERROR_BUFFER_OVERFLOW) {
        IP_ADAPTER_ADDRESSES* palist = (IP_ADAPTER_ADD
RESSES*)LocalAlloc(LMEM_ZEROINIT,alist_size);
        void * palist_free = palist;

        if (palist) {
            GetAdaptersAddresses(AF_UNSPEC, 0, 0, pali
st, &alist_size);
            char mac[6]={0};
            while (palist){
                if (palist->PhysicalAddressLength == 0
x6) {
                    memcpy(mac, palist->PhysicalAddres
s, 0x6);
                    if (!memcmp(mac_vendor, mac, 3))
{  /* First 3 bytes are the same */
                        LocalFree(palist_free);
                        return TRUE;
                    }
                }
                palist = palist->Next;
            }
            LocalFree(palist_free);
        }
    }

    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Code sample (function **GetAdaptersInfo**)

```
BOOL check_mac_addr(TCHAR* szMac)
{
    BOOL bResult = FALSE;
    PIP_ADAPTER_INFO pAdapterInfo;
    ULONG ulOutBufLen = sizeof (IP_ADAPTER_INFO);
    pAdapterInfo = (PIP_ADAPTER_INFO) MALLOC(sizeof(IP
_ADAPTER_INFO));

    if (pAdapterInfo == NULL)
    {

_tprintf(_T("Error allocating memory needed to call
GetAdaptersinfo.\n"));
        return -1;
    }

    // Make an initial call to GetAdaptersInfo to get
the necessary size into the ulOutBufLen variable
    if (GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)
== ERROR_BUFFER_OVERFLOW)
    {
        FREE(pAdapterInfo);
        pAdapterInfo = (PIP_ADAPTER_INFO) MALLOC(ulOut
BufLen);
        if (pAdapterInfo == NULL) {
            printf("Error allocating memory needed to
call GetAdaptersinfo\n");
            return 1;
        }
    }

    // Now, we can call GetAdaptersInfo
    if (GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)
== ERROR_SUCCESS)
    {
        // Convert the given mac address to an array
of multibyte chars so we can compare.
        CHAR szMacMultiBytes [4];
```

```
            for (int i = 0; i < 4; i++) {
                szMacMultiBytes[i] = (CHAR)szMac[i];
            }
            while(pAdapterInfo)
            {
                if (pAdapterInfo->AddressLength == 6 && !m
  emcmp(szMacMultiBytes, pAdapterInfo->Address, 3))
                {
                    bResult = TRUE;
                    break;
                }
                pAdapterInfo = pAdapterInfo->Next;
            }
        }

        return bResult;
    }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Detections table

Check if MAC address starts from one of the following values:		
Detect	**MAC address starts with**	**Bytes**
Parallels	00:1C:42	\x00\x1C\x42
VirtualBox	08:00:27	\x08\x00\x27
VMware	00:05:69	\x00\x05\x69
	00:0C:29	\x00\x0C\x29
	00:1C:14	\x00\x1C\x14
	00:50:56	\x00\x50\x56
Xen	00:16:E3	\x00\x16\xE3

1.2. Check if adapter name is specific

Functions used:

- GetAdaptersAddresses(AF_UNSPEC, ...)
- GetAdaptersInfo

Code sample (function **GetAdaptersAddresses**)

```
int pafish_check_adapter_name(char * name) {
    unsigned long alist_size = 0, ret;
    wchar_t aux[1024];

    mbstowcs(aux, name, sizeof(aux)-sizeof(aux[0]));
    ret = GetAdaptersAddresses(AF_UNSPEC, 0, 0, 0, &al
ist_size);

    if (ret == ERROR_BUFFER_OVERFLOW) {
        IP_ADAPTER_ADDRESSES *palist = (IP_ADAPTER_ADD
RESSES *)LocalAlloc(LMEM_ZEROINIT, alist_size);
        void * palist_free = palist;
        if (palist) {
            if (GetAdaptersAddresses(AF_UNSPEC, 0, 0,
palist, &alist_size) == ERROR_SUCCESS) {
                while (palist) {
                    if (wcsstr(palist->Description, au
x)) {
                        LocalFree(palist_free);
                        return TRUE;
                    }
                    palist = palist->Next;
                }
            }
            LocalFree(palist_free);
        }
    }

    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Code sample (function **GetAdaptersInfo**)

```
BOOL check_adapter_name(TCHAR* szName)
{
    BOOL bResult = FALSE;
    PIP_ADAPTER_INFO pAdapterInfo;
    ULONG ulOutBufLen = sizeof(IP_ADAPTER_INFO);
    pAdapterInfo = (PIP_ADAPTER_INFO)MALLOC(sizeof(IP_
ADAPTER_INFO));

    if (pAdapterInfo == NULL)
    {

_tprintf(_T("Error allocating memory needed to call
GetAdaptersinfo.\n"));
        return -1;
    }

    // Make an initial call to GetAdaptersInfo to get
the necessary size into the ulOutBufLen variable
    if (GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)
== ERROR_BUFFER_OVERFLOW)
    {
        FREE(pAdapterInfo);
        pAdapterInfo = (PIP_ADAPTER_INFO)MALLOC(ulOutB
ufLen);
        if (pAdapterInfo == NULL) {
            printf("Error allocating memory needed to
call GetAdaptersinfo\n");
            return 1;
        }
    }

    if (GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)
== ERROR_SUCCESS)
    {
        while (pAdapterInfo)
        {
            if
(StrCmpI(ascii_to_wide_str(pAdapterInfo-
```

```
  >Description), szName) == 0)
                {
                    bResult = TRUE;
                    break;
                }
                pAdapterInfo = pAdapterInfo->Next;
            }
        }

        return bResult;
    }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Detections table

Check adapter name to be the following:	
Detect	**Name**
VMware	Vmware

1.3. Check if provider's name for network shares is specific

Functions used (see note about native functions):

- WNetGetProviderName(WNNC_NET_RDR2SAMPLE, ...)

Code sample

```
int vbox_network_share() {
    unsigned long pnsize = 0x1000;
    char provider[pnsize];

    int retv =
WNetGetProviderName(WNNC_NET_RDR2SAMPLE, provider, &pn
size);
    if (retv == NO_ERROR) {
        if (lstrcmpi(provider, "VirtualBox Shared
Folders") == 0)
            return TRUE;
        else
            return FALSE;
    }

    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Detections table

Check provider's name for network shares to be the following:	
Detect	**Name**
VirtualBox	VirtualBox Shared Folders

2. Check if network belongs to security perimeter

Malware makes a request to https[:]//www.maxmind.com/geoip/v2.1/city/me which normally requires some kind of authentication or API key. To get around this requirement, the malware makes the request look as if it's coming from the site itself by setting the HTTP Referrer to https[:]//www.maxmind.com/en/locate-my-ip-address and User-Agent to *Mozilla/5.0 (compatible; MSIE 10.0; Windows*

NT 6.1; Trident/6.0). This trick allows the sample to retrieve the information about IP address of the machine it's running on.

The response is returned in JSON format and contains information about the country, city, and, most importantly, the organization associated with the IP address. If some "bad" strings are found in the response, malware knows that it's launched inside some kind of a security perimeter/organization.

Examples

- anti VM tricks (https://www.sentinelone.com/blog/anti-vm-tricks/)
- malicious macros add sandbox evasion techniques to distribute new Dridex (https://www.proofpoint.com/us/threat-insight/post/malicious-macros-add-to-sandbox-evasion-techniques-to-distribute-new-dridex)
- malicious documents with macros (https://www.zscaler.com/blogs/research/malicious-documents-leveraging-new-anti-vm-anti-sandbox-techniques) evading automated analysis systems

"Bad strings" from malware sample (fixed capitalization):

```
Amazon
anonymous
BitDefender
BlackOakComputers
Blue Coat
BlueCoat
Cisco
cloud
Data Center
DataCenter
DataCentre
dedicated
ESET, Spol
FireEye
ForcePoint
Fortinet
Hetzner
hispeed.ch
hosted
Hosting
Iron Port
IronPort
LeaseWeb
MessageLabs
Microsoft
MimeCast
NForce
Ovh Sas
Palo Alto
ProofPoint
Rackspace
security
Server
Strong Technologies
Trend Micro
TrendMicro
TrustWave
```

VMVault
Zscaler

3. NetValidateName result based anti-emulation technique

Initially this technique was designed for bypassing AV detection. It's not an evasion technique itself – instead it abuses interesting side-effects after the function is called.

The main idea is to use the determined result of `NetValidateName` API function call with invalid argument as Server name (for example "123") for calculating jump address dynamically. This jump usually points into the middle of some instruction to bypass heuristic analysis of AV software. But this technique also has (at least) one side-effect.

If default NetBIOS settings are set in the operating system (NetBIOS over TCP/IP is enabled) the return code is always equal to `ERROR_BAD_NETPATH (0x35)`.
If NetBIOS over TCP/IP is switched off then return code is `ERROR_NETWORK_UNREACHABLE (0x4CF)`.

Thus jump address will be calculated incorrectly and it will lead the sample to crash. Therefore, this technique can be used to break emulation in sandboxes where NetBIOS over TCP/IP is switched off for preventing junk traffic generation by the OS.

Note: NetBIOS over TCP/IP is switched off not to generate additional network requests when resolving server IP via DNS. Switching this option off cancels lookup requests in local network.

Code sample (function **GetAdaptersAddresses**)

```
void EntryPoint(void)
{
    HANDLE NetApi32 = LoadLibraryW(L"netapi32.dll");
    TD_NetValidateName NetValidateName = (TD_NetValida
teName)GetProcAddress(NetApi32, "NetValidateName");
    DWORD Result = NetValidateName(L"123", L"", L"", L
"", 1);

    __asm
    {
        call dword ptr ds:[GetLastError]
        add eax, offset TrueEntryPoint
        sub eax, 0xCB  // ERROR_ENVVAR_NOT_FOUND
        call eax
    }
}
```

4. Cuckoo ResultServer connection based anti-emulation technique

This technique can be used for detecting Cuckoo Sandbox virtual environment. Malware enumerates all established outgoing TCP connections and checks if there is a connection to a specific TCP port (2042) that is used by the Cuckoo ResultServer.

Signature recommendations

Signature recommendations are general for each technique: hook the function used and track if it is called. It's pretty hard to tell why application wants to get adapter name, for example. It doesn't necessarily mean applying evasion technique. So the best what can be done in this situation is intercepting target functions and tracking their calls.

Countermeasures

- versus checking network parameters: change them for virtual environment;
- versus checking security perimeter: emulate network responses in an appropriate manner;
- versus NetValidateName result based technique: turn on NetBIOS over TCP/IP;
- versus Cuckoo ResultServer connection based technique: change ResultServer port in the Cuckoo configuration.

Credits

Credits go to open-source project from where code samples were taken:

- pafish project on github (https://github.com/a0rtega/pafish)
- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

---

network.html&title=Evasions:%20Network - Evasion Techniques) (http://news.ycombinator.com/submitlink?u=https://evasions.checkpoint.com/src/Evasions/techniques/network.html&t=Evasions:%20Network - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/network.html&text=Evasions:%20Network - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/network.html&title=Evasions:%20Network)

Evasions: OS features

Go back (..)

Contents

OS features detection methods

Evasions in this group use peculiarities of how OS work.

1. Checking debug privileges

If the malware is running under debugger or in a sandbox like Cuckoo its process token will have a debug privilege in the enabled state. It happens because this privilege is enabled in the parent process and inherited by the malware process.

The malware tries to open crucial system processes like `csrss.exe`, `smss.exe`, `lsass.exe` with `PROCESS_ALL_ACCESS` access right and then tries to terminate them. This will fail in a normal case when the malware is executed from the explorer or command line because even an Administrator user can't terminate those processes. But this will succeed if the process token has the debug privilege in the enabled state. Termination of crucial system process leads OS to crash into BSOD with an error `0x000000F4` so the emulation process will be aborted.

Functions to get snapshot of running processes:

- `CreateToolhelp32Snapshot`
- `psapi.EnumProcesses (WinXP, Vista)`
- `kernel32.EnumProcesses (Win7+)`

Function used to open the process:

- `OpenProcess(PROCESS_ALL_ACCESS, ..., pid) // track for PIDs of 'csrss.exe', 'smss.exe', 'lsass.exe'`

Code sample

```
/*
If we're being debugged and the process has
SeDebugPrivileges
privileges then OpenProcess call will be successful.
This requires administrator privilege!
In Windows XP, Vista and 7, calling OpenProcess with
PROCESS_ALL_ACCESS will fait even with
SeDebugPrivilege enabled,
That's why I used PROCESS_QUERY_LIMITED_INFORMATION
*/

DWORD GetCsrssProcessId()
{
  if (API::IsAvailable(API_IDENTIFIER::API_CsrGetProce
ssId))
  {
    auto CsrGetProcessId = static_cast<pCsrGetId>(API:
:GetAPI(API_IDENTIFIER::API_CsrGetProcessId));

    return CsrGetProcessId();
  }
  else
    return GetProcessIdFromName(_T("csrss.exe"));
}


BOOL CanOpenCsrss()
{
   HANDLE hCsrss = OpenProcess(PROCESS_QUERY_LIMITED_I
NFORMATION, FALSE, GetCsrssProcessId());
   if (hCsrss != NULL)
  {
    CloseHandle(hCsrss);
    return TRUE;
  }
  else
```

```
        return FALSE;
    }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Signature recommendations

If OpenProcess requests all the possible rights when opening one of the critical system processes – it's a strong indicator of malware trying to apply this evasion technique.

2. Using unbalanced stack

This technique was presented at Virus Bulletin 2016 by Check Point Malware Reverse Engineering Team. It is described by this link (https://www.virusbulletin.com/uploads/pdf/magazine/2016/VB2016-Chailytko-Skuratovich.pdf).

To track process behaviour, the CuckooMon/Cuckoo Monitor module hooks relevant functions. In this type of architecture, the hook is called before the original function. A hooked function may use some space on the stack in addition to that used by the original function. Therefore, the total space on the stack used by the hooked function may be larger than the space used only by the original function.

Problem: The malware has information about how much space the called function uses on the stack. It can therefore move the stack pointer towards lower addresses at an offset that is sufficient to store the function arguments, local variables and return address to reserve space for them. The malware fills the space below the stack pointer with some relevant data. It then moves the stack pointer to the original location and calls the library function. If the function is not hooked, the malware fills in the reserved space before the relevant data (see Figure 1). If the function is hooked, the malware overlaps relevant data,

because the space that was reserved for the original function's local variables is smaller than the space occupied by the hook and the original function's local variables combined. The relevant data is therefore corrupted (see Figure 2). If it stores pointers to some functions that are used later during the execution process, the malware jumps to arbitrary code, occasionally crashing the application.

Stack on non-hooked and on hooked function call.

Solution: To avoid this behaviour, the Cuckoo Monitor/ CuckooMon module can use a two-stage hooking process. In the fi rst stage, instead of the hook's code execution, it can move the stack pointer towards lower addresses of a specifi c size that will be enough for the malware's relevant data. Then, the function's arguments are copied under the new stack pointer. Only after these preparatory operations have been completed is the second stage hook (which performs the real hooking) called. Relevant data fi lled in by the malware resides on upper stack addresses, thus it is not affected in any way by the called function.

Code sample

```
bool Cuckoo::CheckUnbalancedStack() const {
  usf_t f = {
    { lib_name_t(L"ntdll"), {
      {sizeof(void *), NULL, "ZwDelayExecution", ARG_I
TEM(kZwDelayExecutionArgs) }
    } }
  };
  const uint8_t canary[8] = { 0xDE, 0xAD, 0xBE, 0xEF,
0xDE, 0xAD, 0xBE, 0xEF };

  uint32_t args_size;
  const void *args_buff;
  uint32_t reserved_size;
  uint32_t reserved_size_after_call;
  uint32_t canary_size;
  FARPROC func;
  bool us_detected;
  void *canary_addr = (void *)&canary[0];

  static_assert((sizeof(canary) % sizeof(void *)) ==
0, "Invalid canary alignement");

  for (auto it = f.begin(), end = f.end(); it != end;
++it) {
    for (auto &vi : it->second) {
      vi.func_addr = GetProcAddress(GetModuleHandleW(i
t->first.c_str()), vi.func_name.c_str());

      // call to Unbalanced Stack
      args_size = vi.args_size;
      args_buff = vi.args_buff;
      canary_size = sizeof(canary);
      reserved_size = sizeof(void *) + vi.local_vars_s
ize + canary_size;
      reserved_size_after_call = reserved_size + args_
size;
      func = vi.func_addr;
      us_detected = false;
```

```
        __asm {
          pusha
          mov ecx, args_size
          sub esp, ecx
          mov esi, args_buff
          mov edi, esp
          cld
          rep movsb
          sub esp, reserved_size
          mov ecx, canary_size
          mov esi, canary_addr
          mov edi, esp
          rep movsb
          add esp, reserved_size
          mov eax, func
          call eax
          sub esp, reserved_size_after_call
          mov ecx, canary_size
          mov esi, canary_addr
          mov edi, esp
          repz cmpsb
          cmp ecx, 0
          setnz us_detected
          add esp, reserved_size_after_call
          popa
        }

        if (us_detected)
          return true;
      }
    }

    return false;
  }
```

Signature recommendations

Signature recommendations are not provided as it's pretty tricky to track such a behavior on malware side.

3. Detect Wine

The `MulDiv` API (https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-muldiv) is being called with specific arguments (`MulDiv(1, 0x80000000, 0x80000000)`) which should logically return 1 - however, due to a bug with the ancient implementation on Windows, it returns 2.

There are more known evasion methods to detect Wine like the good old check of searching for the existence of one of Wine's exclusive APIs such as `kernel32.dll!wine_get_unix_file_name` or `ntdll.dll!wine_get_host_version`) as also mentioned in Processes evasion techniques (https://evasions.checkpoint.com/src/Evasions/techniques/processes.html#check-if-specific-functions-are-present-in-specific-libraries).

Code sample

```
int Check_MulDiv_1() {
    // Call MulDiv with specific arguments
    int result = MulDiv(1, 0x80000000, 0x80000000);

    // Check if the result matches the expected value
    if (result != 2) {
        std::cout << "MulDiv evasion method detected:
Wine environment." << std::endl;
    } else {
        std::cout << "MulDiv evasion method not
detected." << std::endl;
    }

    return 0;
}

int Check_MulDiv_2() {
    // Check for the existence of Wine's exclusive
APIs
    HMODULE hKernel32 =
GetModuleHandle("kernel32.dll");
    FARPROC wineGetUnixFileName = GetProcAddress(hKern
el32, "wine_get_unix_file_name");
    HMODULE hNtdll = GetModuleHandle("ntdll.dll");
    FARPROC wineGetHostVersion =
GetProcAddress(hNtdll, "wine_get_host_version");

    if (wineGetUnixFileName || wineGetHostVersion) {
        std::cout << "Wine's exclusive APIs detected:
Wine environment." << std::endl;
    } else {
        std::cout << "Wine's exclusive APIs not
detected." << std::endl;
    }

    return 0;
}
```

Signature recommendations

Check if `MulDiv(1, 0x80000000, 0x80000000)` is being called.

Countermeasures

- versus checking debug privileges: hook OpenProcess and track for critical system processes PIDs – then return an error.
- versus using unbalanced stack: 1) stack adjusting before function call; 2) kernel-mode hooking.
- versus Detect Wine: If Using Wine, hook MulDiv to return 2 or modify the implementation as it works in Windows.

Credits

Credits go to open-source project from where code samples were taken:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

(http://www.reddit.com/submit?url=https://evasions.checkpoint.com/src/Evasions/techniques/os-features.html&title=Evasions:%20OS%20features - Evasion Techniques) (http://news.ycombinator.com/submitlink?u=https://evasions.checkpoint.com/src/Evasions/techniques/os-features.html&t=Evasions:%20OS%20features - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/os-features.html&text=Evasions:%20OS%20features - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/os-features.html&title=Evasions:%20OS%20features)

Evasions: Processes

Go back (..)

Contents

Processes and libraries detection methods

Virtual environment launches some specific helper processes which are not being executed in usual host OS. There are also some specific modules which are loaded into processes address spaces.

1. Check specific running processes and loaded libraries

1.1. Check if specific processes are running

Functions used:

- CreateToolhelp32Snapshot
- psapi.EnumProcesses *(WinXP, Vista)*
- kernel32.EnumProcesses *(Win7+)*

Code sample

```
check_process_is_running("vmtoolsd.exe");  // sample
value from the table

bool check_process_is_running(const std::string &proc_
name) {
    HANDLE hSnapshot;
    PROCESSENTRY32 pe = {};

    pe.dwSize = sizeof(pe);
    bool present = false;
    hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPR
OCESS, 0);

    if (hSnapshot == INVALID_HANDLE_VALUE)
        return false;

    if (Process32First(hSnapshot, &pe)) {
        do {
            if (!StrCmpI(pe.szExeFile,
proc_name.c_str())) {
                present = true;
                break;
            }
        } while (Process32Next(hSnapshot, &pe));
    }
    CloseHandle(hSnapshot);

    return present;
}
```

Signature recommendations

Signature recommendations are not provided as it’s hard to say what exactly is queried in the processes’ snapshot.

Detections table

Check if the following processes are running:

Detect	Process
JoeBox	joeboxserver.exe
	joeboxcontrol.exe
Parallels	prl_cc.exe
	prl_tools.exe
VirtualBox	vboxservice.exe
	vboxtray.exe
VirtualPC	vmsrvc.exe
	vmusrvc.exe
VMware	vmtoolsd.exe
	vmacthlp.exe
	vmwaretray.exe
	vmwareuser.exe
	vmware.exe
	vmount2.exe
	vmwareservice.exe
Xen	xenservice.exe
	xsvc_depriv.exe
QEMU	qemu-ga.exe
WPE Pro	WPE Pro.exe
KsDumper	ksdumperclient.exe

Notes:

- *WPE Pro is a sniffer, not a VM or a sandbox, however it is used along with VM detects.*
- *KsDumper is a kernel-mode process dumper, not a VM or a sandbox, however it is used along with VM detects in Styx Stealer.*

1.2. Check if specific libraries are loaded in the process address space

Functions used:

- GetModuleHandle

Code sample

```
VOID loaded_dlls()
{
    /* Some vars */
    HMODULE hDll;

    /* Array of strings of blacklisted dlls */
    TCHAR* szDlls[] = {
        _T("sbiedll.dll"),
        _T("dbghelp.dll"),
        _T("api_log.dll"),
        _T("dir_watch.dll"),
        _T("pstorec.dll"),
        _T("vmcheck.dll"),
        _T("wpespy.dll"),
    };

    WORD dwlength = sizeof(szDlls) / sizeof(szDlls[0])
;
    for (int i = 0; i < dwlength; i++)
    {
        TCHAR msg[256] = _T("");
        _stprintf_s(msg, sizeof(msg) / sizeof(TCHAR),
_T("Checking if process loaded modules contains: %s ")
,
                    szDlls[i]);

        /* Check if process loaded modules contains
the blacklisted dll */
        hDll = GetModuleHandle(szDlls[i]);
        if (hDll == NULL)
            print_results(FALSE, msg);
        else
            print_results(TRUE, msg);
    }
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Signature recommendations

If the following function contains its only argument from the table column `Library`:

- GetModuleHandle(module_name)

then it's an indication of application trying to use this evasion technique.

Detections table

Check if the following libraries are loaded in the process address space:	
Detect	**Library**
CWSandbox	api_log.dll
	dir_watch.dll
	pstorec.dll
Sandboxie	sbiedll.dll
ThreatExpert	dbghelp.dll
VirtualPC	vmcheck.dll
WPE Pro	wpespy.dll

Note: WPE Pro is a sniffer, not VM, however it is used along with VM detects.

1.3. Check if specific functions are present in specific libraries

Functions used (see note about native functions):

- kernel32.GetProcAddress
- kernel32.LdrGetProcedureAddress *(called internally)*
- ntdll.LdrGetProcedureAddress
- ntdll.LdrpGetProcedureAddress *(called internally)*

Code sample

```
BOOL wine_exports()
{
    /* Some vars */
    HMODULE hKernel32;

    /* Get kernel32 module handle */
    hKernel32 = GetModuleHandle(_T("kernel32.dll"));
    if (hKernel32 == NULL) {
        print_last_error(_T("GetModuleHandle"));
        return FALSE;
    }

    /* Check if wine_get_unix_file_name is exported by
this dll */
    if (GetProcAddress(hKernel32, "wine_get_unix_file_
name") == NULL)  // sample value from the table
        return FALSE;
    else
        return TRUE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Signature recommendations

If the following functions contain 2nd argument from the table column “Function” and the 1st argument is the address of matching “Library” name from the table:

- kernel32.GetProcAddress(lib_handle, func_name)
- kernel32.LdrGetProcedureAddress(lib_handle, func_name)
- ntdll.LdrGetProcedureAddress(lib_handle, func_name)
- ntdll.LdrpGetProcedureAddress(lib_handle, func_name)

then it’s an indication of application trying to use this evasion technique.

Detections table

Check if the following functions are present in the following libraries:		
Detect	**Library**	**Function**
Wine	kernel32.dll	wine_get_unix_file_name
	ntdll.dll	wine_get_version
		wine_get_host_version

1.4. Check if certain libraries can be loaded and others

Function used:

- LoadLibraryA/W

This technique relies on the assumption that there are some common system libraries in the usual system that can be loaded – and there are also some fake ones, that should not be really present in a usual system. However, in a sandbox, when trying to load some fake libraries, they may be reported as loaded - which is different from how it should be on a usual host.

In other words, if a system library that is usually present (but not so widely used) in non-emulated machines is not loaded, then the application is likely in a sandbox. And if a fake DLL is reported to be loaded, then it is likely a sandbox, as such DLL will not be loaded in a usual machine.

Code sample

```
bool Generic::CheckLoadedDLLs() const {
    std::vector<std::string> real_dlls = {
        "kernel32.dll",
        "networkexplorer.dll",
        "NlsData0000.dll"
    };
    std::vector<std::string> false_dlls = {
        "NetProjW.dll",
        "Ghofr.dll",
        "fg122.dll"
    };
    HMODULE lib_inst;

    for (auto &dll : real_dlls) {
        lib_inst = LoadLibraryA(dll.c_str());
        if (lib_inst == nullptr) {
            return true;
        }
        FreeLibrary(lib_inst);
    }

    for (auto &dll : false_dlls) {
        lib_inst = LoadLibraryA(dll.c_str());
        if (lib_inst != nullptr) {
            return true;
        }
    }

    return false;
}
```

Signature recommendations

Signature recommendations are not provided as it’s hard to say that evasion tehnique is being applied when libraries are just loaded.

1.5. Countermeasures

- for processes: exclude target processes from enumeration or terminate them;
- for libraries: exclude them from enumeration lists in PEB (http://www.codereversing.com/blog/archives/265);
- for functions in libraries: hook appropriate functions and compare their arguments against target ones;
- for libraries that must and must not be loaded: store a list of exclusions for libraries that should not be reported as loaded.

2. Check if specific artifacts are present in process address space (Sandboxie only)

Functions used:

- NtQueryVirtualMemory

Code sample

```
BOOL AmISandboxied(LPVOID
lpMinimumApplicationAddress, LPVOID lpMaximumApplicati
onAddress)
{
  BOOL IsSB = FALSE;
  MEMORY_BASIC_INFORMATION RegionInfo;
  ULONG_PTR i, k;
  SIZE_T Length = 0L;

  i = (ULONG_PTR)lpMinimumApplicationAddress;
  do {

    NTSTATUS Status = NtQueryVirtualMemory(GetCurrentP
rocess(),
                                           (PVOID)i,
                                           MemoryBasic
Information,

&RegionInfo,
                                           sizeof(MEMO
RY_BASIC_INFORMATION),
                                           &Length);
    if (NT_SUCCESS(Status)) {

      // Check if executable code
      if (((RegionInfo.AllocationProtect & PAGE_EXECUT
E_READWRITE) == PAGE_EXECUTE_READWRITE) &&
          ((RegionInfo.State & MEM_COMMIT) == MEM_COMM
IT)) {

        for (k = i; k < i + RegionInfo.RegionSize; k +
= sizeof(DWORD)) {
          if (
            (*(PDWORD)k == 'kuzt') ||
            (*(PDWORD)k == 'xobs')
            )
          {
            IsSB = TRUE;
```

```
            break;
          }
        }
      }
      i += RegionInfo.RegionSize;
    }
    else {
      i += 0x1000;
    }
  } while (i < (ULONG_PTR)lpMaximumApplicationAddress)
;

  return IsSB;
}
```

Take a look at VMDE project sources (https://github.com/hfiref0x/VMDE/blob/c1f439fbe58eaa83a09aa5804c4dd45de967337e/src/vmde/detect.c#L676).

Signature recommendations

Signature recommendations are not provided as it's hard to say what exactly is queried when memory buffer is being examined.

2.1. Countermeasures

Erase present artifacts from memory.

Credits

Credits go to open-source project from where code samples were taken:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)
- VMDE project on github (https://github.com/hfiref0x/VMDE)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions: Registry

Go back (..)

Contents

Registry detection methods

The principle of all the registry detection methods is the following: there are no such registry keys and values in usual host. However they exist in particular virtual environments.

Sometimes usual system may cause false positives when these checks are applied because it has some virtual machines installed and thus some VM artifacts are present in the system. Though in all other aspects such a system is treated clean in comparison with virtual environments.

Registry keys may be queries via WinAPI calls.

Functions used in kernel32.dll:

- RegOpenKey
- RegOpenKeyEx
- RegQueryValue
- RegQueryValueEx
- RegCloseKey
- RegEnumKeyEx

Functions above are wrappers on top of the following ntdll.dll functions:

- NtOpenKey

- NtEnumerateKey
- NtQueryValueKey
- NtClose

1. Check if particular registry paths exist

Take a look at title section to get the list of used functions.

Code sample

```
/* sample of usage: see detection of VirtualBox in the
table below to check registry path */
int vbox_reg_key7() {
    return pafish_exists_regkey(HKEY_LOCAL_MACHINE, "H
ARDWARE\\ACPI\\FADT\\VBOX__");
}

/* code is taken from "pafish" project, see references
on the parent page */
int pafish_exists_regkey(HKEY hKey, char * regkey_s) {
    HKEY regkey;
    LONG ret;

    /* regkey_s == "HARDWARE\\ACPI\\FADT\\VBOX__"; */
    if (pafish_iswow64()) {
        ret = RegOpenKeyEx(hKey, regkey_s, 0,
KEY_READ | KEY_WOW64_64KEY, &regkey);
    }
    else {
        ret = RegOpenKeyEx(hKey, regkey_s, 0,
KEY_READ, &regkey);
    }

    if (ret == ERROR_SUCCESS) {
        RegCloseKey(regkey);
        return TRUE;
    }
    else
        return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Signature recommendations

If the following function contains 2nd argument from the table column `Registry path`:

- NtOpenKey(..., registry_path, ...)

then it’s an indication of application trying to use the evasion technique.

Detections table

Check if the following registry paths exist:	
Detect	**Registry path**
[general]	HKLM\Software\Classes\Folder\shell\sandbox
Hyper-V	HKLM\SOFTWARE\Microsoft\Hyper-V
	HKLM\SOFTWARE\Microsoft\VirtualMachine
	HKLM\SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters
	HKLM\SYSTEM\ControlSet001\Services\vmicheartbeat
	HKLM\SYSTEM\ControlSet001\Services\vmicvss
	HKLM\SYSTEM\ControlSet001\Services\vmicshutdown
	HKLM\SYSTEM\ControlSet001\Services\vmicexchange
Parallels	HKLM\SYSTEM\CurrentControlSet\Enum\PCI\VEN_1AB8*
Sandboxie	HKLM\SYSTEM\CurrentControlSet\Services\SbieDrv
	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Sandboxie
VirtualBox	HKLM\SYSTEM\CurrentControlSet\Enum\PCI\VEN_80EE*
	HKLM\HARDWARE\ACPI\DSDT\VBOX__
	HKLM\HARDWARE\ACPI\FADT\VBOX__
	HKLM\HARDWARE\ACPI\RSDT\VBOX__
	HKLM\SOFTWARE\Oracle\VirtualBox Guest Additions
	HKLM\SYSTEM\ControlSet001\Services\VBoxGuest
	HKLM\SYSTEM\ControlSet001\Services\VBoxMouse
	HKLM\SYSTEM\ControlSet001\Services\VBoxService
	HKLM\SYSTEM\ControlSet001\Services\VBoxSF
	HKLM\SYSTEM\ControlSet001\Services\VBoxVideo
VirtualPC	HKLM\SYSTEM\CurrentControlSet\Enum\PCI\VEN_5333*
	HKLM\SYSTEM\ControlSet001\Services\vpcbus
	HKLM\SYSTEM\ControlSet001\Services\vpc-s3
	HKLM\SYSTEM\ControlSet001\Services\vpcuhub
	HKLM\SYSTEM\ControlSet001\Services\msvmmouf
VMware	HKLM\SYSTEM\CurrentControlSet\Enum\PCI\VEN_15AD*
	HKCU\SOFTWARE\VMware, Inc.\VMware Tools

	HKLM\SOFTWARE\VMware, Inc.\VMware Tools
	HKLM\SYSTEM\ControlSet001\Services\vmdebug
	HKLM\SYSTEM\ControlSet001\Services\vmmouse
	HKLM\SYSTEM\ControlSet001\Services\VMTools
	HKLM\SYSTEM\ControlSet001\Services\VMMEMCTL
	HKLM\SYSTEM\ControlSet001\Services\vmware
	HKLM\SYSTEM\ControlSet001\Services\vmci
	HKLM\SYSTEM\ControlSet001\Services\vmx86
	HKLM\SYSTEM\CurrentControlSet\Enum\IDE\CdRomNECVMWar_VMware_IDE_CD*
	HKLM\SYSTEM\CurrentControlSet\Enum\IDE\CdRomNECVMWar_VMware_SATA_CD*
	HKLM\SYSTEM\CurrentControlSet\Enum\IDE\DiskVMware_Virtual_IDE_Hard_D
	HKLM\SYSTEM\CurrentControlSet\Enum\IDE\DiskVMware_Virtual_SATA_Hard_
Wine	HKCU\SOFTWARE\Wine
	HKLM\SOFTWARE\Wine
Xen	HKLM\HARDWARE\ACPI\DSDT\xen
	HKLM\HARDWARE\ACPI\FADT\xen
	HKLM\HARDWARE\ACPI\RSDT\xen
	HKLM\SYSTEM\ControlSet001\Services\xenevtchn
	HKLM\SYSTEM\ControlSet001\Services\xennet
	HKLM\SYSTEM\ControlSet001\Services\xennet6
	HKLM\SYSTEM\ControlSet001\Services\xensvc
	HKLM\SYSTEM\ControlSet001\Services\xenvdb

In particular cases malware may enumerate sub-keys and check if a name of the sub-key contain some string instead of checking if the specified key exists.

For example: enumerate sub-keys of "HKLM\SYSTEM\ControlSet001\Services\" and search for "VBox" string.

2. Check if particular registry keys contain specified strings

Take a look at title section to get the list of used functions. Please note that case is irrelevant for these checks: it may be either upper or lower.

Code sample

```
/* sample of usage: see detection of VirtualBox in the
table below to check registry path and key values */
int vbox_reg_key2() {
    return pafish_exists_regkey_value_str(HKEY_LOCAL_M
ACHINE, "HARDWARE\\Description\\System", "SystemBiosVe
rsion", "VBOX");
}

/* code is taken from "pafish" project, see references
on the parent page */
int pafish_exists_regkey_value_str(HKEY hKey, char * r
egkey_s, char * value_s, char * lookup) {
    /*
        regkey_s == "HARDWARE\\Description\\System";
        value_s == "SystemBiosVersion";
        lookup == "VBOX";
    */

    HKEY regkey;
    LONG ret;
    DWORD size;
    char value[1024], * lookup_str;
    size_t lookup_size;

    lookup_size = strlen(lookup);
    lookup_str = malloc(lookup_size+sizeof(char));
    strncpy(lookup_str, lookup, lookup_size+sizeof(cha
r));
    size = sizeof(value);

    /* regkey_s == "HARDWARE\\Description\\System"; */
    if (pafish_iswow64()) {
        ret = RegOpenKeyEx(hKey, regkey_s, 0,
KEY_READ | KEY_WOW64_64KEY, &regkey);
    }
    else {
        ret = RegOpenKeyEx(hKey, regkey_s, 0,
KEY_READ, &regkey);
```

```
    }

    if (ret == ERROR_SUCCESS) {
        /* value_s == "SystemBiosVersion"; */
        ret = RegQueryValueEx(regkey, value_s, NULL, N
ULL, (BYTE*)value, &size);
        RegCloseKey(regkey);

        if (ret == ERROR_SUCCESS) {
            size_t i;
            for (i = 0; i < strlen(value); i++) { /*
case-insensitive */
                value[i] = toupper(value[i]);
            }
            for (i = 0; i < lookup_size; i++) { /*
case-insensitive */
                lookup_str[i] = toupper(lookup_str[i])
;
            }
            if (strstr(value, lookup_str) != NULL) {
                free(lookup_str);
                return TRUE;
            }
        }
    }

    free(lookup_str);
    return FALSE;
}
```

Credits for this code sample: pafish project (https://github.com/a0rtega/pafish)

Signature recommendations

If the following function contains 2nd argument from the table column `Registry path`:

- NtOpenKey(..., registry_path, ...)

and is followed by the call to the following function with 2nd argument from the table column `Registry key`:

- NtQueryValueKey(..., registry_item, ...)

then it’s an indication of application trying to use the evasion technique.

Detections table

Check if the following registry values contain the following strings (case ins		
Detect	**Registry path**	**Registry**
[general]	HKLM\HARDWARE\Description\System	SystemBiosDa
	HKLM\HARDWARE\Description\System\BIOS	SystemProduc
BOCHS	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	VideoBiosVer
Anubis	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	ProductID
	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion	ProductID
CwSandbox	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	ProductID
	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion	ProductID
JoeBox	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	ProductID
	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion	ProductID
Parallels	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	VideoBiosVer
QEMU	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	VideoBiosVer
	HKLM\HARDWARE\Description\System\BIOS	SystemManufa
VirtualBox	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 1\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 2\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	VideoBiosVer
	HKLM\HARDWARE\Description\System\BIOS	SystemProduc
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	FriendlyName
	HKLM\SYSTEM\ControlSet002\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet002\Services\Disk\Enum	FriendlyName
	HKLM\SYSTEM\ControlSet003\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet003\Services\Disk\Enum	FriendlyName

	HKLM\SYSTEM\CurrentControlSet\Control\SystemInformation	SystemProduc
	HKLM\SYSTEM\CurrentControlSet\Control\SystemInformation	SystemProduc
VMware	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 1\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 2\Scsi Bus 0\Target Id 0\Logical Unit Id 0	Identifier
	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	SystemBiosVe
	HKLM\HARDWARE\Description\System	VideoBiosVer
	HKLM\HARDWARE\Description\System\BIOS	SystemProduc
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	0
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	1
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet001\Services\Disk\Enum	FriendlyName
	HKLM\SYSTEM\ControlSet002\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet002\Services\Disk\Enum	FriendlyName
	HKLM\SYSTEM\ControlSet003\Services\Disk\Enum	DeviceDesc
	HKLM\SYSTEM\ControlSet003\Services\Disk\Enum	FriendlyName
	HKCR\Installer\Products	ProductName
	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall	DisplayName
	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall	DisplayName
	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall	DisplayName
	HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000	CoInstallers
	HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000	DriverDesc
	HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000	InfSection
	HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000	ProviderName
	HKLM\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000\Settings	Device Descr
	HKLM\SYSTEM\CurrentControlSet\Control\SystemInformation	SystemProduc
	HKLM\SYSTEM\CurrentControlSet\Control\Video\{GUID}\Video	Service
	HKLM\SYSTEM\CurrentControlSet\Control\Video\{GUID}\Video	Service
	HKLM\SYSTEM\CurrentControlSet\Control\Video\{GUID}\0000	Device Descr
Xen	HKLM\HARDWARE\Description\System\BIOS	SystemProduc

3. Check if VBAWarnings enabled

“Enable all macros” prompt in Office documents means the macros can be executed without any user interaction. This behavior is common for sandboxes. A malware can use that in order to check if it is running on a sandbox checking the flag in the registry keys SOFTWARE\Microsoft\Office<version>\Word\Security\VBAWarning while the version is between 12.0 to 19.0.

Code sample

```
// Function to check if VBScript warnings are enabled
in Office
bool IsVBScriptWarningEnabled() {
    HKEY hKey;
    LPCWSTR keyPath = L"SOFTWARE\\Microsoft\\Office\\<
Office_Version>\\Common\\Security";
    LPCWSTR valueName = L"VBAScriptWarnings";

    // Open the registry key
    LONG result = RegOpenKeyEx(HKEY_CURRENT_USER, keyP
ath, 0, KEY_READ, &hKey);
    if (result == ERROR_SUCCESS) {
        DWORD dwType;
        DWORD dwValue;
        DWORD dwSize = sizeof(DWORD);

        // Query the value of VBAScriptWarnings
        result = RegQueryValueEx(hKey, valueName,
NULL, &dwType, reinterpret_cast<LPBYTE>(&dwValue), &dw
Size);
        if (result == ERROR_SUCCESS && dwType == REG_D
WORD && dwValue == 1) {
            // VBScript warnings are enabled
            RegCloseKey(hKey);
            return true;
        }
        RegCloseKey(hKey);
    }
    return false;
}

int main() {
    if (IsVBScriptWarningEnabled()) {
        std::cout <<
"VBScript warnings are enabled in Office." << std::end
l;
    } else {
        std::cout << "VBScript warnings are not
```

```
  enabled in Office." << std::endl;
      }

      return 0;
  }
```

Countermeasures

Hook target functions and return appropriate results if indicators (registry strings from tables) are checked.

Credits

Credits go to open-source project from where code samples were taken:

- pafish project on github (https://github.com/a0rtega/pafish)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

Evasions/techniques/registry.html&t=Evasions:%20Registry - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/registry.html&text=Evasions:%20Registry - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/registry.html&title=Evasions:%20Registry)

Evasions: Timing

Go back (..)

Contents

Time-based sandbox evasion techniques

Sandbox emulation usually lasts a short time because sandboxes are heavy loaded with thousands of samples. Emulation time rarely exceeds 3-5 minutes. Therefore, malware can use this fact to avoid detection: it may perform long delays before starting any malicious activity.
To counteract this, sandboxes may implement features which manipulate time and execution delays. For example, the Cuckoo sandbox has a sleep skipping feature that replaces delays with a very short value. This should force the malware to start its malicious activity before an analysis timeout.

However, this can also be used to detect a sandbox.
There are also some differences in the time of execution of some instructions and API functions that can be used to detect a virtual environment.

Signature recommendations are not provided for this class of techniques as executing functions described in this chapter does not imply their usage for evasion purposes. It is hard

to differentiate between the code which aims to perform an evasion code and the one which uses the same functions with non-evasion intentions.

1. Delayed execution

Execution delays are used to avoid detection of malicious activity during the emulation time.

1.1. Simple delaying operation

Functions used:

- Sleep, SleepEx, NtDelayExecution
- WaitForSingleObject, WaitForSingleObjectEx, NtWaitForSingleObject
- WaitForMultipleObjects, WaitForMultipleObjectsEx, NtWaitForMultipleObjects
- SetTimer, SetWaitableTimer, CreateTimerQueueTimer
- timeSetEvent (multimedia timers)
- IcmpSendEcho
- select (Windows sockets)

While the use of most of these functions is obvious, we show examples of using the `timeSetEvent` function from Multimedia API and the `select` function from the Windows sockets API.

Code sample (delay using the “select” function)

```
int iResult;
DWORD timeout = delay; // delay in milliseconds
DWORD OK = TRUE;

SOCKADDR_IN sa = { 0 };
SOCKET sock = INVALID_SOCKET;

// this code snippet should take around Timeout
milliseconds
do {
    memset(&sa, 0, sizeof(sa));
    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = inet_addr("8.8.8.8");    //
we should have a route to this IP address
    sa.sin_port = htons(80); // we should not be able
to connect to this port

    sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock == INVALID_SOCKET) {
        OK = FALSE;
        break;
    }

    // setting socket timeout
    unsigned long iMode = 1;
    iResult = ioctlsocket(sock, FIONBIO, &iMode);

    iResult = connect(sock, (SOCKADDR*)&sa, sizeof(sa)
);
    if (iResult == false) {
        OK = FALSE;
        break;
    }

    iMode = 0;
    iResult = ioctlsocket(sock, FIONBIO, &iMode);
    if (iResult != NO_ERROR) {
        OK = FALSE;
```

```
            break;
        }

        // fd set data
        fd_set Write, Err;
        FD_ZERO(&Write);
        FD_ZERO(&Err);
        FD_SET(sock, &Write);
        FD_SET(sock, &Err);
        timeval tv = { 0 };
        tv.tv_usec = timeout * 1000;

        // check if the socket is ready, this call should
    take Timeout milliseconds
        select(0, NULL, &Write, &Err, &tv);

        if (FD_ISSET(sock, &Err)) {
            OK = FALSE;
            break;
        }

    } while (false);

    if (sock != INVALID_SOCKET)
        closesocket(sock);
```

Code sample (delay using the “timeSetEvent” function)

```
VOID CALLBACK TimerFunction(UINT uTimerID, UINT uMsg,
DWORD_PTR dwUser, DWORD_PTR dw1, DWORD_PTR dw2)
{
    bProcessed = TRUE;
}

VOID timing_timeSetEvent(UINT delayInSeconds)
{
    // Some vars
    UINT uResolution;
    TIMECAPS tc;
    MMRESULT idEvent;

    // We can obtain this minimum value by calling
    timeGetDevCaps(&tc, sizeof(TIMECAPS));
    uResolution = min(max(tc.wPeriodMin, 0), tc.wPerio
dMax);

    // Create the timer
    idEvent = timeSetEvent(
        delayInSeconds,
        uResolution,
        TimerFunction,
        0,
        TIME_ONESHOT);

    while (!bProcessed){
        // wait until our function finishes
        Sleep(0);
    }

    // destroy the timer
    timeKillEvent(idEvent);

    // reset the timer
    timeEndPeriod(uResolution);
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

1.2. Deferred execution using Task Scheduler

This method can be used both for delaying execution and evading sandbox tracking.

Code sample (PowerShell)

```
    $tm = (get-date).AddMinutes(10).ToString("HH:mm")
    $action = New-ScheduledTaskAction -Execute "some_malicious_app.exe"
    $trigger = New-ScheduledTaskTrigger -Once -At $tm
    Register-ScheduledTask TaskName -Action $action -Trigger $trigger
```

1.3. No suspicious actions until reboot

The idea behind this technique is that a sandbox doesn't reboot a virtual machine during the emulation of a malicious sample. The malware may just set up persistence using any of available methods and silently exit. Malicious actions are performed only after the system is rebooted.

1.4. Running only on certain dates

Malware samples may check the current date and perform malicious actions only on certain dates. For example, this technique was used in the Sazoora malware (https://www.cyphort.com/sazoora-dissecting-bundle-evasion-stealth/), which checks the current date and verifies if the day is either the 16th, 17th or 18th of a given month.

Example:

Countermeasures

Countermeasures for this class of evasion techniques should be comprehensive and include all described attack vectors. The implementation cannot be simple and its description deserves a separate article. Therefore, we only provide general recommendations here:

- Implement sleep skipping.
- System-wide dynamic time flow speed manipulation.
- Run emulation multiple times on different dates.

Although sleep skipping is already implemented in the Cuckoo sandbox, it is very easy to deceive it. Sleep skipping is disabled after a new thread or process is created to avoid sleep skipping detection. However, it can still be easily detected as shown below.

2. Sleep skipping detection

Techniques of this type are generally aimed at the Cuckoo monitor sleep skipping feature and other time-manipulation techniques that can be used in sandboxes to skip long delays performed by the malware.

2.1. Parallel delays using different methods

The idea behind the techniques is to perform different types of delays in parallel and to measure the elapsed time.

Code sample

```
DWORD StartingTick, TimeElapsedMs;
LARGE_INTEGER DueTime;
HANDLE hTimer = NULL;
TIMER_BASIC_INFORMATION TimerInformation;
ULONG ReturnLength;

hTimer = CreateWaitableTimer(NULL, TRUE, NULL);
DueTime.QuadPart = Timeout * (-10000LL);

StartingTick = GetTickCount();
SetWaitableTimer(hTimer, &DueTime, 0, NULL, NULL, 0);
do
{
    Sleep(Timeout/10);
    NtQueryTimer(hTimer, TimerBasicInformation, &Timer
Information, sizeof(TIMER_BASIC_INFORMATION), &ReturnL
ength);
} while (!TimerInformation.TimerState);

CloseHandle(hTimer);

TimeElapsedMs = GetTickCount() - StartingTick;
printf("Requested delay: %d, elapsed time: %d\n", Time
out, TimeElapsedMs);

if (abs((LONG)(TimeElapsedMs - Timeout)) > Timeout /
2)
    printf("Sleep-skipping DETECTED!\n");
```

In the code sample above, the delay timeout is set using the `SetWaitableTimer()` timer function. The `Sleep()` function is called in a loop until the timer timeout. In the Cuckoo sandbox, delays that are performed by the `Sleep()` function are skipped (replaced with a very short timeout) and the virtually elapsed time will be much higher than the requested timeout:

```
Requested delay: 60000, elapsed time: 1906975
Sleep-skipping DETECTED!
```

2.2. Measure time intervals using different methods

We need to perform a delay that will be skipped in a sandbox and to measure elapsed time using different methods. While the Cuckoo monitor hooks the `GetTickCount()`, `GetLocalTime()`, `GetSystemTime()` and makes them return the skipped time, we still can find methods to measure time that are not handled by the Cuckoo monitor.

Functions used:

- `GetTickCount64`
- `QueryPerformanceFrequency`, `QueryPerformanceCounter`
- `NtQuerySystemInformation`

Code sample (using “QueryPerformanceCounter” to measure elapsed time)

```
  LARGE_INTEGER StartingTime, EndingTime;
  LARGE_INTEGER Frequency;
  DWORD TimeElapsedMs;

  QueryPerformanceFrequency(&Frequency);
  QueryPerformanceCounter(&StartingTime);

  Sleep(Timeout);

  QueryPerformanceCounter(&EndingTime);
  TimeElapsedMs = (DWORD)(1000ll * (EndingTime.QuadPart
  - StartingTime.QuadPart) / Frequency.QuadPart);

  printf("Requested delay: %d, elapsed time: %d\n", Time
  out, TimeElapsedMs);

  if (abs((LONG)(TimeElapsedMs - Timeout)) > Timeout /
  2)
      printf("Sleep-skipping DETECTED!\n");
```

Code sample (using “GetTickCount64” to measure elapsed time)

```
  ULONGLONG tick;
  DWORD TimeElapsedMs;

  tick = GetTickCount64();
  Sleep(Timeout);
  TimeElapsedMs = GetTickCount64() - tick;

  printf("Requested delay: %d, elapsed time: %d\n", Time
  out, TimeElapsedMs);

  if (abs((LONG)(TimeElapsedMs - Timeout)) > Timeout /
  2)
      printf("Sleep-skipping DETECTED!\n");
```

We can also use our own implementation of GetTickCount to detect sleep skipping. In the next code sample, we acquire the tick count directly from the KUSER_SHARED_DATA structure. This way we can get the original tick count value even if the GetTickCount() function was hooked.

Code sample (getting the tick count from the KUSER_SHARED_DATA structure)

```
#define KI_USER_SHARED_DATA         0x7FFE0000
#define SharedUserData  ((KUSER_SHARED_DATA * const) KI_USER_SHARED_DATA)
#define MyGetTickCount() ((DWORD)((SharedUserData->TickCountMultiplier * (ULONGLONG)SharedUserData->TickCount.LowPart) >> 24))

// ...
StartingTick = MyGetTickCount();
Sleep(Timeout);
TimeElapsedMs = MyGetTickCount() - StartingTick;

printf("Requested delay: %d, elapsed time: %d\n", Time
out, TimeElapsedMs);

if (abs((LONG)(TimeElapsedMs - Timeout)) > Timeout / 2)
    printf("Sleep-skipping DETECTED!\n");
```

2.3. Get system time using different methods

This method is similar to the previous one. Instead of measuring intervals we try to obtain the current system time using different methods.

Code sample

```
SYSTEM_TIME_OF_DAY_INFORMATION  SysTimeInfo;
ULONGLONG time;
LONGLONG diff;

Sleep(60000); // should trigger sleep skipping
GetSystemTimeAsFileTime((LPFILETIME)&time);

NtQuerySystemInformation(SystemTimeOfDayInformation,
&SysTimeInfo, sizeof(SysTimeInfo), 0);
diff = time - SysTimeInfo.CurrentTime.QuadPart;
if (abs(diff) > 10000000) // differ in more than 1
second
    printf("Sleep-skipping DETECTED!\n);
```

2.4. Check if the delay value changes after calling a delay function

Sleep-skipping is usually implemented as a replacement of the delay value with a smaller interval. Let's look at the `NtDelayExecution` function. The delay value is passed to this function using a pointer:

```
NTSYSAPI NTSTATUS NTAPI
NtDelayExecution(
    IN BOOLEAN              Alertable,
    IN PLARGE_INTEGER       DelayInterval );
```

Therefore, we can check if the value of `DelayInterval` changes after the function execution. If the value differs from the initial value, the delay was skipped.

Code sample

```
LONGLONG SavedTimeout = Timeout * (-10000LL);
DelayInterval->QuadPart = SavedTimeout;
status = NtDelayExecution(TRUE, DelayInterval);
if (DelayInterval->QuadPart != SavedTimeout)
    printf("Sleep-skipping DETECTED!\n");
```

2.5. Use absolute timeout

For Nt-functions that perform delays we can use either a relative delay interval or an absolute time for timeout. A negative value for the delay interval means a relative timeout, and a positive value means an absolute timeout. High-level API functions such as `WaitForSingleObject()` or `Sleep()` operate with relative intervals. Therefore sandbox developers may not care about absolute timeouts and handle them incorrectly. In the Cuckoo sandbox such delays are skipped, but skipped time and ticks are counted incorrectly. This can be used to detect sleep skipping.

Code sample

```
void SleepAbs(DWORD ms)
{
    LARGE_INTEGER SleepUntil;

    GetSystemTimeAsFileTime((LPFILETIME)&SleepUntil);
    SleepTo.QuadPart += (ms * 10000);
    NtDelayExecution(TRUE, &SleepTo);
}
```

2.6. Get time from another process

Sleep skipping in the Cuckoo sandbox is not system-wide. Therefore, if there are performing delays, time moves with different speeds in the different processes. After a delay

we should synchronize the processes and compare the current time in the two processes. A big difference in measured time values indicates sleep skipping was performed.

The current version of the Cuckoo monitor disables sleep skipping after creating new threads or processes. Therefore, we should use a process creation method that is not tracked by the Cuckoo monitor, for example, using a scheduled task.

3. Get the current date and time from an external source (NTP, HTTP)

A sandbox may set different dates to check how the behavior of analyzed samples is changed depending on the date. The malware can use an external date and time source to prevent time manipulation attempts inside the VM. This method can also be used to measure time intervals, perform delays, and detect sleep skipping attempts. NTP servers, and the HTTP header “Date” can be used as an external source for the date and time. For example, the malware may connect to google.com to check the current date and use it as a DGA seed.

Countermeasures

Implement fake web infrastructure or spoof NTP data and HTTP headers returned by real servers. The returned/spoofed date and time should be synchronized with the date and time in a virtual machine.

4. Difference in time measurement in VM and hosts

The execution of some API functions and instructions may take different amounts of time in a VM and in the usual host systems. These peculiarities can be used to detect a virtual environment.

4.1. RDTSC (with CPUID to force a VM Exit)

Code sample

```
BOOL rdtsc_diff_vmexit()
{
    ULONGLONG tsc1 = 0;
    ULONGLONG tsc2 = 0;
    ULONGLONG avg = 0;
    INT cpuInfo[4] = {};

    // Try this 10 times in case of small fluctuations
    for (INT i = 0; i < 10; i++)
    {
        tsc1 = __rdtsc();
        __cpuid(cpuInfo, 0);
        tsc2 = __rdtsc();

        // Get the delta of the two RDTSC
        avg += (tsc2 - tsc1);
    }

    // We repeated the process 10 times so we make
sure our check is as much reliable as we can
    avg = avg / 10;
    return (avg < 1000 && avg > 0) ? FALSE : TRUE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

4.2. RDTSC (Locky version with GetProcessHeap and CloseHandle)

Code sample

```
#define LODWORD(_qw)    ((DWORD)(_qw))
BOOL rdtsc_diff_locky()
{
    ULONGLONG tsc1;
    ULONGLONG tsc2;
    ULONGLONG tsc3;
    DWORD i = 0;

    // Try this 10 times in case of small fluctuations
    for (i = 0; i < 10; i++)
    {
        tsc1 = __rdtsc();

        // Waste some cycles - should be faster than
CloseHandle on bare metal
        GetProcessHeap();

        tsc2 = __rdtsc();

        // Waste some cycles - slightly longer than
GetProcessHeap() on bare metal
        CloseHandle(0);

        tsc3 = __rdtsc();

        // Did it take at least 10 times more CPU
cycles to perform CloseHandle than it took to perform
GetProcessHeap()?
        if ((LODWORD(tsc3) - LODWORD(tsc2)) /
(LODWORD(tsc2) - LODWORD(tsc1)) >= 10)
            return FALSE;
    }

    // We consistently saw a small ratio of difference
between GetProcessHeap and CloseHandle execution times
    // so we're probably in a VM!
```

```
    return TRUE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Countermeasures

Implement RDTSC instruction "hooking." It is possible to make RDTSC a privileged instruction that can be called in kernel-mode only. Calling the "hooked" RDTSC in user-mode leads to an execution of our handler that can return any desired value.

5. Check the system last boot time using different methods

This technique is a combination of techniques described in Generic OS queries: Check if the system uptime is small (generic-os-queries.html#check-if-system-uptime) and WMI: Check the last boot time (wmi.html#check-last-boot-time) sections. Depending on a method used for getting system last boot time, the measured sandbox OS uptime can be too small (several minutes), or conversely, too big (months or even years), because the system is usually restored from a snapshot after the analysis starts.
We can detect a sandbox by comparing the two values for the last boot time, acquired through WMI and through NtQuerySystemInformation(SystemTimeOfDayInformation).

Code sample

```
bool check_last_boot_time()
{
    SYSTEM_TIME_OF_DAY_INFORMATION  SysTimeInfo;
    LARGE_INTEGER LastBootTime;

    NtQuerySystemInformation(SystemTimeOfDayInformatio
n, &SysTimeInfo, sizeof(SysTimeInfo), 0);
    LastBootTime = wmi_Get_LastBootTime();
    return (wmi_LastBootTime.QuadPart - SysTimeInfo.Bo
otTime.QuadPart) / 10000000 != 0; // 0 seconds
}
```

Countermeasures

- Adjust the KeBootTime value
- Reset the WMI repository or restart the "winmgmt" service after the KeBootTime adjustment

6. Call a potentially hooked delay function with invalid arguments

The second argument of the NtDelayExecution function is a pointer to the delay interval value. In the kernel-mode, the NtDelayExecution function validates this pointer and can also return the following values:

- STATUS_ACCESS_VIOLATION - If the value is not a valid user-mode address
- STATUS_DATATYPE_MISALIGNMENT - If the address is not aligned (DelayInterval & 3 != 0)

In a sandbox, the input arguments for NtDelayExecution and similar functions might not be handled correctly. If we call NtDelayExecution with an unaligned pointer for DelayInterval, normally it returns the STATUS_DATATYPE_MISALIGNMENT. However, in a sandbox, the value for DelayInterval may be copied to a new variable

without the appropriate checks. In this case, a delay is performed and the returned value will be STATUS_SUCCESS. This can be used to detect a sandbox.

Code sample

```
__declspec(align(4)) BYTE aligned_bytes[sizeof(LARGE_INTEGER) * 2];
DWORD tick_start, time_elapsed_ms;
DWORD Timeout = 10000; //10 seconds
PLARGE_INTEGER DelayInterval = (PLARGE_INTEGER)(aligned_bytes + 1); //unaligned
NTSTATUS status;

DelayInterval->QuadPart = Timeout * (-10000LL);
tick_start = GetTickCount();
status = NtDelayExecution(FALSE, DelayInterval);
time_elapsed_ms = GetTickCount() - tick_start;
// If the pointer is not aligned the delay should not be performed
if (time_elapsed_ms > 500 || status != STATUS_DATATYPE_MISALIGNMENT )
    printf("Sandbox detected\n");
```

On the other hand, if an inaccessible address is set for DelayInterval, the return code should be STATUS_ACCESS_VIOLATION. This can be used to detect a sandbox as well.

Code sample

```
if (NtDelayExecution(FALSE, (PLARGE_INTEGER)0) != STATUS_ACCESS_VIOLATION)
    printf("Sandbox detected");
```

If the DelayInterval argument is not verified before it is accessed, this may lead to an exception in the case of using an invalid pointer. For example, the next code leads the Cuckoo monitor to crash.

Code sample

```
NtDelayExecution(FALSE, (PLARGE_INTEGER)0xFFDF0000);
```

As stated earlier, normally this call should return STATUS_ACCESS_VIOLATION without causing an exception.

Countermeasures

Hooked functions should check arguments and return appropriate error codes if arguments are invalid.

Countermeasures

Countermeasures are present in the appropriate sub-sections above.

Credits

Credits go to open-source projects from where code samples were taken:

- al-khaser project on GitHub (https://github.com/LordNoteworthy/al-khaser)

Go back (..)

---

timing.html&title=Evasions:%20Timing - Evasion Techniques) (http://news.ycombinator.com/submitlink?u=https://evasions.checkpoint.com/src/Evasions/techniques/timing.html&t=Evasions:%20Timing - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/timing.html&text=Evasions:%20Timing - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/timing.html&title=Evasions:%20Timing)

Evasions: UI artifacts

Go back (..)

Contents

UI artifacts detection methods

Techniques described in this group abuse the fact that some windows' names are only present in virtual environment and not is usual host OS. Even more, host OS contains a lot of windows while VM and sandboxes prefer keeping opened windows at the minimum. Their quantity is checked and the conclusion is drawn whether it is a VM or not.

1. Check if windows with certain class names are present in the OS

Detections table

Check if windows with the following class names are present in the OS:	
Detect	**Class name**
VirtualBox	**VBoxTrayToolWndClass**
	VBoxTrayToolWnd

Code sample

```
BOOL vbox_window_class()
{
  HWND hClass =
FindWindow(_T("VBoxTrayToolWndClass"), NULL);
  HWND hWindow = FindWindow(NULL,
_T("VBoxTrayToolWnd"));

  if (hClass || hWindow)
    return TRUE;
  else
    return FALSE;
}
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

2. Check if top level windows' number is too small

As it was stated above, host OS contains a lot of windows while VMs and sandboxes strive to keep opened windows at possible minimum. Windows count is measured and the conclusion is drawn on whether it’s a VM or not.
In case there are too few windows in the OS, it could be an indication of virtual environment. Typical hosts have a lot (>10) top level windows.

Code sample

```
BOOL CALLBACK enumProc(HWND, LPARAM lParam)
{
    if (LPDWORD pCnt = reinterpret_cast<LPDWORD>(lPara
m))
        *pCnt++;
    return TRUE;
}

bool enumWindowsCheck(bool& detected)
{
    DWORD winCnt = 0;

    if (!EnumWindows(enumProc,LPARAM(&winCnt))) {
        std::cerr << "EnumWindows() failed\n";
        return false;
    }

    return winCnt < 10;
}
```

Signature recommendations

No signature recommendations are provided for this evasion group as it’s hard to tell that code aims to perform some evasion technique and not “legal” action.

Countermeasures

- versus windows with certain class names: Exclude windows with particular names from enumeration or modify these names.
- versus checking top level windows' number: Create fake windows in the system so that their number will not be small or equal to the predefined numbers.

Credits

Credits go to open-source project from where code samples were taken:

- al-khaser project on github (https://github.com/LordNoteworthy/al-khaser)

Though Check Point tool InviZzzible has them all implemented, due to modular structure of the code it would require more space to show a code sample from this tool for the same purposes. That's why we've decided to use other great open-source projects for examples throughout the encyclopedia.

Go back (..)

---

ui-artifacts.html&t=Evasions:%20UI%20artifacts - Evasion Techniques) (https://twitter.com/intent/tweet?via=_CPResearch_&url=https://evasions.checkpoint.com/src/Evasions/techniques/ui-artifacts.html&text=Evasions:%20UI%20artifacts - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/ui-artifacts.html&title=Evasions:%20UI%20artifacts)

Evasions: WMI

Go back (..)

Contents

WMI detection methods

Windows Management Interface (WMI) queries are another way to get OS and hardware information. WMI uses COM interfaces and their methods.

Background

Standard COM functions are used to process queries. They are called in the sequence described below and can be split into 6 steps.

1. COM initialization:

- CoInitialize/CoInitializeEx

2. Create the required interface instance:

- CoCreateInstance/CoCreateInstanceEx

3. Connect to the particular services via the interface instance with the following function:

- ConnectServer

4. Get methods of the services and set their arguments with these functions:

- Method (to get methods)
- Put (to set arguments)

5. Retrieve information from the services and execute the methods of the services with the functions below. The functions on the left are proxies for the functions on the right - which are called internally:

- ExecQuery -> IWbemServices_ExecQuery (retrieve information)
- ExecMethod -> IWbemServices_ExecMethod (execute method)

- ExecMethodAsync -> IWbemServices_ExecMethodAsync (execute method)

6. Examine the result of the query with the following functions:

- [enumerator]->Next
- [object]->Get

To see how the described theory is applied to practice, please check the examples below.

1. Generic WMI queries

As WMI provides another way to collect system information, it can be used to perform evasion techniques described in other articles, for example:

- Check if the number of processors is low (generic-os-queries.html#check-if-number-of-processors)
- Check if the hard disk size is small (generic-os-queries.html#check-if-hard-disk)
- Check if the MAC address is specific (network.html#check-if-mac-address-is-specific)
- Check if the CPU temperature information is available (hardware.html#check-if-cpu-temperature-information-is-available)

Code sample

```
/*
Check number of cores using WMI
*/
BOOL number_cores_wmi()
{
  IWbemServices *pSvc = NULL;
  IWbemLocator *pLoc = NULL;
  IEnumWbemClassObject *pEnumerator = NULL;
  BOOL bStatus = FALSE;
  HRESULT hRes;
  BOOL bFound = FALSE;

  // Init WMI
  bStatus = InitWMI(&pSvc, &pLoc);
  if (bStatus)
  {
    // If success, execute the desired query
    bStatus = ExecWMIQuery(&pSvc, &pLoc,
&pEnumerator, _T("SELECT * FROM Win32_Processor"));
    if (bStatus)
    {
      // Get the data from the query
      IWbemClassObject *pclsObj = NULL;
      ULONG uReturn = 0;
      VARIANT vtProp;

      // Iterate over our enumator
      while (pEnumerator)
      {
        hRes = pEnumerator->Next(WBEM_INFINITE, 1, &pc
lsObj, &uReturn);
        if (0 == uReturn)
          break;

        // Get the value of the Name property
        hRes = pclsObj->Get(_T("NumberOfCores"), 0, &v
tProp, 0, 0);
        if (V_VT(&vtProp) != VT_NULL) {
```

```
          // Do our comparaison
          if (vtProp.uintVal < 2) {
            bFound = TRUE; break;
          }

          // release the current result object
          VariantClear(&vtProp);
          pclsObj->Release();
        }
      }

      // Cleanup
      pEnumerator->Release();
      pSvc->Release();
      pLoc->Release();
      CoUninitialize();
    }
  }

  return bFound;
}


/*
Check hard disk size using WMI
*/
BOOL disk_size_wmi()
{
  IWbemServices *pSvc = NULL;
  IWbemLocator *pLoc = NULL;
  IEnumWbemClassObject *pEnumerator = NULL;
  BOOL bStatus = FALSE;
  HRESULT hRes;
  BOOL bFound = FALSE;
  INT64 minHardDiskSize = (80LL * (1024LL * (1024LL *
(1024LL))));

  // Init WMI
  bStatus = InitWMI(&pSvc, &pLoc);
```

```
  if (bStatus)
  {
    // If success, execute the desired query
    bStatus = ExecWMIQuery(&pSvc, &pLoc,
&pEnumerator, _T("SELECT * FROM Win32_LogicalDisk"));
    if (bStatus)
    {
      // Get the data from the query
      IWbemClassObject *pclsObj = NULL;
      ULONG uReturn = 0;
      VARIANT vtProp;

      // Iterate over our enumator
      while (pEnumerator)
      {
        hRes = pEnumerator->Next(WBEM_INFINITE, 1, &pc
lsObj, &uReturn);
        if (0 == uReturn)
          break;

        // Get the value of the Name property
        hRes = pclsObj->Get(_T("Size"), 0, &vtProp,
0, 0);
        if (V_VT(&vtProp) != VT_NULL) {

          // Do our comparaison
          if (vtProp.llVal < minHardDiskSize) { //
Less than 80GB
            bFound = TRUE; break;
          }

          // release the current result object
          VariantClear(&vtProp);
          pclsObj->Release();
        }
      }

      // Cleanup
      pEnumerator->Release();
      pSvc->Release();
```

```
        pLoc->Release();
        CoUninitialize();
      }
    }

    return bFound;
  }
```

Credits for this code sample: al-khaser project (https://github.com/LordNoteworthy/al-khaser)

Code sample (PowerShell)

```
(Get-CimInstance -ClassName Win32_BIOS -Property SerialNumber).SerialNumber
```

Signature recommendations

If the following function contains a 3rd argument from the table column "Query":

- IWbemServices_ExecQuery(..., query, ...)

then it's an indicator of the application trying to use the evasion technique.

Detections table

The following WMI queries may be used to detect virtual environment:			
Query	**Field**	**Value**	**De**
SELECT * FROM Win32_Processor	NumberOfCores	< 2	
	ProcessorId	[empty]	
SELECT * FROM Win32_LogicalDisk	Size	< 60GB	
SELECT * FROM Win32_BaseBoard	SerialNumber	None	**[gen**
	Version	None	
SELECT * FROM MSAcpi_ThermalZoneTemperature	CurrentTemperature	"Not supported"	
SELECT * FROM Win32_PnPEntity	DeviceId	PCI\VEN_80EE&DEV_CAFE	
		IDE\CDROOMVBOX	**Virt**
		IDE\DISKVBOX*	
		VEN_VMWARE	**VMwa**

		PROD_VMWARE_VIRTUAL	
SELECT * FROM Win32_NetworkAdapterConfiguration	MACAddress	08:00:27	**Virt**
		00:1C:42	**Para**
		00:05:69	**VMwa**
		00:0C:29	
		00:1C:14	
		00:50:56	
		00:16:E3	**XEN**
SELECT * FROM Win32_Bios	Serial Number	VMware-	**VMwa**
		0	**Virt**
	Version	INTEL - 6040000	**VMwa**
		BOCHS	**BOCH**
		PARALLELS	**Para**
		QEMU	**QEMU**
		VBOX	**Virt**
SELECT * FROM Win32_ComputerSystem	Model	VMware	**VMwa**
		VirtualBox	**Virt**
	Manufacturer	VMware	**VMwa**
		innotek GmbH	**Virt**
SELECT * FROM Win32_VideoController	AdapterCompatibility	VMware	**VMwa**
		Oracle Corporation	**Virt**
	Caption	VMware	**VMwa**
		VirtualBox	**Virt**
	Description	VMware	**VMwa**
		VirtualBox	**Virt**
	Name	VMware	**VMwa**
		VirtualBox	**Virt**
SELECT * FROM Win32_PointingDevice	Description	VMware	**VMwa**

Queries listed in the table are not the only ones possible, and are presented to give an idea of how they work and what information can be retrieved with these calls.

Countermeasures

Countermeasures depend on the particular checks implemented via the WMI method and they are the same as for the corresponding methods described in the relevant articles. Additionally, you must restart the “winmgmt” service.

2. Escape from tracking using WMI

WMI provides a way to create new processes and to schedule tasks. Sandboxes usually use the CreateProcessInternalW function hooking to track child processes. However, when you create the process using WMI the function CreateProcessInternalW is not called in the parent process. Therefore, the processes created using WMI may not be tracked by a sandbox and their behavior will not be recorded.

2.1. Start process using WMI

You can create a new process with WMI using the “Win32_Process” class with the method “Create”.

Code sample

```
// Initialize COM
CoInitializeEx(NULL, COINIT_MULTITHREADED);

//  Set general COM security levels
hres = CoInitializeSecurity(NULL, -1, NULL, NULL, RPC_
C_AUTHN_LEVEL_DEFAULT, RPC_C_IMP_LEVEL_IMPERSONATE, NU
LL, 0, NULL);
if (FAILED(hres) && hres != RPC_E_TOO_LATE)
    break;

// create an instance of WbemLocator
CoCreateInstance(CLSID_WbemLocator, NULL, CLSCTX_INPRO
C_SERVER, IID_IWbemLocator, (LPVOID*)&wbemLocator);
wbemLocator->ConnectServer(CComBSTR("ROOT\\CIMV2"), NU
LL, NULL, NULL, 0, NULL, NULL, &wbemServices);

// get Win32_Process object
wbemServices->GetObject(CComBSTR("Win32_Process"), 0,
NULL, &oWin32Process, &callResult);
wbemServices->GetObject(CComBSTR("Win32_ProcessStartup
"), 0, NULL, &oWin32ProcessStartup, &callResult);
oWin32Process->GetMethod(CComBSTR("Create"), 0, &oMeth
Create, &oMethCreateSignature);
oMethCreate->SpawnInstance(0, &instWin32Process);
oWin32ProcessStartup->SpawnInstance(0, &instWin32Proce
ssStartup);
// set startup information for process
instWin32ProcessStartup->Put(CComBSTR("CreateFlags"),
0, &varCreateFlags, 0);
instWin32Process->Put(CComBSTR("CommandLine"), 0, &var
CmdLine, 0);
instWin32Process->Put(CComBSTR("CurrentDirectory"),
0, &varCurDir, 0);
CComVariant varStartupInfo(instWin32ProcessStartup);
instWin32Process->Put(CComBSTR("ProcessStartupInformat
ion"), 0, &varStartupInfo, 0);
wbemServices->ExecMethod(CComBSTR("Win32_Process"), CC
```

```
omBSTR("Create"), 0, NULL, instWin32Process, &pOutPara
ms, &callResult);
```

Code sample is taken from InviZzzible tool (https://github.com/CheckPointSW/InviZzzible)

Signature recommendations

If one of the following functions is called with the 2nd argument “Win32_Process” and the 3rd argument “Create”:

- IWbemServices_ExecMethod(..., BSTR("Win32_Process"), BSTR("Create"), ...)
- IWbemServices_ExecMethodAsync(..., BSTR("Win32_Process"), BSTR("Create"), ...)

then it’s an indicator of the application trying to use the evasion technique.

Countermeasures

If you use a kernel-mode monitor, hook target functions or register callback on the process creation with PsSetCreateProcessNotifyRoutineEx.

2.2. Start process using Task Scheduler via WMI (Windows 7)

The technique is essentially the same as described in the “Deferred execution using Task Scheduler” (timing.html#deferred-execution-using-task-scheduler) section in the “Timing” chapter. WMI just provides another way to schedule a task.

You can create a new task with WMI using the “Win32_ScheduledJob” class with the method “Create”.

However, the “Win32_ScheduledJob” WMI class was designed to work with the AT command, which is deprecated since Windows 8.

In Windows 8 and higher, you can only create scheduled jobs with WMI if the registry key “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\Configuration” has a value “EnableAt”=”1” of type REG_DWORD. Therefore, this technique is unlikely to be found in the wild.

Code sample (VB)

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{imperson
ationLevel=Impersonate}!\\" & strComputer & "\root\cim
v2")
Set objSWbemDateTime = CreateObject("WbemScripting.SWb
emDateTime")
objSWbemDateTime.SetVarDate(DateAdd("n", 1, Now()))
Set objNewJob =
objWMIService.Get("Win32_ScheduledJob")
errJobCreate = objNewJob.Create("malware.exe", objSWbe
mDateTime.Value, False, , , True, "MaliciousJob")
```

Signature recommendations

If one of the following functions is called with the 2nd argument “Win32_ScheduledJob” and the 3rd argument “Create”:

- IWbemServices_ExecMethod(..., BSTR("Win32_ScheduledJob"), BSTR("Create"), ...)
- IWbemServices_ExecMethodAsync(..., BSTR("Win32_ScheduledJob"), BSTR("Create"), ...)

then it’s an indicator of the application trying to use the evasion technique.

Countermeasures

Use a kernel-mode monitor, and register callback on the process creation with PsSetCreateProcessNotifyRoutineEx.

3. Check the last boot time

If the last boot time is queried immediately after restoring a VM from a snapshot, the WMI database may contain the value saved at the moment the VM snapshot was created. If the snapshot was created a year ago, the calculated system uptime will be a year as well even if a sandbox updates the last boot time.

This fact can be used to detect a virtual machine restored from a snapshot. Also, any anomalies in the last boot time can be used as sandbox indicators:

- The system uptime is too big (months or even years)
- The system uptime is to small (less than several minutes)
- The last boot time obtained using other methods (timing.html#get-system-time) differs from the last boot time obtained using WMI

Code sample (VB)

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set colOperatingSystems = objWMIService.ExecQuery ("Select * from Win32_OperatingSystem")

For Each objOS in colOperatingSystems
    dtmBootup = objOS.LastBootUpTime
    dtmLastBootUpTime = WMIDateStringToDate(dtmBootup)
    dtmSystemUptime = DateDiff("n", dtmLastBootUpTime, Now)
    Wscript.Echo "System uptime minutes: " & dtmSystemUptime
Next

Function WMIDateStringToDate(dtm)
    WMIDateStringToDate =  CDate(Mid(dtm, 5, 2) & "/" & _
        Mid(dtm, 7, 2) & "/" & Left(dtm, 4) & " " & Mid (dtm, 9, 2) & ":" & _
        Mid(dtm, 11, 2) & ":" & Mid(dtm, 13, 2))
End Function
```

Code sample is taken from Microsoft Docs (https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-tasks--desktop-management)

Signature recommendations

If the following function is called with the 3rd argument BSTR(“Win32_OperatingSystem”):

- IWbemServices_ExecQuery(..., BSTR("Win32_OperatingSystem"), ...)

then it’s a possible indicator of the application trying to use the evasion technique.

Countermeasures

- Adjust the KeBootTime value
- Reset the WMI repository or restart the "winmgmt" service after you adjust the KeBootTime value

4. Check the network adapters last reset time

We need to check if there are any adapters that were last reset a long time ago. This may indicate the application is running in a virtual machine restored from a snapshot.

Code sample (VB)

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set colOperatingSystems = objWMIService.ExecQuery ("Select * from Win32_NetworkAdapter")

For Each objOS in colNetworkAdapters
    dtmLastReset = objOS.TimeOfLastReset
    dtmLastResetTime = WMIDateStringToDate(dtmLastReset)  'WMIDateStringToDate function from the previous example
    dtmAdapterUptime = DateDiff("n", dtmLastResetTime, Now)
    Wscript.Echo "Adapter uptime minutes: " & dtmAdapterUptime
Next
```

Signature recommendations

If the following function is called with the 3rd argument BSTR(“Win32_OperatingSystem”):

- IWbemServices_ExecQuery(..., BSTR("Win32_NetworkAdapter"), ...)

then it’s a possible indicator of the application trying to use the evasion technique.

Countermeasures

- Ensure an adequate last reset time for the network adapters
- Reset the WMI repository or restart the "winmgmt" service

Countermeasures

Countermeasures are presented in the appropriate sub-sections above.

Credits

- al-khaser project on GitHub (https://github.com/LordNoteworthy/al-khaser)
- Microsoft Docs - WMI Tasks: Desktop Management (https://docs.microsoft.com/en-us/windows/win32/wmisdk/)

Go back (..)

---

evasions.checkpoint.com/src/Evasions/techniques/wmi.html&text=Evasions:%20WMI - Evasion Techniques) (https://www.linkedin.com/shareArticle?mini=true&url=https://evasions.checkpoint.com/src/Evasions/techniques/wmi.html&title=Evasions:%20WMI)